A Probabilistic Approach for Reducing the Search Cost in Binary Decision Trees

Athanasios Rontogiannis and Nikitas J. Dimopoulos

Abstract— In many complex problems a particular decision making procedure is often required in order for a final solution to be found. Such a procedure may consist of a large number of intermediate steps where "local" decisions must be taken and can be sometimes represented as a decision tree. When that structure is used the final solutions obtained vary depending on the available information. However, if the same model is applied many times, experimental data can be collected and observations on the acquired knowledge can be made. In this work, we present a probabilistic approach for reducing the number of decisions (tests) that are required in a particular decision making situation. Specifically, we consider that a problem is structured as a complete binary balanced decision tree the interior nodes of which correspond to decision points; the paths of the tree represent different decision making processes. By assuming that there exists sufficient probabilistic information concerning the decisions at the interior nodes, we propose techniques in order to minimize the average number of these decisions when we search for a final solution.

I. Introduction

People often cope with problems in which they have to make decisions under conditions of uncertainty. This is the case because some components of the problem under consideration are only partially known or the cost associated with obtaining all the necessary information is high. When the decision maker faces such a situation he must base his decisions and conclusions on his experience or on available information. The whole procedure can be facilitated when it is properly structured in a way that clarifies the particularities of the problem. Decision trees constitute an effective representation for modeling some aspects of human reasoning. Different paths of the decision tree express different rationales that can be followed in order for a solution to be found. Often decisions at the interior nodes of the decision tree are taken based on incomplete information. A utility function can then be constructed and the decisions are taken so as to optimize such a utility function.

In [1]–[3], the interior nodes of the tree are divided into *decision nodes* where the decision maker is in control of the choice and *chance nodes* which correspond to events (often called "states of nature") that lie outside the control of the human expert. The main goal of the decision maker is to follow a path which maximizes an expected utility function that is related to the problem parameters. In [4]–[7] and [8] the decision tree structure is used as a classification tool. The leaves of the tree constitute a disjoint set of classes while at each interior node the value of a particular attribute is checked. In [4], [5], the author describes how such a tree can be constructed from a set of initial objects that are distinguished through the values of different attributes. Then, that tree can be used for the classification of new objects.

In our work a different approach to decision problems is adopted. Specifically, we consider complete binary balanced decision trees and assume (as in [8]) that there exist sufficient probabilistic (statistical) information available concerning the decisions at the interior nodes. Our goal is to develop a model in which, relying on the given information, we reduce the number of decisions that are required when we search for a conclusion. That can be achieved by selecting the nodes at which an explicit decision, based on the state of the environment, is not taken but instead the branch of the tree that is to be followed is determined based on the likelihood, estimated from past experience, that the outcome is the correct one. Reducing the number of decisions taken is important because each decision requires human intervention or "expensive" observation. Additionally, this process seems to emulate the approach adopted by human experts who, based on their experience, may assume that the outcome of certain decision nodes have been "fixed" for the majority of cases.

Such an approach has the effect that we may sometimes end up with wrong conclusions. We can then search for the correct path by appropriately backtracking into the tree. We must therefore choose both the nodes where we will not test for the value but we assume that the outcome will be the most probable one, and the proper backtracking method in case that a wrong conclusion is reached, so that the average number of tests performed until a correct conclusion is reached, is minimized. We have formulated a gain function and we propose techniques in order to maximize this gain.

After introducing the problem in Section II, the required steps for the formulation of the gain function are described. This function is subject to maximization. In Section III, the notion of the cost associated with the backtracking procedure, is defined and its general expression is derived for a single path of the decision tree. In Section IV, we present two techniques for selecting the interior nodes of the tree at which a decision is based on our probabilistic information. The two techniques are compared to each other in terms of the value of the gain achieved. Finally, concluding remarks are presented in Section V.

Manuscript received June 11, 1993; revised March 14, 1994. This work was supported by a grant from the Canadian Cable Labs Fund.

The authors are with the Department of Electrical and Computer Engineering, University of Victoria, Victoria, British Columbia, V8W 3P6 Canada.

IEEE Log Number 9405711.

II. The Gain Function

We consider a complete binary balanced tree of depth n. We assume that the interior nodes of the tree are decision nodes at which an attribute test takes place as described in [5]. There exist only two possible outcomes for each attribute test: either the value of the attribute is verified or not. In other words, we can assume that each interior node of the tree contains a question with only two possible answers: "yes" or "no". The "yes" answer can be arbitrarily assigned to the left "child" and the "no" answer to the right "child" of a particular node. The leaves of the tree constitute a set of distinct final conclusions to the problem that is represented by the given tree structure. In our model, we make the assumption that the arbitrary interior node k of the three is assigned two probabilities, P_{ky}, P_{kn}, (see Fig. 1), which represent the probability of occurrence of the two possible test outcomes. We have to emphasize that these probabilities are conditional probabilities. That is, P_{ky} is the probability that the answer in node k is positive given that we have followed the path that leads to this node. For example, in Fig. 5, P_{5n} can be written as

$$P_{5n} = P(E|E_1, E_2) \tag{1}$$

where E is the statement that the answer at node 5 is "no" and E_1, E_2 are the facts that the answers at nodes 1 and 2 are "yes" and "no" respectively.

The probabilities of the interior nodes of the tree sometimes carry very important information. For instance, if $P_{ky} = 0.9$, our belief is very strong that the outcome of the test at node k is positive. Thus, when we reach that node, we may decide not to perform any test but instead follow the left branch since such a decision is correct with a very high probability. By avoiding a test we have a gain whose value depends on the importance of the test under consideration. In our analysis, all the tests are assumed to have the same significance and therefore we can assign a weight of 1 to each of them. The aim of our work is to partition the interior nodes of the tree into nodes where the test occurs and nodes where it does not, in such a manner that the gain is maximized. In other words, we will attempt to minimize the number of nodes where the prescribed test is performed and yet the conclusion reached is correct.

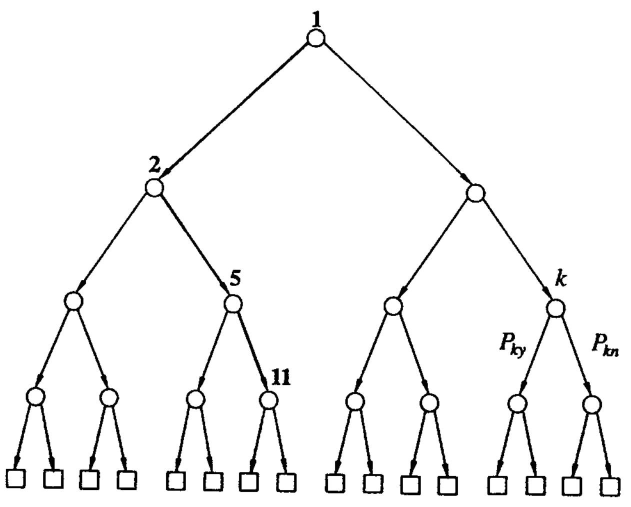

Fig. 1. Example Tree.

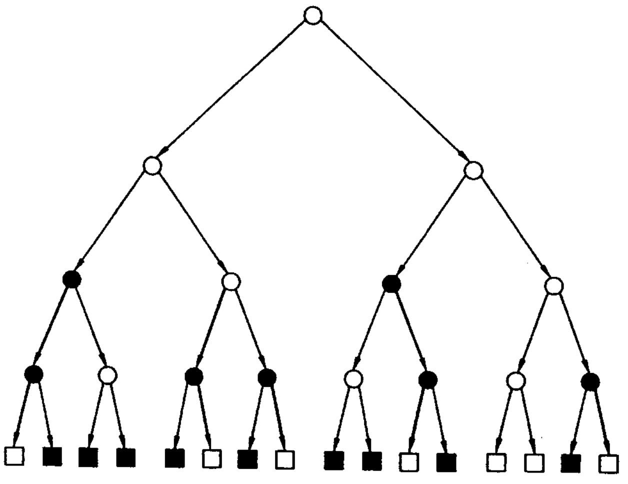

Fig. 2. Representation of statistical and non-statistical nodes.

By partitioning the interior nodes as described above, we may sometimes end up with wrong conclusions (final nodes). That is possible because the decisions which are based on probabilistic information may not be correct, resulting in wrong final nodes. In such a case we assume that we are able to backtrack and search for the correct path. We have to note that when a conclusion is reached using our model, the truth of that conclusion must be verified. Therefore, there may be a cost associated with that process. In order to simplify our model we assume that we are able to distinguish between correct and incorrect final conclusions without any cost.

A. Derivation of the Gain Function

Before we proceed to the derivation of the gain function we give the following definition of a *statistical node.*

Definition 1: An interior node of the tree is called statistical when the test is not performed at that node but instead the decision is based on the probabilistic information associated with the node. The branch of the statistical node which is followed is called a statistical branch. All the other branches are called non-statistical.

By partitioning the interior nodes of the tree into statistical and non-statistical ones, we separate the set of final nodes F into two disjoint subsets: the subset of the reachable nodes F_1 and the subset of the non-reachable nodes F_2. This can become more obvious by considering the tree of Fig. 2. The tree has depth $n = 4$. The solid circles represent the statistical nodes while the statistical branches are emboldened. The solid squares constitute the non-reachable final nodes. In general, a path which corresponds to a reachable final node has the following characteristics

1) The number m_j of its statistical nodes.
2) The probability P_{Pj} that the path is going to be followed. P_{Pj} can be expressed as the product of the probabilities of the non-statistical branches of the path ([9]).
3) The probability P_{Cj} that the path is correct given that it has been followed. P_{Cj} is the product of the probabilities of the statistical branches of the path ([9]).
4) The backtracking cost B_j. The backtracking cost represents the number of tests that take place after the path j turns out to be wrong and before the correct path is found. Its general expression is given in the next section.

It was stated before that all the tests of the tree are equally significant and therefore we can assign to each of them a weight of 1. Thus, we define the cost as the number of nodes where a test is performed. As we traverse the tree we have two choices: either we use the statistical data that we have collected or not. In the latter case the cost will be expressed as

$$L_j = n \tag{2}$$

where L_j is the cost associated with the j-th conclusion and n is the depth of the decision tree assuming a complete binary balanced decision tree.

Equation (2) holds because we are following a path by asking the questions in all the nodes of the path. Furthermore we are certain that the path is correct, that is, backtracking is not required. Let us now consider the former case. Assume that we have followed a path and we have reached a final node j. Let us also assume that m_j out of n nodes of the path are statistical nodes. This means that in m_j out of n nodes of the path (we do not consider the last node which is not a decision node), probabilistic information has been used to make the decision. We have two exclusive and exhaustive events: a) Path j that we followed is correct and b) Path j is wrong. If the probabilities of these two events are P_{Cj} and P_{Wj} respectively then

$$P_{Cj} + P_{Wj} = 1 \tag{3}$$

The cost in the case of a correct path is given by

$$L_{Cj} = n - m_j \tag{4}$$

As it was mentioned before, when we end up with a wrong path, we backtrack into the tree in order to find the correct path. Such a procedure results in a cost which will be called the *backtracking cost* for path j and denoted as B_j. The cost in the case of a wrong path is therefore

$$L_{Wj} = n - m_j + B_j \tag{5}$$

Assume, for example, that after a wrong final conclusion we back up to the first statistical node of the path and then we traverse the tree by asking all the questions at the nodes we meet. This way the correct conclusion is reached because the path from the root node to the first statistical node is certainly correct. If h_j is the height of that node (where n is the height of the root and 0 the height of node j), L_{Wj} can be written as

$$L_{Wj} = n - m_j + h_j \tag{6}$$

Equation (6) holds because h_j more tests take place after backtracking for the example considered.

In general, if we weight the costs L_{Cj} and L_{Wj} given in (4), (5) with the corresponding probabilities P_{Cj} and P_{Wj} we can express the cost of using the probabilistic information for the path j as follows

$$\begin{aligned} L_{Sj} &= P_{Cj} \cdot L_{Cj} + P_{Wj} \cdot L_{Wj} \\ &= P_{Cj} \cdot (n - m_j) + (1 - P_{Cj}) \cdot (n - m_j + B_j) \end{aligned} \tag{7}$$

If we simplify (7) we end up with the following expression for L_{Sj}

$$L_{Sj} = (n - m_j) + B_j \cdot (1 - P_{Cj}) \qquad (8)$$

From (2) and (8) we can calculate the gain in the case of a single path of the tree as follows

$$G_j = L_j - L_{Sj} = m_j - B_j \cdot (1 - P_{Cj}) \qquad (9)$$

We must note that the gain G_j can be negative. A negative gain indicates that the cost of using the statistical information is more than the cost of determining the path at each node.

The tree gain which represents the average number of questions that are not asked as we search for the correct conclusions, can be computed by weighting the gain given in (9) with the posterior probability P_{Pj} that the path j is followed and then building up the following sum

$$G = \sum_{j \in F_1} P_{Pj} \cdot G_j = \sum_{j \in F_1} P_{Pj} \cdot [m_j - B_j \cdot (1 - P_{Cj})] \qquad (10)$$

For a particular partition of the interior nodes of the tree, all the components of (10) except for B_j are known and well-defined. Therefore, in order for the gain G to be maximized, two specific tasks must be accomplished: a) the optimal partition of the nodes must be found and b) the backtracking cost B_j for each reachable final node resulting from that partition must be minimized. The general expression for B_j is given in the next section. In Section IV two partitioning techniques are proposed. The problem of finding the optimal partition appears to be a difficult and complicated one.

III. Derivation of the Backtracking Cost

As stated in the previous section, when we end up with a wrong diagnosis we backtrack and search for the correct path. Such a procedure results in a cost which is called the backtracking cost. The backtracking cost represents the number of tests that take place after a wrong path is diagnosed and before the correct one is found. In order to simplify our model we assume that during the search for the correct path, after backtracking, the probabilistic information that is available at the nodes of the tree is not taken into account any longer.

From (10) it is clear that for a specific partitioning of the interior nodes, the gain G is maximized if the backtracking cost B_j is minimized for every reachable node j. In this section a single path of the tree is considered and the general expression of its backtracking cost is derived. The results can be applied to each path of the set F_1. To simplify our notation the subscript j which refers to a specific path j is not used in the analysis below.

A. Probabilistic Considerations for a Single Path

Let us consider an arbitrary path in the tree which ends at a reachable final node. Let $A_1, A_2, \cdots, A_n$ be the n nodes in the path (we do not include the final node) and let A_1 be the root. Assume that the path contains m statistical nodes $A_{k_1}, A_{k_2}, \cdots, A_{k_m}$; $(1 \leq m \leq n)$, with heights $h_1, h_2, \cdots, h_m$ respectively. We define as the height h of a node A the number of nodes between A and the leaf node of the path; this count include node A. If we follow that path and the final diagnosis turns out wrong, it happened because a wrong decision was made in at least one statistical node. We are most interested in the first statistical node of the path where a wrong decision occurred. This is so because, after passing that node, we are already on a wrong path. Let us now define the following facts

- W: The arbitrary path that we followed is wrong.
- A_i: A correct decision was made at node A_i of the path.
- $\overline{A}_i$: A wrong decision was made at node A_i of the path.

We form the following conditional probability

$$P(A_1, A_2, \cdots, A_{i-1}, \overline{A}_i | W) \qquad i = 1, 2, \cdots, n \qquad (11)$$

The last expression gives the probability that the path is wrong because a wrong decision was taken at node A_i and all decisions preceding that were correct. Obviously, if A_i is a non-statistical node

$$P(A_1, A_2, \cdots, A_{i-1}, \overline{A}_i | W) = 0$$

On the other hand, for an arbitrary statistical node A_{k_i} of the path, the expression

$$P(A_1, A_2, \cdots, A_{k_i - 1}, \overline{A}_{k_i} | W) \qquad i = 1, 2, \cdots, m \qquad (12)$$

gives the probability that the first wrong decision occurred at node A_{k_i}. In order to calculate this probability we apply the well-known inversion formula ([3]). We have

$$P(A_1, \cdots, A_{k_i - 1}, \overline{A}_{k_i} | W) = \frac{P(W | A_1, \cdots, A_{k_i - 1}, \overline{A}_{k_i}) \cdot P(A_1, \cdots, A_{k_i - 1}, \overline{A}_{k_i})}{P(W)} \qquad (13)$$

Clearly,

$$P(W | A_1, \cdots, A_{k_i - 1}, \overline{A}_{k_i}) = 1 \qquad i = 1, 2, \cdots, m \qquad (14)$$

because the path is certainly wrong if a wrong decision was made at any node. Furthermore,

$$P(W) = 1 - P(C)$$

and $P(C)$ is the probability that the path is correct given that it has been followed, defined in Section II. Therefore, (13) can be rewritten as follows

$$P(A_1, \cdots, A_{k_i - 1}, \overline{A}_{k_i} | W) = \frac{P(A_1, \cdots, A_{k_i - 1}, \overline{A}_{k_i})}{1 - P(C)} \qquad (15)$$

In order to compute the probability $P(A_1, \cdots, A_{k_i - 1}, \overline{A}_{k_i})$ we apply the chain rule ([3]).

$$P(A_1, \cdots, A_{k_i - 1}, \overline{A}_{k_i}) = P(\overline{A}_{k_i} | A_1, \cdots, A_{k_i - 1}) \cdot \cdots \cdot P(A_2 | A_1) \cdot P(A_1) \qquad (16)$$

But for a non-statistical node A_r, $1 \leq r \leq k_i - 1$

$$P(A_r | A_1, \cdots, A_{r-1}) = 1 \qquad (17)$$

Substituting (17) and (16) for every non-statistical node we get

$$\begin{aligned} P(A_1, \cdots, A_{k_i - 1}, \overline{A}_{k_i}) &= P(\overline{A}_{k_i} | A_1, \cdots, A_{k_i - 1}) \cdot P(A_{k_{i-1}} | A_1, \cdots, A_{k_{i-1} - 1}) \cdots \\ &\quad \cdot P(A_{k_2} | A_1, \cdots, A_{k_2 - 1}) \cdot P(A_{k_1} | A_1, \cdots, A_{k_1 - 1}) \end{aligned} \qquad (18)$$

that is, $P(A_1, \cdots, A_{k_i - 1}, \overline{A}_{k_i})$ depends exclusively on the probabilities assigned to the statistical nodes which are located before node A_{k_i} in the path. More specifically, it is derived by multiplying the probabilities of all the statistical branches that are before A_{k_i} and then by multiplying again the outcome with the probability of the non-statistical branch of A_{k_i}. All these probabilities are known from our initial model and therefore the expression in (15) can be calculated. By applying the procedure which has been described so far, the probability that the first wrong decision occurred at a particular statistical node can be computed (see (12)). We note that

$$\sum_{i=1}^{m} P(A_1, \cdots, A_{k_i - 1}, \overline{A}_{k_i} | W) = 1 \qquad (19)$$

because the probabilities that appear in the above summation correspond to a set of exhaustive events.

We now define P_r, $r = 1, 2, \cdots, m$, to be the probability that the first wrong decision took place either at node A_{k_r} or at a statistical

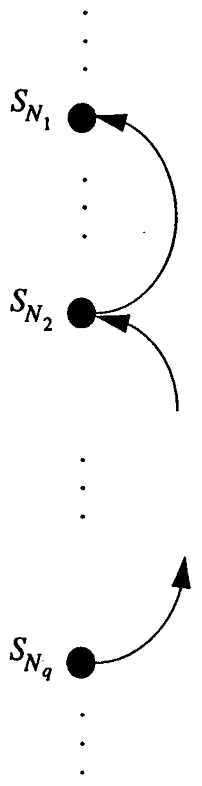

Fig. 3. Illustration of the Backtracking Procedure for a Single Path.

node which follows A_{k_r} in the path $(A_{k_{r+1}}, \cdots, A_{k_m})$. Then P_r can be expressed as follows

$$P_r = \sum_{i=r}^{m} P(A_1, \cdots, A_{k_i - 1}, \overline{A}_{k_i} | W) \tag{20}$$

If we back up to node A_{k_r} after a wrong final diagnosis, P_r expresses the probability of successful backtracking, that is, the probability that we are going to find the correct final node after going back to node A_{k_r} and traversing the tree from this point by performing all the tests at all the nodes we meet. Obviously $P_1 = 1$. This is the case of backtracking to node A_{k_1}.

B. The Backtracking Cost

To simplify the notation, let $S_1, S_2, \cdots, S_m$ denote the m statistical nodes in a given path. It is clear from the previous analysis that only these nodes need to be considered as backtracking points. Furthermore, a particular statistical node S_i, $1 \leq i \leq m$, has two characteristics

- Its *height* h_i, which also represents the number of tests between that node and a leaf node.
- The *probability of successful backtracking* P_i. P_i is the probability that the first wrong decision took place either at node S_i or at a statistical node which follows S_i in the path $(S_{i+1}, \cdots, S_m)$.

Out of these m statistical nodes we want to choose q nodes in such a manner that the backtracking cost is minimized when we backtrack to these q nodes. We note that after backtracking to a particular node, we traverse the tree from that point by explicitly performing the test at all the subsequent nodes. If we end up again with a wrong final node we back up to the next statistical node of the sequence and we repeat the same procedure.

We will prove the following theorem in which the general expression of the backtracking cost is derived (see also Fig. 7)

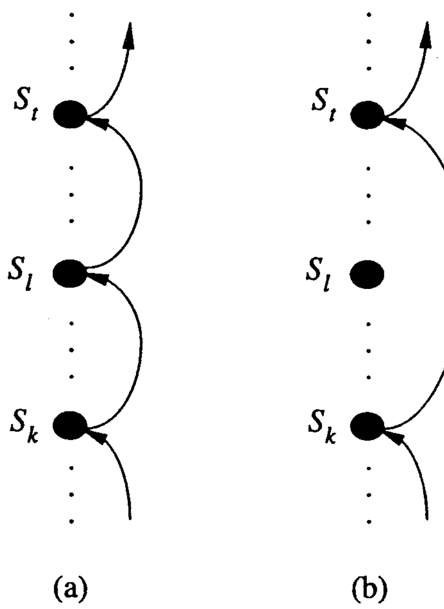

Fig. 4. The Backtracking Procedure used to calculate (a) B and (b) B' in lemma 1.

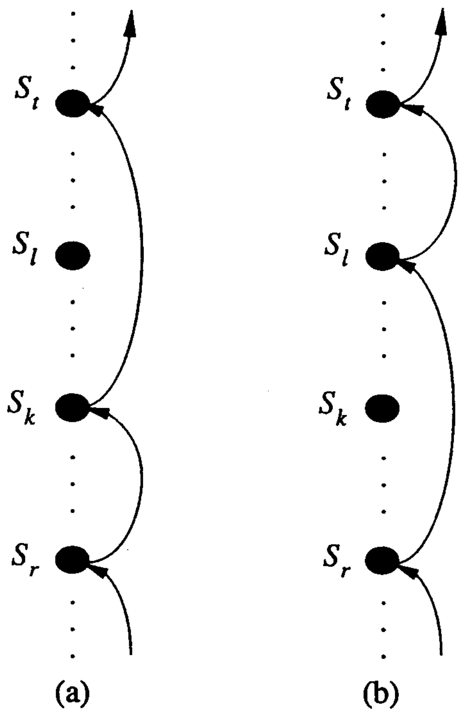

Fig. 5. The Backtracking Procedure used to calculate (a) B and (b) B' in lemma 2.

Theorem 1: Given a path composed of n nodes $A_1, A_2, \cdots, A_n$ denote by $S_1, S_2, \cdots, S_m$ the m statistical nodes of the path and by $S_{N_1}, S_{N_2}, \cdots, S_{N_q}$ the q statistical nodes at which we back up. S_{N_q} is the first node at which we backtrack (i.e., the node closest to the leaf) and $S_{N_1} \equiv S_1$. The backtracking cost B can then be written in the following form

$$B = \sum_{i=1}^{q} h_{N_i} \cdot (1 - P_{N_{i+1}}), \quad \text{where} \quad P_{N_{q+1}} = 0 \tag{21}$$

Proof: As was stated previously, the cost B gives the number of nodes where the test at the node is performed, during the backtracking

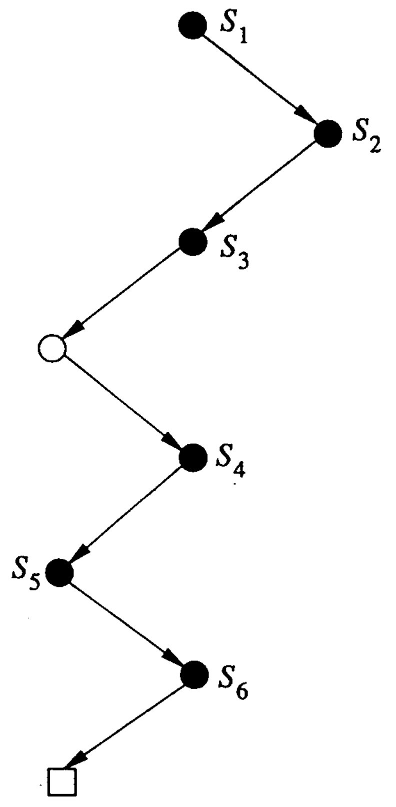

Fig. 6. The Path of Example 1.

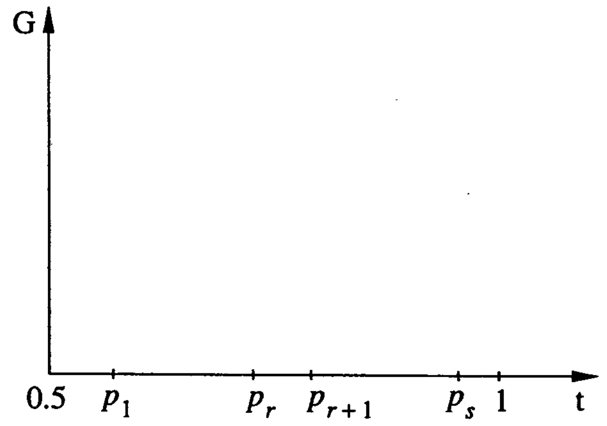

Fig. 7. Representation of the Subintervals.

procedure. If we assume that we back up to nodes $S_{N_1}, S_{N_2}, \cdots, S_{N_q}$ as was described before, the cost B can be expressed as follows

$$\begin{aligned} B = &P_{N_q} \cdot h_{N_q} + (1 - P_{N_q}) \\ &\cdot \{h_{N_q} + P'_{N_{q-1}} \cdot h_{N_{q-1}} + (1 - P'_{N_{q-1}}) \\ &\cdot \{h_{N_{q-1}} + \cdots (1 - P'_{N_2}) \cdot \{h_{N_2} + h_{N_1} \cdot P'_{N_2}\} \cdots\} \end{aligned} \quad (22)$$

■

In (22), the term P'_{N_k}, $2 \leq k \leq q-1$ represents the probability that the first wrong decision took place at node S_{N_k}, denoted as fact A, given that it did not take place at the nodes $S_{N_{k+1}}, \cdots, S_{N_q}$, denoted as fact D. According to the definition of the probability of successful backtracking we have

$$P'_{N_k} = P(A|D) = \frac{P(A, D)}{P(D)} = \frac{P_{N_k} - P_{N_{k+1}}}{1 - P_{N_{k+1}}} \quad (23)$$

Furthermore

$$1 - P'_{N_k} = \frac{1 - P_{N_k}}{1 - P_{N_{k+1}}} \quad (24)$$

If we substitute (23) and (24) into (22), we observe that the denominators of the above fractions cancel out. Thus the cost B can be rewritten as

$$\begin{aligned} B = &P_{N_q} \cdot h_{N_q} + (1 - P_{N_q}) \cdot h_{N_q} + (P_{N_{q-1}} - P_{N_q}) \cdot h_{N_{q-1}} \\ &+ (1 - P_{N_{q-1}}) \cdot h_{N_{q-1}} + \cdots + (P_{N_1} - P_{N_2}) \cdot h_{N_2} \\ &+ (1 - P_{N_1}) \cdot h_{N_2} + (1 - P_{N_2}) \cdot h_{N_1} \\ = &h_{N_1} + (1 - P_{N_1}) \cdot h_{N_{q-1}} + \cdots + (1 - P_{N_3}) \cdot h_{N_2} \\ &+ (1 - P_{N_2}) \cdot h_{N_1} \Leftrightarrow \end{aligned}$$

$$B = \sum_{i=1}^{q} h_{N_i} \cdot (1 - P_{N_{i+1}}), \quad \text{where} \quad P_{N_{q+1}} = 0$$

■

The above result can also be verified intuitively. Indeed, after ending up with a wrong path and following the backtracking process described in theorem 1, $(1 - P_{N_{i+1}})$ expresses the probability that the first wrong decision occurred at a statistical node of the path before we reach the statistical node $S_{N_{i+1}}$. Therefore, in such a case we always back up to node S_{N_i} and thus in (21) we weight h_{N_i} (the height of S_{N_i}) with $(1 - P_{N_{i+1}})$. Indeed, assuming that the tree is complete and that after backtracking to a particular node S_{N_i} we traverse the tree by asking all the questions at the nodes we meet, we conclude that the number of these nodes is equal to h_{N_i}. Hence the summation in (21) represents the cost of the backtracking process according to the definition given before.

To achieve the minimum value of the backtracking cost B we must solve the following optimization problem

$$\min \left[\sum_{i=1}^{q} h_{N_i} \cdot (1 - P_{N_{i+1}}) \right] \quad (25)$$

In general, if there are m statistical nodes, we have 2^{m-1} different backtracking procedures which result in 2^{m-1} costs. That is so because the last node to which we back up is always node S_1 and 2^{m-1} is the number of all the possible combinations of the remaining $m-1$ nodes. Out of these 2^{m-1} procedures there is one (or possibly more) with the minimum cost given in (25). In other words there is a combination of q nodes, $1 \leq q \leq m$, which minimizes the backtracking cost given in (21). By exhaustively examining all the possible combinations, the minimum cost B for a particular path through the tree can be obtained. If we then repeat the same procedure for every path j of the set F_1 and substitute the results in (10) the maximum G for a specific partition of the interior nodes of the tree can be achieved. We must note that it is not necessary to calculate the costs of all the possible backtracking procedures. The backtracking process which results in the minimum cost has some special properties the application of which reduces the number of cost evaluations significantly. These properties are summarized in the following lemmas:

Lemma 1: The optimal backtracking procedure, that is, the one which achieves the minimum cost given in (25), does not contain a node S_l such that

$$\frac{h_l}{P_l} > h_r \quad (26)$$

where h_r is the height of node S_r, S_r being part of the backtracking procedure, and $h_r > h_l$.

Proof: Let us assume that the optimal backtracking procedure contains a node S_l which satisfies (26). The cost associated with that procedure can be written as follows (see Fig. 4)

$$B = B_1 + h_l \cdot (1 - P_k) + h_i \cdot (1 - P_l) + B_2$$

In the last expression, we assume that S_k is the node which precedes node S_l (if any) and S_t is the node which follows S_l in the backtracking process. B_1 stands for the cost up to the node S_k and B_2 gives the cost after node S_t. If node S_l is excluded from the backtracking procedure, i.e., instead of backtracking to S_l after node S_k, we back up directly to node S_t the cost can be expressed as

$$B' = B_1 + h_t \cdot (1 - P_k) + B_2$$

We shall prove that $B' < B$. In other words we shall show that the cost B of a backtracking procedure cannot be the minimum if the procedure includes a node S_l that satisfies (26). We have

$$\begin{aligned} B' - B &= h_t \cdot (1 - P_k) - h_l \cdot (1 - P_k) - h_t \cdot (1 - P_l) \\ &= [h_t \cdot P_l + P_k \cdot (h_l - h_t)] - h_l \end{aligned} \tag{27}$$

But from (26) we have

$$h_l > h_r \cdot P_l \geq h_t \cdot P_l > h_t \cdot P_l + P_k \cdot (h_l - h_t)$$

because $h_l - h_t < 0$. Hence, from the last inequality and (27), we conclude that if (26) is true then $B' < B$. ∎

The corollaries stated below follow directly from Lemma 1.

Corollary 1: The optimal backtracking procedure does not contain a node S_l such that $h_l/P_l > h_1$, where h_1 is the height of node S_1.

This is valid since S_1 is always part of any backtracking procedure and $h_1 > h_l$.

Corollary 2: The optimal backtracking procedure does not contain two consecutive nodes S_l and S_k such that $h_k < h_l/P_l$ and $h_k > h_l$.

Lemma 2: If $S_1, S_2, \cdots, S_m$ are the nodes of the path at which a statistical decision was taken and S, $1 \leq l \leq m$, is the node with the minimum value of h_i/P_i, that is

$$\forall i, \quad 1 \leq i \leq m, \quad i \neq l \quad \frac{h_l}{P_l} < \frac{h_i}{P_i} \tag{28}$$

then the optimal backtracking process cannot start at a node S_p where $h_p < h_l$.

Proof: Let us assume the opposite, i.e., the backtracking procedure includes nodes which are lower than S_l. In such a case, if S_k is the node of the optimal backtracking procedure closest to S_l with $h_k < h_l$, then the minimum cost can be written as follows (see Fig. 5)

$$B = B_1 + h_k \cdot (1 - P_r) + h_t \cdot (1 - P_k) + B_2$$

There are two possibilities. Either $S_l \equiv S_t$ i.e. S_l is part of the backtracking procedure, or $h_l < h_t$ and S_l is not part of the backtracking procedure but is located between two nodes which are. In the former case we have

$$\frac{h_l}{P_l} < \frac{h_k}{P_k} \Rightarrow h_l < \frac{h_k}{P_k}$$

which is not possible according to lemma 1. In the latter case, consider the cost of backtracking to node S_l instead of node S_k

$$B' = B_1 + h_l \cdot (1 - P_r) + h_t \cdot (1 - P_l) + B_2$$

We shall prove that $B' < B$. We have

$$\begin{aligned} B' - B &= h_l \cdot (1 - P_r) + h_t \cdot (1 - P_l) \\ &\quad - h_k \cdot (1 - P_r) - h_t \cdot (1 - P_k) \\ &= (h_l - h_k) \cdot (1 - P_r) + h_t \cdot (P_k - P_l) \end{aligned}$$

We will prove that the last expression is negative. Since we know that $h_l > h_k$ and $0 < 1 - P_r \leq 1$ it is sufficient to show that

$$(h_l - h_k) + h_t \cdot (P_k - P_l) < 0 \tag{29}$$

TABLE I
CHARACTERISTICS OF THE PATH OF FIG. 6

i	h_i	P_{S_i}	P_i	P_i/h_i
1	7	0.9720	1.0000	7.0000
2	6	0.7405	0.9552	6.2813
3	5	0.9464	0.5530	9.0413
4	3	0.8185	0.4916	6.1031
5	2	0.8429	0.2944	6.7928
6	1	0.7934	0.1548	6.4602

From (28) and lemma 1 we have

$$\frac{h_l}{P_l} < \frac{h_k}{P_k} < h_t \tag{30}$$

Therefore

$$\frac{h_l}{P_l} < \frac{h_k}{P_k} \Leftrightarrow h_t - \frac{h_k}{P_k} < h_t - \frac{h_l}{P_l} \Leftrightarrow$$

$$\frac{h_t \cdot P_k - h_k}{P_k} < \frac{h_t \cdot P_l - h_l}{P_l} \Rightarrow \frac{h_t \cdot P_k - h_k}{P_k} < \frac{h_t \cdot P_l - h_l}{P_k}$$

because the numerators of both fractions are positive according to (30) and furthermore $P_k < P_l$ (from 30 since $h_l > h_k$. Thus,

$$h_t \cdot P_k - h_k < h_t \cdot P_l - h_l$$

or

$$(h_l - h_k) + h_t \cdot (P_k - P_l) < 0$$

Therefore, the assumption that the backtracking procedure includes nodes S_p such that $h_p < h_l$ leads to a non-optimal cost B. ∎

Let us now summarize the procedure for obtaining the minimum backtracking cost for a single path of the tree. A particular path has n nodes and m statistical nodes. In order to select the q statistical nodes which result in the minimum backtracking cost we form the fractions h_i/P_i for all the statistical nodes. Then, we apply lemmas 1 and 2 and corollaries 1 and 2 in order to eliminate some of them. For each possible combination of the remaining statistical nodes we calculate the backtracking cost (21) and we end up with the set of nodes $S_{N_1}, S_{N_2}, \cdots, S_{N_q}$ that corresponds to the minimum cost. Therefore, every time that we follow that path and find out that the path is wrong we first backtrack to node S_{N_q} and traverse the tree by explicitly performing all the tests at the nodes we meet. If we end up again with a wrong final node we backtrack to node $S_{N_{q-1}}$ and repeat the same procedure until the correct final node is reached. This procedure is illustrated in the following example.

Example 1: Consider the path in Fig. 6 with $n = 7$. The characteristics of this path are summarized in Table I. In this table, P_{S_i} stands for the probability attached to the "child" of the statistical node S_i. Since, $m = 6$, $2^5 = 32$ different costs should be evaluated. But, according to lemma 1, node S_3 cannot be in the optimal path because $h_3/P_3 > h_1$. Notes S_5 and S_6 must also be excluded because from lemma 3 and the last column of Table I we conclude that the optimal procedure cannot start before node S_4. Finally, from lemma 2 we observe that the optimal backtracking procedure cannot contain consecutively nodes S_4 and S_2 because $h_2 < h_4/P_4$.

Thus, there are only two costs left that need to be computed. The cost of backtracking to node S_4 and then to node S_1 and that of backtracking to the node S_2 and then to the node S_1. In the former case we have

$$B_1 = h_4 + h_1 \cdot (1 - P_4) = 6.5588$$

In the latter case we have

$$B_2 = h_2 + h_1 \cdot (1 - P_2) = 6.3136$$

We conclude that 6.3136 is the minimum cost. Therefore, every time that we follow this path and find out that it is wrong we first backtrack to node S_2. If we end up again with wrong final node we back up to node S_1.

IV. Partitioning of the Interior Nodes

It was mentioned earlier that in order for the maximum value of the gain G, given in (10), to be attained, the appropriate partition of the interior nodes of the tree must be found. This partition can be called the optimal. Under the hypothesis that the optimal separation of the nodes is given, the maximum gain G can be obtained by minimizing the backtracking cost B_j for all j in F_1. In general, if there exist s interior nodes, 2^s possible partitions of them into statistical and non-statistical can be considered. For each of them the maximum gain must be calculated according to (10) and (25) and the results must be compared to each other in order for the optimal gain to be obtained. More specifically, in a tree of depth n there exist $2^n - 1$ interior nodes. Clearly, the corresponding number of possible partitions is large. Even for a tree with $n = 5$ that number is too high (2^{31}) resulting in an excessive amount of computation during the maximization procedure. For these reasons an exhaustive evaluation of the gain for each possible partition is to be avoided. However, in the following, two partitioning techniques are proposed. In both of them the computational complexity is kept linear to the number of the interior nodes of the tree.

A. The Threshold Method

As it was stated in section II, each interior node (test) k is associated with two probabilities that add up to 1 and represent the probability of occurrence of the two possible test outcomes. Obviously, the maximum of these two probabilities, say P^k_{max}, is greater than or equal to 0.5. We define a *threshold* t to be a number in the interval [0.5, 1]. When node k is reached, P^k_{max} is compared to the threshold t. If it is greater than or equal to t we follow the branch with the maximum probability without performing any test at the node. Otherwise, the branch which will be followed is determined according to the test outcome.

Any threshold t, $0.5 \leq t \leq 1$, separates the interior nodes of the tree into statistical and non-statistical as follows: all the interior nodes whose maximum probability is greater than or equal to t are considered statistical and the remaining become non-statistical. Therefore, the gain G is expressed as a function of the probability threshold t in the interval [0.5, 1].

In a complete binary balanced tree of depth n there are $s = 2^n - 1$ interior nodes and therefore there exist s probabilities greater than or equal to 0.5. For simplicity and without loss of generality we can assume that these probabilities are distinct and different from 0.5 and 1. Under this assumption, these probabilities divide the interval [0.5, 1] into 2^n subintervals of the form $I_r = (p_r, p_{r+1}]$; $r = 1, 2, \cdots, 2^n$, $p_{2^n} = 1$, plus the interval $I_0 = [0.5, p_1)$ (Fig. 7). Among these subintervals there is one (and possibly more), denote it by I_{opt}, such that if $t \in I_{opt}$ the gain G is maximized. The value of G is constant in each of the subintervals. This is so because, for each value of the threshold in a particular interval, the resulting partition of the nodes is the same. In order to get the maximum value of G, equation (10) is evaluated in each of the subintervals. We must note that the maximum of G obtained is not the absolute maximum discussed before. It simply is the optimal gain achieved by applying the threshold method. The gain as a function of the threshold for a tree of depth 4 is shown in Fig. 8. For the production of the probabilities of the tree a uniform random generator has been used. In this example, the maximum value of the gain obtained is 1.4015, for $t \in (0.575, 0.712]$.

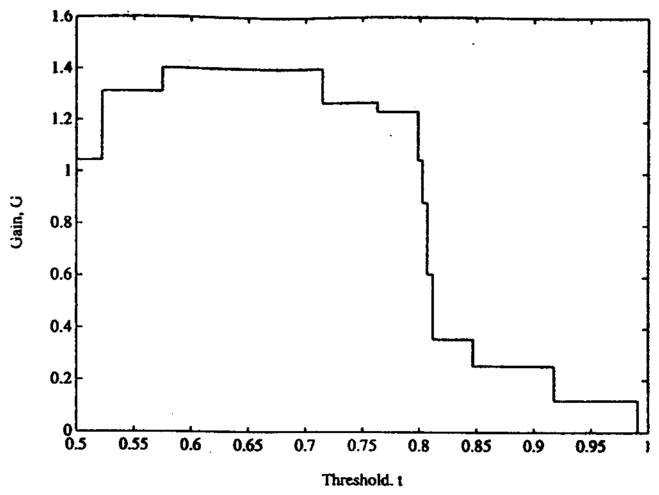

Fig. 8. The Gain versus the Threshold.

B. An Alternative Method

Irrespective of their heights, the threshold method handles nodes with the same maximum probability in a similar manner. But, in general, a node which is located closer to the root of the tree has a higher backtracking cost. Therefore, in addition to the maximum probability P^k_{max}, the height of an arbitrary node k must be also taken into account when we search for a partition of the interior nodes of the tree.

We observe from (21) that the backtracking cost associated with a single path is increased when statistical decisions take place at high levels of the tree. That happens because when we end with a wrong path we back up with some probability to high levels of the tree and then we have to traverse the tree by performing all the tests at all the nodes we meet. Therefore, if we reduce the number of probabilistic decisions in such nodes and at the same time we increase the number of probabilistic decisions in nodes with relatively small height, we may obtain a higher value for the gain G. Thus, a different partition of the interior nodes of the tree results.

In the new method we make a level-order visit to all the interior nodes of the tree starting from the last level (see Fig. 9). At each node we calculate the total gain of the paths which contain that node by considering it as a statistical one. The gain so obtained, is compared with the sum of the gains of the two sub-trees originating at the node, and if it is greater than the sum, the node becomes statistical. Otherwise it remains non-statistical and we proceed to the next node of the same level of the tree (if any) or to the first node of the next level. We note that the gains of the two sub-trees have already been calculated in the previous level of the tree.

As it is clear the new technique is based on a greedy algorithm [10]. At each step of the algorithm we make the "locally" optimal choice. We must note that not every greedy approach succeeds in producing the best result overall. However, if the problem is such that the only way to get an optimal solution is to use an exhaustive search technique, then a greedy algorithm or other heuristic for getting a good but not necessarily optimal solution may be our only real choice.

C. Comparison of the two Methods

Consider the tree of depth $n = 4$ drawn in Fig. 10. The probabilities that are greater than 0.5 appear attached to the corresponding

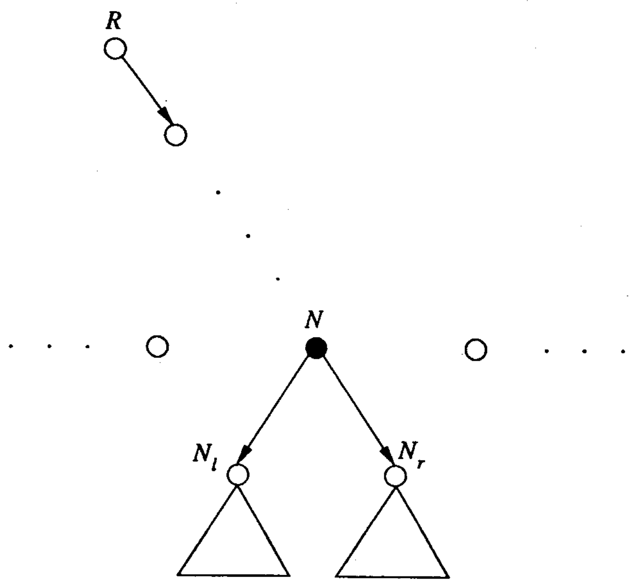

Fig. 9. Illustration of the Alternative Method.

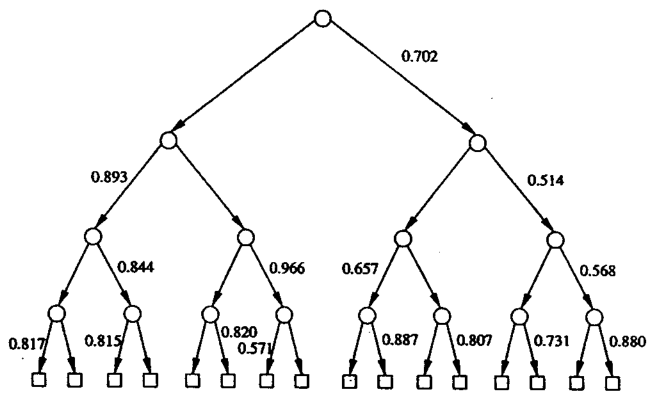

Fig. 10. Example Tree.

branches. After applying the threshold and the alternative techniques the state of the tree appear in the Figs. 11 and 12, respectively. We can observe the different partitions of the interior nodes resulting from the two methods. In these figures, we use solid circles to indicate statistical nodes and solid squares to indicate nonreachable final nodes. Observe that for both methods children of statistical nodes corresponding to the minimum probability path are unreachable and have been marked as non-statistical by default. When the threshold method is used the maximum value of the gain is $G = 1.157$. On the other hand, the gain is equal to 1.327 when the alternative method is applied.

In Table II the two techniques discussed so far are compared for trees of different depths. In that table $\overline{G}_t$, $\overline{G}_a$ stand for the average gain of the threshold and the alternative technique respectively. The third and fifth columns contain the corresponding standard deviations. For each value of n, we calculated the gain for 5000 different trees, that is trees with different uniform distributions of probabilities at the interior nodes. Then the entries of Table II are computed ([11]). From that table we observe that the alternative method performs better than the threshold method. In general, the performance of both methods depends exclusively on the probabilities and their locations in the tree structure. Therefore, the value of the gain may vary significantly for trees of the same depth, which explains why large deviations appear in the corresponding columns of Table II.

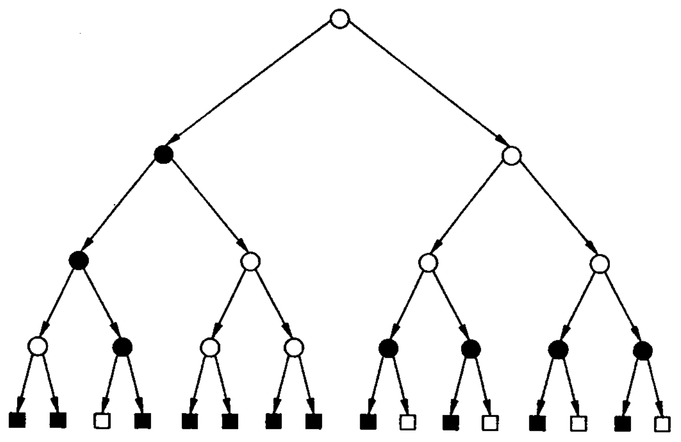

Fig. 11. The State of the Tree after applying the Threshold Method. Solid circles indicate statistical nodes, open squares indicate reachable final nodes. The threshold interval is (.702, .731].

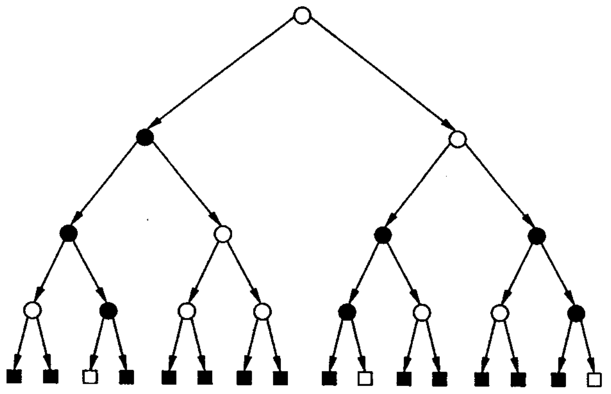

Fig. 12. The State of the Tree after applying the Alternative Method. Solid circles indicate statistical nodes, open squares indicate reachable final nodes.

We must note that the alternative technique which gives the best results so far was compared to a method that exhaustively evaluates the gain for all the possible partitions of the interior nodes into statistical and non-statistical and therefore achieves the absolutely maximum value of the gain. That took place for several trees of depth 4 and one tree of depth 5. Exhaustive evaluation of the gain is computationally expensive (the depth 5 tree required several days of run time on an HP 715/50 system) and thus larger trees or more examples of trees of depth 5 were not attempted. In all cases tried, the results obtained by the alternative method were identical to those of the exhaustive evaluation of the gain.

V. Summary-Conclusions

In this paper we presented a probabilistic model for reducing the number of tests that are required in a specific decision procedure. We assumed that a problem is structured as a complete binary balanced decision tree and we attempted to select the nodes of the tree where a probabilistic decision is taken. A gain function was constructed and dominance criteria were derived which reduce the computational complexity of the gain function. Two heuristic methods were proposed for the selection of the nodes where a decision is taken probabilistically and they were compared to each other in terms of the value of the gain achieved. Both techniques are linear to the number of the interior nodes of the tree while an exhaustive evaluation of the gain would result in exponential complexity.

It is envisioned that the probabilities attached to the decision nodes will be acquired during a learning process where the decision making

TABLE II
PERFORMANCE OF THE TWO TECHNIQUES FOR DIFFERENT DEPTHS OF THE TREE

n	$\bar{G}_t$	s_t	$\bar{G}_a$	s_a
4	1.586	0.521	1.658	0.472
5	1.660	0.556	1.806	0.483
6	1.691	0.562	1.934	0.467
7	1.700	0.545	2.034	0.448
8	1.696	0.520	2.123	0.427
9	1.688	0.492	2.204	0.408
10	1.672	0.445	2.269	0.382

of a user(s) will be observed for a period of time, and the node probabilities will be adjusted accordingly.

This process resembles the maturing of a novice decisionmaker to an expert who, based on past experience, is capable of reaching directly to a conclusion after some early observations have narrowed the number of alternatives to a few highly probable ones.

ACKNOWLEDGMENT

The authors would like to thank the anonymous referees for their helpful comments and suggestion.

REFERENCES

[1] R. E. Neapolitan, *Probabilistic Reasoning in Expert Systems.* New York: John Wiley, 1990, pp. 371–380.
[2] M. Hamburg, *Statistical Analysis of Decision Making.* Harcourt, Brace & World Inc., 1970, pp. 726–737.
[3] J. Pearl, *Probabilistic Reasoning in Intelligent Systems.* San Mateo, CA: Morgan Kaufmann, 1988.
[4] J. R. Quinlan, "Induction of decision trees", *Machine Learning*, vol. 1, no. 1, pp. 81–106, 1986.
[5] ——, "Decision trees and decisionmaking," *IEEE Trans. Syst., Man and Cybern*, vol. 20, no. 2, pp. 339–346, Mar./Apr. 1990.
[6] C. Carter and J. Catlett, "Assessing credit card applications using machine learning," *IEEE Expert*, vol. 2, no. 3, pp. 71–79, Fall 1987.
[7] K. B. Irani, J. Cheng, U. M. Fayyad, and Z. Qian, "Applying machine learning to semiconductor manufacturing," *IEEE Expert*, vol. 8, no. 1, pp. 41–47, Feb. 1993.
[8] J. R. Quinlan, "Decision trees as probabilistic classifiers," *Proc. Fourth Int. Workshop on Machine Learning*, P. Langley, Ed. Los Altos, CA: Morgan Kaufmann, 1987, pp. 31–37.
[9] A. Rontogiannis, "A probabilistic approach for reducing the search cost in binary decision trees", *Master's Thesis*, University of Victoria, 1993.
[10] A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *Data Structures and Algorithms.* Addison Wesley, 1983, pp. 306–310.
[11] G. W. Snedecor and W. G. Cochran, *Statistical Methods*, 7th ed. The Iowa State University Press, 1970, pp. 26–38.

A New Safe-Point Thinning Algorithm Based on the Mid-Crack Code Tracing

Frank Y. Shih and Wai-Tak Wong

Abstract—**A new thinning algorithm for binary images, based on the safe-point testing and mid-crack code tracing, is presented in this paper. Thinning is treated as the deletion of nonsafe border pixels from the contour to the center of the object layer-by-layer. The deletion is determined by masking a 3 × 3 weighted template and by the use of look-up tables. The resulting skeleton does not require cleaning or pruning. The obtained skeleton possesses single-pixel thickness and preserves the object's connectivity. The algorithm is very simple and efficient since only boundary pixels are processed at each iteration and look-up tables are used.**

I. INTRODUCTION

Skeletonization or thinning is a very important preprocessing step in pattern analysis such as industrial parts inspection [1], fingerprint recognition [2], optical character recognition [3], and biomedical diagnosis [4]. One advantage of skeletonization is the reduction of memory space required for storing the essential structural information presented in a pattern. Moreover, it simplifies the data structure required in pattern analysis. Most of the skeletonization algorithms require iterative passes through the whole image, or at least through each pixel of the object considered. At each pass, a relatively complicated analysis over each pixel's neighborhood must be performed, which makes the algorithms time-consuming.

A practical problem with the definition of the skeleton is that circular neighborhoods cannot be represented exactly on a discrete grid. A reasonable compromise reached is that the generated skeleton must be essential to preserve the object's topology and to represent informatively the pattern's shape. Many skeletonization algorithms are available in the literature. Different algorithms produce slightly different skeletons. Rosenfeld et. al. [5], [6] classified skeletonization algorithms as being parallel or sequential.

A parallel skeletonization algorithm is called "*safe-point thinning algorithm*" (SPTA) [7] where the safe-point testing is conducted by examining a set of Boolean expressions on eight neighbors of each edge-point to determine whether it is a safe or nonsafe point. The nonsafe points are then removed by a two-scan algorithm. A decision tree is constructed to minimize the number of neighbors to be examined. Since the algorithm is performed by scanning all object pixels at each iteration until no more nonsafe point exists, it is inefficient compared to our algorithm.

Some thinning algorithms are based on the contour generation method [8], [9], [13]. The essential idea is first to convert the input image into chain codes for each closed contour, and then to trace around the contour. If a boundary point is removed, the algorithm will generate a few new chain codes to replace the old one. The new chain codes are generated from two directional vectors, inward and outward, of each boundary point. All cases of two directional vectors are tabulated as a look-up table. The contour tracing will process each contour layer-by-layer iteratively until no further deletion occurs. The

Manuscript received April 24, 1992; revised December 26, 1993. This work was supported by the National Science Foundation under Grant IRI-9109138.
The authors are with the Department of Computer and Information Science, New Jersey Institute of Technology, Newark, NJ 07102 USA.
IEEE Log Number 9405710.