Bottom-Up Construction of Minimum-Cost AND/OR Trees for Sequential Fault Diagnosis

O. Erhun Kundakcioglu and Tonguç Ünlüyurt

***Abstract*—The problem of generating the sequence of tests required to reach a diagnostic conclusion with minimum average cost, which is also known as a test-sequencing problem, is considered. The traditional test-sequencing problem is generalized here to include asymmetrical tests. In general, the next test to execute depends on the results of previous tests. Hence, the test-sequencing problem can naturally be formulated as an optimal binary AND/OR decision tree construction problem, whose solution is known to be $\mathcal{NP}$-hard. Our approach is based on integrating concepts from one-step look-ahead heuristic algorithms and basic ideas of Huffman coding to construct an AND/OR decision tree bottom-up as opposed to heuristics proposed in the literature that construct the AND/OR trees top-down. The performance of the algorithm is demonstrated on numerous test cases, with various properties.**

***Index Terms*—AND/OR trees, asymmetrical tests, Huffman coding, sequential fault diagnosis.**

I. Introduction

Due to the increasing complexity of systems, maintenance has become a very critical activity. The purpose of maintenance is to keep the system running, and if the system fails, to diagnose and repair detected failures as quickly as possible with minimum cost. In this paper, a specific maintenance framework is considered. A system consisting of a number of components can be in a number of failure states, depending on the functionality of individual components (or there is no failure). Since for each failure state, a different action will be taken for repair it is important to find out the correct failure state that the system is in. Specifically, the goal of a diagnostic procedure is to identify the actual system state. The system is in a certain unknown state before the diagnostic procedure. The diagnostic procedure identifies the actual state of the system by gathering information using available tests. Any measurement, observation, signal can be considered as an available test. Since each test has an associated cost and one needs not apply all the tests to figure out the failure state of the system, one can avoid unnecessary costs by carefully choosing the tests and the order to execute these tests. In general, the next test to execute depends on the results obtained from previously executed tests.

Manuscript received December 20, 2004; revised February 3, 2006. This paper was recommended by Associate Editor P. K. Willett.

O. E. Kundakcioglu is with the Industrial and Systems Engineering Department, University of Florida, Gainesville, FL 32611 USA.

T. Ünlüyurt is with Sabancı University, 34956 Istanbul, Turkey.

Digital Object Identifier 10.1109/TSMCA.2007.893459

Our goal in this paper is to develop an algorithm that uses the (*a priori*) failure probabilities and test costs to construct efficient diagnostic procedures, to minimize the expected cost of diagnosis. Minimizing expected cost is appropriate when the same system (or identical systems) is tested many times over time. In such a case, one can generate the diagnostic procedure off line and then use the same strategy over and over. Hence, it is feasible to spend sufficient time to construct a good strategy especially if the reductions in expected cost are very important. Finding the minimum expected cost diagnostic procedure, which is formally defined as a test-sequencing problem in literature, is known to be an $\mathcal{NP}$-hard problem [1].

The test-sequencing problem belongs to the general class of binary identification problems that arise in a wide area of applications. Other than maintenance operations [2]–[5], such problems arise in botanical and zoological field of work, plant pathology, medical diagnosis, decision table programming, computerized banking, pattern recognition, nuclear power plant control [1], [6], [7], discriminant analysis of test data, reliability analysis of coherent systems, research and development planning (e.g., in allocation of limited funds among high-risk projects), communication networks, speech/voice recognition (e.g., in classification of pattern vectors), distributed computing, and in the design of interactive expert systems [8].

There is a certain degree of analogy between a noiseless coding problem and a test-sequencing problem. The traditional test-sequencing heuristics suggest classification/separation whereas Huffman's algorithm [9], which is known to give optimal solutions for noiseless coding problem, is based on tieing/binding sets of states. In other words, the traditional test-sequencing heuristics build the solution top-down and Huffman's algorithm builds the optimal code bottom-up. In this paper, we present how to construct the AND/OR decision tree bottom-up, in contrast to the previous approaches for the test-sequencing problem. Besides, we also present computational results, that demonstrate that the performance of the bottom-up strategies is better than some other heuristic algorithms proposed in the literature.

This paper is organized as follows. In Section II, we describe and define the test-sequencing problem in detail. Previous studies on test-sequencing problem are presented in Section III. In Section IV, bottom-up AND/OR decision tree construction technique is described in detail. Results for randomly generated numerous instances are presented in Section V. Finally, Section VI provides a summary and future extensions.

TABLE I
EXAMPLE

	Tests		
s	t_1	t_2	t_3
s_0	0	0	0
s_1	0	1	0
s_2	1	1	0
s_3	1	1	1

II. Test-Sequencing Problem

In this paper, the test-sequencing problem is considered in the following context. We are given [1], [2] the following:

1) a set of $m+1$ system states $S = \{s_0, s_1, s_2, \ldots, s_m\}$, where s_0 denotes the fault-free state of the system and s_i denotes one of the m potential faulty states of the system for $1 \leq i \leq m$;
2) the prior conditional probability vector of the system states $P = [p(s_0), \ldots, p(s_m)]^T$, where $p(s_0)$ is the conditional probability that no fault exists in the system and $p(s_i)$ denotes the probability that system is in state i[1] for $1 \leq i \leq m$;
3) a set of n reliable tests $T = \{t_1, t_2, \ldots, t_n\}$ and a cost vector $C = [c_1, c_2, \ldots, c_n]^T$, where c_j denotes the cost of applying test t_j, measured in terms of time, pain, manpower requirements, other economic factors, etc.; and
4) an $m \times n$ diagnostic dictionary matrix $D = [d_{ij}]$, where d_{ij} is 1 if test t_j detects a failure state s_i, and 0 otherwise.

The goal of the test-sequencing problem is to generate a diagnostic procedure such that the criterion function given by

$$J = P^T AC = \sum_{i=0}^{m} \sum_{j=1}^{n} \alpha_{ij} p(s_i) c_j \tag{1}$$

(i.e., the average cost of the decision tree) is minimized [1]. In (1), $A = (\alpha_{ij})$ is an $(m+1)$ by n matrix such that $\alpha_{ij} = 1$ if test t_j is used in the path leading to the identification of system state s_i and is 0 otherwise. An optimal diagnostic procedure is one which has the minimum cost over all diagnostic procedures using tests from T to determine actual system state. The construction of an optimal diagnostic procedure is an $\mathcal{NP}$-hard problem [1], [7].

A feasible solution for the test-sequencing problem can be described as a binary decision tree where nodes correspond to the tests, arcs correspond to the outcome of the tests, and leaves correspond to the actual system states. A feasible binary decision tree for the example in Table I, is shown in Fig. 1. For instance, tests t_1 and t_2 are used in the path leading to the identification of state s_0. In other words, if the system is in state s_0, total cost to identify the system state is $(c_1 + c_2)$. Summing up over all states and tests, average cost of the decision tree is

$$J = p_0[c_1 + c_2] + p_1[c_1 + c_2] + p_2[c_1 + c_3] + p_3[c_1 + c_3].$$

The problem above can be considered as a Markov decision problem (MDP) [1], wherein the Markov state x denotes the suspect set of system states (also termed the ambiguity set),

[1]The techniques to compute these probabilities are explained in detail in [2].

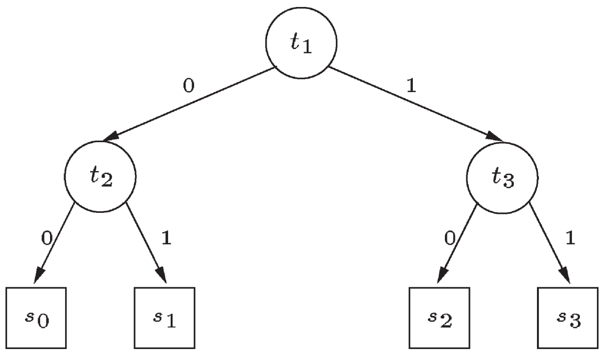

Fig. 1. Decision tree for the example in Table I.

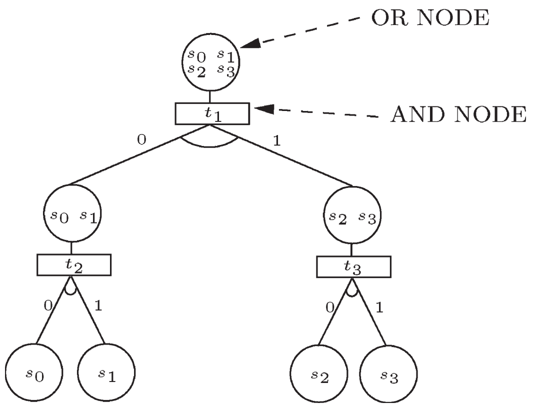

Fig. 2. AND/OR binary decision tree for the example in Table I.

and the decision corresponds to the test to perform in state x. The solution to the MDP is a deterministic AND/OR binary decision tree, with OR nodes labeled by ambiguity status x, AND nodes denoting tests (decisions) at OR nodes, and the weighted average length of the tree representing the expected test cost, J. The corresponding AND/OR binary decision tree for the case represented in Fig. 1 is shown in Fig. 2.

The tests described above have binary outcomes, i.e., a test fails (outcome $= 1$) if it has detected a failure and passes otherwise (outcome $= 0$). It is assumed that the results obtained from the tests are always correct. In case of binary tests, two sets of system states are defined: one corresponding to the fail test outcome (set A) and the other to the pass test outcome (set B). It is obvious that every system state has to be an element of at least one set (i.e., $A \cup B = S$, since the test always has a result). We distinguish between two types of tests [3], as follows:

1) **symmetrical** test where $A \cap B = \emptyset$;
2) **asymmetrical** test which is a more general test form including the cases when $A \cap B \neq \emptyset$.

The conventional test-sequencing-problem formulation assumes that tests are symmetrical. For a symmetrical test, the outcome is determined by the state of the system: $s_i \in A \Longleftrightarrow$ test fails. In other words, the ith element of the test vector d_{ij} is 1, iff test t_j detects a failure state s_i.

In the case of asymmetrical test, there is at least one system state that remains on the candidate list regardless of the test outcome. In this case, in the diagnostic dictionary matrix contains fractional numbers between 0 and 1. Let us say d_{ij} is fractional. This means that the system is in state i regardless of the result of test j. In particular, the outcome of test j is 1 with probability d_{ij}, if the state is in fact in state i. This is equivalent to having two rows in the diagnostic matrix for a symmetrical test-sequencing problem that leads to the same faulty state.

Systems with symmetrical tests are one of the most widely studied cases. However, even if we impose the additional constraint that tests have uniform costs, the problem is still $\mathcal{NP}$-hard [10]. In this case, the problem is simply minimizing the expected number of tests.

III. Previous Work

Exact and approximate solution approaches have been proposed in the literature for different versions of the test-sequencing problem. A dynamic programming (DP) approach has been proposed for the exact solution of the problem. Due to the structure of the problem, from a formulation point of view, DP is a convenient method. The DP algorithm builds the optimal decision tree from the leaves up by identifying successively larger subtrees until the entire tree from the initial node of complete ambiguity is generated. The DP technique [7] has storage and computational requirements $O(3^n)$, and, hence, is impractical for large n, where n is the number of tests. Therefore, approximation techniques for constructing nearly optimal decision trees are essential. Most of the traditional approximation techniques employ "greedy" heuristics; that is they perform a local optimization at each step.

One of the earliest greedy heuristics is the information heuristic developed by Johnson [11]. The information heuristic is a one-step look-ahead procedure with computational complexity $O(mn)$.

Another similar heuristice[2] is the "separation heuristic" [12], where at each node of ambiguity a test t_k is selected that maximizes the distinguishability criterion.

Varshney *et al.* [13] develops an algorithm that is based on minimizing the upper bound at each step during the construction. A multistep look-ahead procedure similar to the information heuristic is used to derive these upper bounds.

The approach of Pattipati and Alexandridis [1] is based on integrating concepts from information theory and heuristic AND/OR graph search methods [14].

A different approach is proposed by Fraughnaugh *et al.* [8]. A number of different algorithms using various heuristic techniques including hillclimbing, random search, and taboo search are developed. Various generic decision rules are used to determine which test to perform next in any state.

Another interesting paper, where the same problem comes up, is by Raghavan *et al.* [2]. In this paper, various graph search and information heuristic-based algorithms are developed to solve the test-sequencing problem. The major contribution in this paper is that, a generalized test-sequencing problem that incorporate various practical features such as precedence constraints and rectification is considered.

In [10], Raghavan *et al.* consider not only the test-sequencing problem (i.e., how to determine a test sequence that minimizes expected testing cost), but three more problems as well; how to determine a test sequence that does not depend on the failure probability distribution, how to determine a test sequence that minimizes average ambiguity group size without using more than a number of tests, and how to determine a test sequence that minimizes the storage cost of tests[3] in the diagnostic strategy.

In [15], Shakeri *et al.* develop algorithms for multiple fault diagnosis. The single-fault strategy of their previous work [1], [2] is extended to diagnose multiple faults by successively isolating the potential single-fault candidates, then double-fault candidates, and so on.

Tu and Pattipati combine rollout algorithm (RH) with test-sequencing heuristics in [16]. RH based on a heuristic test-sequencing algorithm H proceeds as follows.

The cost of constructing a test tree at an intermediate node (set of ambiguity) i is denoted by $H(i)$. Let $N(i)$ denote the set of immediate successor nodes of an OR node i, that is $N(i) = \{j | (i, j) \text{ is an arc}\}$ which contains all the tests available at node i. The RH starts with the root node S and at any intermediate OR node i, RH adds to the test sequence a test j_{k+1} such that

$$j_{k+1} = \arg \min_{j \in N(i)} H_j(i) \tag{2}$$

where $H_j(i)$, $j \in N(i)$ denotes the expected test cost starting at OR node i and applying test j as the first test at that node. The algorithm terminates when all successors are destination nodes.

A more generalized, hence, harder to solve systems, are the ones that include asymmetrical tests. Although most of the algorithms used for systems with symmetrical tests can be applied to systems with asymmetrical tests, the performance of the algorithm will obviously decrease since it cannot make use of the special structure of the problem.

In [3], Biasizzo *et al.* prove that the same heuristics that have been employed in the traditional solution of the problem such as AO* algorithm with heuristics based on Huffman coding, can also be employed for the generalized case with asymmetrical tests.

In [4], Žužek *et al.* present the sequential diagnosis tool that enables the user to generate solutions of the generalized test-sequencing problem. The purpose of this paper is to report this first sequential diagnosis software, which provides solutions to the generalized case including asymmetrical tests.

Optimal test algorithms with computational complexity $O(m \log m)$ can be designed for two extreme cases of the test-sequencing problem [1].

The first case occurs when the binary test matrix is diagonal with singleton tests, that is, test t_j detects faults in system state j $(1 \le j \le m)$. In this case, the total expected testing cost J for a given ordering of tests $([1], [2], \ldots, [m])$ is given by

$$J = \sum_{j=1}^{m} c_{[j]} \left[p(s_0) + \sum_{i=j}^{m} p\left(s_{[i]}\right) \right]. \tag{3}$$

[2]The information heuristic provides the same decision tree as the distinguishability criterion when test costs are equal [1].

[3]Minimizing the storage cost of the tests is essentially minimizing the number of tests in the tree.

The optimal test sequence is the priority rule

$$\frac{p\left(s_{[1]}\right)}{c_{[1]}} \geq \frac{p\left(s_{[2]}\right)}{c_{[2]}} \geq \cdots \geq \frac{p\left(s_{[m]}\right)}{c_{[m]}}. \tag{4}$$

On the other extreme, if all 2^m tests are available and the test costs are equal, the test-sequencing problem is identical to Huffman coding problem; that is, the problem of generating the minimum redundancy, prefix-free binary code of a set of binary messages for transmission over a noiseless channel [9]. In [1], [2], and [13], the analogy between the test-sequencing problem and Huffman coding problem is discussed in detail.

A special case of the general test-sequencing problem where there are only two classes but there is a high degree of asymmetry has been considered in various studies. In this particular case, one is interested in finding the correct value of a Boolean function with the minimum expected costs by learning the values of individual arguments via costly tests. Polynomial time exact algorithms have been proposed for certain cases and effective heuristic algorithms are developed (see [17] for a general review).

IV. Bottom-Up Construction of AND/OR Trees

A. Constructing Feasible Decision Trees

In this section, we shall provide a strategy for a test-sequencing problem that constructs the decision tree bottom-up. This strategy is based on binding two states or sets of ambiguity, unlike the traditional methods that are based on dividing a set of ambiguity at each iteration. Constructing the decision tree upwards (i.e., beginning from actual states, ending with the set of all states S) is similar to the idea of Huffman coding. Unfortunately, Huffman coding algorithm cannot be directly adapted for test-sequencing problem. In the Huffman coding problem, we have the classes (different faulty states) as in the test-sequencing problem, but it essentially assumes all tests are available. Consequently, one has to find ways to incorporate the constraint that there are only some tests available to adapt Huffman coding algorithm for the test-sequencing problem. The advantages and disadvantages of bottom-up construction of AND/OR trees are as follows.

Generally, there is one good answer at each iteration of traditional methods but there are usually more than one in a binding strategy-based algorithm. Suppose that the binary decision tree shown in Fig. 3 is the only optimal tree for a specific test-sequencing problem. If a greedy heuristic based on traditional classification approach is employed, the algorithm must produce the result "perform test t^*" at the end of first iteration to come up with the optimal tree. On the other hand, if a heuristic based on binding strategy that constructs the tree upward is employed, the algorithm may produce either "bind states s_{i^*} and s_{j^*}" or "bind states $s_{i^{**}}$ and $s_{j^{**}}$" both of which have the possibility of constructing the optimal tree. The only situation that we cannot make use of this useful property is the case when the only optimal binary decision tree is as shown in Fig. 4.

The major disadvantage of using binding strategy is the additional computational work at each iteration. In traditional

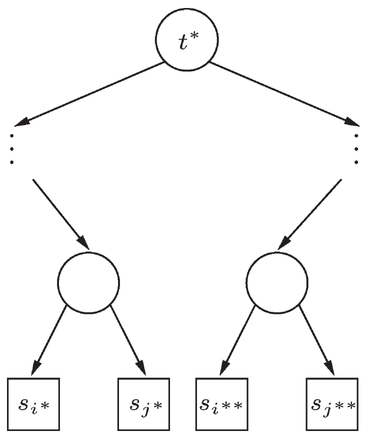

Fig. 3. Traditional approach versus binding strategy.

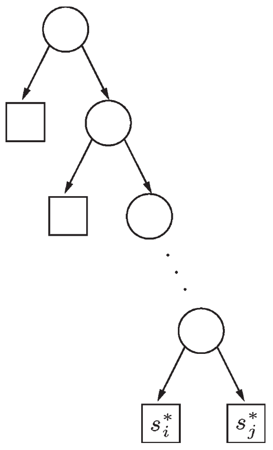

Fig. 4. Exceptional case. One path leading the optimal solution using binding strategy.

approaches, at a node of ambiguity, any test is guaranteed to lead to a decision tree. However, when an algorithm based on binding strategy is employed, binding some states may not lead to a situation where it is possible to reach the set of ambiguity formed by the binding using available tests. At any stage, if it is possible to classify every state or set of ambiguity using tests available, the problem is referred to as feasible.

It is crucial to note that when an algorithm based on binding strategy is employed, as long as the problem is feasible at any time, there always exists at least one pair of states that is allowed to be bound together using tests available. Note also that, when defining the test-sequencing problem, the standard assumption that the problem should be feasible, is imposed. These statements and the following lemma form a base for the proposed algorithm in this paper.

Lemma 1: If the problem is feasible at any stage, then there exists at least one pair of states, and at least one available test for this binding, that leads to a new feasible problem.

Proof: If the problem is feasible at any stage, then there exists at least one feasible binary decision tree that can perform the classification of the problem.

If there exists such a feasible binary decision tree, intuitively every OR node in this tree is feasible. Also, if there exists

TABLE II
DISADVANTAGE OF USING BINDING STRATEGY

	Tests		
	t_1	t_2	t_3
	Test Costs		
s	1	1	1
s_1	1	1	0
s_2	0	0	1
s_3	0	1	1

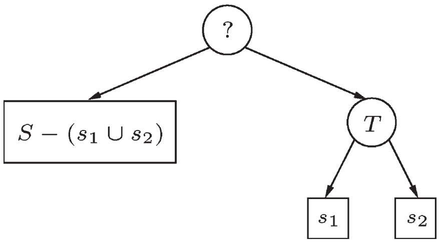

Fig. 5. Unallowed binding for the case in Table II.

a feasible binary decision tree, then there exist at least one binding and a test that performs that binding at the bottom of the tree. Since every OR node in this tree is feasible, this binding is guaranteed to give a new feasible problem. ■

Suppose the diagnostic dictionary matrix in Table II is the case. It is easy to see that the traditional approach may come up with any test at any iteration, and that will lead to a solution. Therefore, no test[4] is forbidden at any iteration. On the other hand, if an algorithm based on binding strategy is employed, that algorithm should not allow binding some states in general. For instance, at the first iteration binding s_1 and s_2 should not be allowed. If binding s_1 and s_2 is allowed and a new set is defined as $s_4 = s_1 \cup s_2$, the algorithm necessitates that all other states (in this case s_3 only) should be separated from this new set s_4 using any available test as in Fig. 5 (i.e., the problem should be feasible). s_1 and s_2 may be classified using t_1, t_2, or t_3 all of which have different values for s_1 and s_2. This implies that none of these tests can be used in subsequent iterations that considers binding s_4 and any other set (i.e., test nodes above node T). At the same time, it is necessary to bind s_4 and s_3 but no test is available above node T, which means that binding s_1 and s_2 leads to an infeasible result.

To overcome this difficulty, when two states (say s_{i^*} and s_{j^*}) are considered to be bound together, we propose to update the diagnostic dictionary matrix temporarily as follows.

1) Introduce a new state, say s_k, where $p(s_k) = p(s_{i^*}) + p(s_{j^*})$.
2) If $d_{i^*l} \neq d_{j^*l}$, set $d_{kl} = 2\ \forall l$, otherwise set $d_{kl} = d_{i^*l}$. Note that a value of 2 in the diagnostic dictionary matrix implies that the test is not available in the subsequent steps (i.e., upper parts of the tree).
3) Delete rows corresponding to s_{i^*} and s_{j^*}.

[4]Intuitively, a useless test that does not perform any separation (i.e., the values in diagnostic matrix are the same for all states in current set or shows asymmetrical behavior) is not preferred, but not forbidden because it leads to a feasible solution.

TABLE III
EXAMPLE: DIAGNOSTIC DICTIONARY MATRIX, FAULT PROBABILITIES, AND TEST COSTS

System state	Tests t_1	t_2	t_3	t_4	t_5	System state probabilities
	Test costs c_j					
s_i	1	1	1	1	1	$p(s_i)$
s_0	0	0	0	0	0	0.32
s_1	0	0	1	1	0	0.30
s_2	0.8	0	0	1	1	0.16
s_3	1	0	0.5	0	0	0.12
s_4	1	1	1	1	0	0.10

TABLE IV
APPROPRIATE FORM TO APPLY THE BINDING STRATEGY

System state	Tests t_1	t_2	t_3	t_4	t_5	System state probabilities
	Test costs c_j					
s_i	1	1	1	1	1	$p(s_i)$
s_0	0	0	0	0	0	0.32
s_1	0	0	1	1	0	0.30
s_2	0	0	0	1	1	$0.16 \cdot 0.2 = 0.032$
$s_{2'}$	1	0	0	1	1	$0.16 \cdot 0.8 = 0.128$
s_3	1	0	0	0	0	$0.12 \cdot 0.5 = 0.06$
$s_{3'}$	1	0	1	0	0	$0.12 \cdot 0.5 = 0.06$
s_4	1	1	1	1	0	0.10

Binding s_{i^*} and s_{j^*} is allowed if the updated diagnostic dictionary matrix is that of a feasible problem. The feasibility of the updated problem is checked by applying the tests sequentially to S until all states are classified. The binding is allowed if the updated states can be classified using applicable tests at each node of ambiguity. At a node of ambiguity, if a test has a value of 2 for at least one of the states, then the test is not applicable at that node. If no test can be applied to a set of ambiguity, the updated problem is infeasible and the binding is not allowed.

After above procedure is completed, diagnostic dictionary matrix is reverted back to its original condition to check other pairs of states. As discussed earlier, it is guaranteed that after an allowed binding, the problem is feasible, i.e., there exists at least one allowed pair of states to be bound together. To sum up, using the technique above at each iteration, one can construct feasible test sequences. This method will also work if system states are not uniquely isolatable by identifying the ambiguity groups first and then allowing those groups to be at the leaves in the algorithm.

When there exist asymmetrical tests, a feasibility check is applied similarly, but the diagnostic matrix should be converted to the appropriate form. For example, the problem in Table III is converted to the appropriate form in Table IV.

During the feasibility check procedure, there is no need to classify a node of ambiguity if all states in this set originally belong to the same state. If a set of ambiguity only consists of s_i and $s_{i'}$, although not classified completely, it does not ruin the allowance of the binding since they originally belong to the same state s_i. In other words, the decision tree does not have to classify a set, consisting of s_i variants (i.e., s_i, $s_{i'}$, $s_{i''}$, and so on), further.

B. How to Make Use of Look-Ahead Heuristic Algorithms

The heuristic described in this section is based on binding and rollout strategies. At each iteration, allowance check is applied automatically. An average cost Ω_{ij}^{H} is estimated for each pair via a heuristic test-sequencing algorithm, H, and the most promising selection is made that is similar to the rollout strategy of [16] but the decision tree is constructed bottom-up. Any one-step or multistep look-ahead algorithm can be employed as the heuristic test-sequencing algorithm in this strategy. At each iteration Ω_{ij} is calculated as

$$\Omega_{ij}^{\mathrm{H}} = (p_i + p_j)u_{ij} + H_{i,j}^{*} \tag{5}$$

where u_{ij} is the test with minimum cost that can bind sets i and j. If sets i and j belong to the same state, $u_{ij} = 0$, otherwise

$$u_{ij} = \min_{k}\{c_k | d_{ik} + d_{jk} = 1\}. \tag{6}$$

The term $(p_i + p_j)u_{ij}$ in (5) gives the exact cost of binding i and j and $H_{i,j}^{*}$ gives an estimate for the cost of the decision tree excluding the test used for classifying i and j. This operation is simply performed by temporarily binding i and j, updating diagnostic dictionary matrix, and applying a heuristic test-sequencing algorithm, H for an average cost estimate. Note that the heuristic test-sequencing algorithm is applied in the traditional way (i.e., tree is constructed normally, from top to bottom after i and j are strictly bound together). If the updated problem is infeasible, (i.e., no applicable test exists in a node of ambiguity when applying the heuristic test-sequencing algorithm), then $H_{i,j}^{*} = \infty$.

```
Algorithm Bottom-Up
  INPUT: Diagnostic dictionary matrix $D = [d_{ij}]$, cost vector $C$, prior conditional probability vector $P$.
  OUTPUT: Binary decision tree.
Step 0: Construct a set of current states $S_c = S$.
Step 1: For each state pair $i$ and $j$ in $S_c$, compute $\Omega_{ij}^{\mathrm{H}}$.
Step 2: Choose the pair with smallest $\Omega_{ij}^{\mathrm{H}}$ set $i^* = i$ and $j^* = j$.
Step 3: Permanently insert a new state, say $s_k$, where $p(s_k) =$
        $p(s_{i^*}) + p(s_{j^*})$.
Step 4: If $d_{i^*l} \neq d_{j^*l}$, set $d_{kl} = 2$, otherwise $d_{kl} = d_{i^*l}$.
Step 5: Permanently delete rows corresponding to $s_{i^*}$ and $s_{j^*}$.
Step 6: Set $S_c = S_c \cup s_k - (s_{i^*} \cup s_{j^*})$. If $S_c$ consists of one
        state TERMINATE ALGORITHM, otherwise go to
        Step 1.
```

Lemma 2: When the bottom-up algorithm is applied via any heuristic test-sequencing algorithm H, one would not obtain a worse solution than when H is applied on its own.

Proof: Suppose that T_{H} is the binary decision tree constructed via heuristic test-sequencing algorithm H and the average cost is J^{H}. The bottom-up algorithm starts to construct the decision tree $T_{\mathrm{Bottom-Up}}$ from the bottom of the AND/OR tree via heuristic test-sequencing algorithm H. Suppose that at any iteration, J' is the total cost of bindings performed up to that iteration. Then, at any iteration

$$J^{\mathrm{H}} = J' + \Omega_{ij}^{\mathrm{H}} \tag{7}$$

if the bindings performed up to that iteration are consistent with T_{H} and H proposes to bind i and j at that iteration.

The bottom-up algorithm calculates Ω values for each pair, and therefore $T_{\mathrm{Bottom-Up}}$ deviates from T_{H} at an iteration only if

$$\Omega_{i^*j^*}^{\mathrm{H}} \leq \Omega_{ij}^{\mathrm{H}} \tag{8}$$

where the bottom-up algorithm proposes to bind i^* and j^*, and H proposes to bind i and j. Adding the cost of bindings performed up to that iteration to both sides in (8)

$$\Omega_{i^*j^*}^{\mathrm{H}} + J' \leq \Omega_{ij}^{\mathrm{H}} + J'$$

and using (7)

$$\Omega_{i^*j^*}^{\mathrm{H}} + J' \leq \Omega_{ij}^{\mathrm{H}} + J' = J^{\mathrm{H}}$$
$$\Omega_{i^*j^*}^{\mathrm{H}} + J' \leq J^{\mathrm{H}}.$$

Since Ω gives an upper bound for the upcoming cost in the bottom-up algorithm

$$J^{\mathrm{Bottom-Up}} \leq \Omega_{i^*j^*}^{\mathrm{H}} + J' \leq J^{\mathrm{H}}$$

and finally

$$J^{\mathrm{Bottom-Up}} \leq J^{\mathrm{H}}. \tag{9}$$

■

C. Computational Complexity of the Algorithm

For a test-sequencing problem with m states and n tests, there exist m iterations (i.e., bindings) for the bottom-up algorithm in the worst case. At each iteration the algorithm performs the following:

1) at most $\binom{m}{2}$ heuristic test-sequencing algorithm, H implementations;
2) selection of the minimum of these $\binom{m}{2}$ estimates.

Suppose that the complexity of heuristic test-sequencing algorithm employed is $O(H)$. The computational complexity of bottom-up algorithm, $O(\text{Bottom-Up})$ is

$$\begin{aligned} O(\text{Bottom-Up}) &= O\left(m\left(m^2 O(H) + m^2\right)\right) \\ &= O\left(m^3 O(H)\right). \end{aligned} \tag{10}$$

Note that the algorithm essentially iterates by making updates on the diagnostic dictionary matrix. Therefore, the space complexity to execute the algorithm is $O(mn)$.

V. Computational Results

The computers used for this paper have an IntelXeon 2.00-GHz CPU and 1.00-GB RAM. The algorithms are coded in C++.

TABLE V
SIZE OF TEST PROBLEMS

Problems	Number of Classes (m)	Number of Tests (n)
PROB1, PROB11, PROB21	6	4
PROB2, PROB12, PROB22	10	5
PROB3, PROB13, PROB23	15	8
PROB4, PROB14, PROB24	20	10
PROB5, PROB15, PROB25	25	15
PROB6, PROB16, PROB26	15*	8
PROB7, PROB17, PROB27	25*	15

TABLE VI
ROLLOUT HEURISTIC VERSUS BOTTOM-UP HEURISTIC VIA INFORMATION HEURISTIC

	Information Heuristic			Rollout Heuristic			Bottom-Up Heuristic		
Problems	OPT	AVG	MAX	OPT	AVG	MAX	OPT	AVG	MAX
PROB1	31	0.695	9.226	40	0	0	40	0	0
PROB11	19	1.213	11.482	33	0.136	1.408	39	0.009	0.346
PROB21	26	1.583	14.662	34	0.385	5.963	37	0.103	2.072
PROB2	27	0.741	4.478	33	0.218	3.482	35	0.111	3.482
PROB12	11	1.49	5.265	30	0.326	3.788	31	0.169	2.007
PROB22	16	1.952	12.103	21	0.968	4.536	34	0.13	1.611
PROB3	15	1.243	6.468	23	0.289	2.2	29	0.121	1.827
PROB13	4	1.938	7.264	10	0.859	3.316	17	0.427	2.353
PROB23	6	2.719	8.65	17	1.292	6.639	23	0.247	1.412
PROB4	15	1.053	4.824	22	0.252	1.467	27	0.147	1.309
PROB14	3	1.967	4.849	4	0.798	3.188	13	0.26	1.742
PROB24	7	2.33	6.812	13	0.86	3.632	23	0.186	1.823
PROB5	11	1.127	4.289	18	0.332	1.75	26	0.158	0.987
PROB15	3	1.936	5.148	6	0.777	2.336	13	0.286	1.632
PROB25	1	3.189	12.706	5	1.415	7.229	17	0.343	1.336
PROB6	4	3.137	9.524	10	1.178	4.298	21	0.467	2.005
PROB16	0	4.072	9.716	2	2.332	7.897	17	0.447	3.059
PROB26	3	3.341	8.938	10	1.366	5.831	22	0.528	3.225
PROB7	0	5.21	11.434	1	2.692	6.132	13	0.808	5.88
PROB17	0	4.679	9.263	0	2.598	5.939	4	0.806	2.977
PROB27	0	3.687	10.467	1	1.609	6.772	9	0.534	2.181

A. *Closeness to Optimal Solutions for Rollout Heuristic and Bottom-Up Heuristic*

The experiments are conducted with two one-step look-ahead algorithms; information heuristic developed by Johnson [11] and separation heuristic [12] that is slightly modified where at each node of ambiguity a test t_k is selected that maximizes $(p(x_{jp}) \cdot p(x_{jf})/c_j)$. These heuristics are then used to construct the decision trees by RH and bottom-up algorithm. Finally, the optimal solution is obtained using enumeration.

To obtain a standard set of problems for comparison of bottom-up heuristic, a problem generator is developed to generate random problem instances of various sizes and properties. The test bed of problems consists of a set of small problems and a set of larger problems, whose sizes are shown in Table V. Problems, whose number of classes are marked with asterisk, include asymmetrical tests. The test bed consists of problems from three categories.

1) The first category is a set of problems with costs equal to 1. [PROB1–PROB7]
2) The second category is a set of problems with costs ranging from 25 to 30. [PROB11–PROB17]
3) The third category is a set of problems with costs ranging from 1 to 50. [PROB21–PROB27]

For each problem set, ten random runs are performed where state probabilities are uniform (i.e., $p(s_i) = (1/m + 1)\ \forall i$). In the second set of ten runs, relative state probabilities are uniformly distributed in the interval (0.3, 0.7). In the next set of ten runs, relative state probabilities are uniformly distributed in the interval (0, 1). Finally, ten random runs are performed where relative probabilities of half of the states are uniformly distributed in the interval (0.1, 0.2) and other half of the states are uniformly distributed in the interval (0.8, 0.9). Therefore, each problem set consists of 40 independent runs and their averages are reported.

Note that in Tables VI and VII, OPT is the number of times the heuristic reached the optimal solution among 40 runs, MAX is the maximum percentage error of the heuristic, and AVG is the average percentage error of the heuristic.

B. *Effect of Asymmetrical Tests*

It is seen in Tables VI and VII when there are asymmetrical tests, average solution quality of the heuristics, that do not have any special tool to handle this property of tests, decreases rapidly.

In Section V-A, when asymmetrical tests are introduced, $\lceil 6m/5 \rceil$ rows are generated for m states. In other words, for a problem set with m states and n asymmetrical tests, a $\lceil 6m/5 \rceil \times n$ diagnostic dictionary matrix, an appropriate cost and probability matrices are created according to the relevant distributions. Then, for the first m rows, the state numbers are set to the row numbers. Finally, for the last $\lceil m/5 \rceil$ rows, state numbers are randomly chosen from a set $S = \{1, \ldots, m\}$ which satisfies the asymmetrical property for the problem. For instance when there are 15 states, 18 rows are generated, but states can be 1, 2, ..., or 15.

In this experiment, number of states and tests are fixed and the solution quality is monitored when number of rows change. This is identical to increasing the degree of asymmetry.

TABLE VII
ROLLOUT HEURISTIC VERSUS BOTTOM-UP HEURISTIC VIA MODIFIED SEPARATION HEURISTIC

	Modified Separation Heuristic			Rollout Heuristic			Bottom-Up Heuristic		
Problems	OPT	AVG	MAX	OPT	AVG	MAX	OPT	AVG	MAX
PROB1	31	0.695	9.226	40	0	0	40	0	0
PROB11	27	0.856	11.482	38	0.016	0.393	39	0.009	0.346
PROB21	33	0.344	5.208	40	0	0	39	0.052	2.072
PROB2	27	0.741	4.478	33	0.218	3.482	35	0.111	3.482
PROB12	18	0.787	4.635	33	0.203	3.748	36	0.078	2.007
PROB22	27	0.573	4.545	32	0.371	4.493	35	0.107	1.59
PROB3	15	1.243	6.468	23	0.289	2.2	29	0.121	1.827
PROB13	11	0.824	4.843	25	0.202	1.636	33	0.053	0.944
PROB23	11	0.74	3.26	27	0.193	1.459	31	0.122	1.407
PROB4	15	1.053	4.824	22	0.252	1.467	27	0.147	1.309
PROB14	6	1.033	3.537	16	0.345	2.272	19	0.133	0.903
PROB24	14	0.677	3.506	23	0.199	1.408	30	0.07	0.606
PROB5	11	1.127	4.289	18	0.332	1.75	26	0.158	0.987
PROB15	2	1.149	3.877	10	0.386	1.974	23	0.157	1.51
PROB25	8	0.759	3.206	19	0.231	1.317	26	0.116	0.918
PROB6	4	3.137	9.524	10	1.178	4.298	21	0.467	2.005
PROB16	3	2.596	7.774	10	1.259	6.347	25	0.16	1.159
PROB26	9	1.089	6.427	18	0.547	5.405	32	0.079	0.692
PROB7	0	5.21	11.434	1	2.692	6.132	13	0.808	5.88
PROB17	0	3.423	7.669	0	1.641	3.223	16	0.43	2.461
PROB27	2	1.245	3.738	9	0.58	2.181	22	0.123	0.732

TABLE VIII
DEGREE OF ASYMMETRY CHANGE FOR 25 STATES

$n = 10$	Rollout Heuristic			Bottom-Up Heuristic		
Number of rows	OPT	AVG	MAX	OPT	AVG	MAX
25	17	0.319	2.065	18	0.166	1.363
30	1	1.441	3.607	19	0.412	2.812
35	0	2.017	7.527	16	0.628	3.624
40	0	2.38	4.854	11	0.664	1.969
50	0	2.547	4.217	2	0.839	3.33

TABLE IX
DEGREE OF ASYMMETRY CHANGE FOR 30 STATES

$n = 15$	Rollout Heuristic			Bottom-Up Heuristic		
Number of rows	OPT	AVG	MAX	OPT	AVG	MAX
30	12	0.558	2.387	18	0.339	1.571
36	1	2.664	5.541	14	0.664	3.865
42	0	3.132	5.607	10	0.696	3.351
48	0	3.51	7.687	3	1.319	3.834
60	0	3.632	6.809	2	1.651	4.186

The experiments are conducted using information heuristic. Again 40 independent runs are performed and the probability distributions are the same with that of the runs in Section V-A. Costs of tests are uniform and the results are reported in Tables VIII and IX.

C. Larger Problems

In this experiment, relatively large problems are studied. However, only a head-to-head comparison between the solutions obtained from the rollout heuristic and bottom-up heuristic is made instead of solution qualities since it is impossible to obtain optimal solutions in acceptable time periods for such instances. The experiments are conducted using an information heuristic and 40 independent runs, whose probability distributions are the same with that of the runs in Section V-A, are performed for each set.

In Table X, the numbers in parentheses show that how many times that heuristic obtains solutions better than the other one and the problems, whose number of classes are marked with asterisk, include asymmetrical tests. Costs of tests are uniformly distributed between 25 and 30.

TABLE X
HEAD-TO-HEAD COMPARISON FOR LARGER PROBLEMS

	Rollout Heuristic	Bottom-Up Heuristic
Problems	Average cost of the decision tree ()	Average cost of the decision tree ()
$m = 30^{*}, n = 15$	132.411 (4)	130.767 (36)
$m = 40, n = 20$	136.148 (3)	135.614 (34)
$m = 50, n = 20$	143.689 (6)	143.155 (29)
$m = 60^{*}, n = 30$	154.557 (1)	153.08 (39)
$m = 80^{*}, n = 35$	165.126 (0)	163.635 (40)

In general, our approach produces better solutions than rollout heuristic. Also, for small to medium problems it produced solutions very close to the optimal solution. The improvements in percentages may seem low but the impact can be large depending on the frequency of the usage of the strategy or the value of the costs.

The power of this approach is its effort to see the whole picture. In traditional methods, the algorithms are myopic, because the test that currently seems to be the best is selected at each iteration. However, there may not exist the desired tests with reasonable costs to use later. These methods usually do not take the next move into account, hence may come up with solutions far away from optimum. In the bottom-up approach, the algorithm performs a move considering the next moves. Exact cost of the current move is added to the average estimated cost of upcoming moves, which is the typical property of the rollout strategy.

Algorithms described in the literature construct the tree starting from the first test to perform which is the most critical one. It is obvious that, in a classification problem, a test that is performed earlier in sequential progress (i.e., in the upper parts of decision tree) is more critical than a test performed later. Since there exist more states in a node of ambiguity in the upper parts of a tree, cost contribution in that part is more than the lower parts of the tree. Also, the corresponding column of the diagnostic dictionary matrix affects the solution much more if a test is performed earlier. It is not a good idea to guess the answer without binding the pieces together. In the bottom-up algorithm, relatively unimportant decisions are performed first, which makes it much more flexible compared to the traditional methods. This strategy also provides the possibility of the existence of multiple correct decisions at a current iteration that lead to good results.

The drawback of this algorithm is the additional computational work to perform at each iteration. Yet, the time required for the algorithm to solve even the largest problem instances does not exceed 4 min, and it took less than 1 min to solve all other instances. Considering that this algorithm is run only once for an application, the increase in quality of solutions may be worth the increase in time required for certain applications.

VI. Conclusion

In this paper, we considered the fault diagnosis problem with asymmetrical tests and developed algorithms that are based on ideas of Huffman coding. The algorithms differ from previously proposed algorithms in the literature fundamentally in the way the solution is produced. Namely, the binary decision trees are constructed bottom-up by tieing/binding system states. The experimental results show that the proposed algorithm performs very well with respect to the optimal solutions and the results obtained by other solution procedures proposed in the literature.

To the best of our knowledge, this is the first study that proposes building the decision trees bottom-up. Therefore, there are various research directions that could be pursued. First of all, one could explore ways to speed up the construction process. Another direction is to consider precedence constraints. In this case, checking the feasibility of a binding is much more complicated than the case considered in this paper.

References

[1] K. R. Pattipati and M. G. Alexandridis, “Application of heuristic search and information theory to sequential fault diagnosis,” *IEEE Trans. Syst., Man, Cybern.*, vol. 20, no. 4, pp. 872–887, Jul./Aug. 1990.

[2] V. Raghavan, M. Shakeri, and K. R. Pattipati, “Optimal and near-optimal test sequencing algorithms with realistic test models,” *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 29, no. 1, pp. 11–26, Jan. 1999.

[3] A. Biassizo, A. Žužek, and F. Novak, “Sequential diagnosis with asymmetrical tests,” *Comput. J.*, vol. 41, no. 3, pp. 163–170, Apr. 1998.

[4] A. Žužek, A. Biassizo, and F. Novak, “Sequential diagnosis tool,” *Microprocess. Microsyst.*, vol. 24, no. 4, pp. 191–197, Aug. 2000.

[5] M. Mastrianni, “Virtual integrated test and diagnostics,” *Sikorsky Pres. ARPA*, Nov. 9, 1994.

[6] K. R. Pattipati and M. Dontamsetty, “On a generalized test sequencing problem,” *IEEE Trans. Syst., Man, Cybern.*, vol. 22, no. 2, pp. 392–396, Mar./Apr. 1992.

[7] M. R. Garey, “Optimal binary identification procedures,” *SIAM J. Appl. Math.*, vol. 23, no. 2, pp. 173–186, Jul. 1972.

[8] K. Fraughnaugh, J. Ryan, H. Zullo, and L. A. Cox, Jr., “Heuristics for efficient classification,” *Ann. Oper. Res.*, vol. 78, no. 1, pp. 189–200, 1998.

[9] D. A. Huffman, “A method for the construction of minimum-redundancy codes,” *Proc. IRE*, vol. 40, no. 9, pp. 1098–1101, Sep. 1952.

[10] V. Raghavan, M. Shakeri, and K. R. Pattipati, “Test sequencing problems arising in test planning and design for testability,” *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 29, no. 2, pp. 153–163, Mar. 1999.

[11] R. A. Johnson, “An information theory approach to diagnosis,” in *Proc. 6th Symp. Rel. Quality Control*, 1960, vol. 24, pp. 102–109.

[12] M. R. Garey and R. L. Graham, “Performance bounds on the splitting algorithm for binary testing,” *Acta Inform.*, vol. 3, pp. 347–355, 1974.

[13] P. K. Varshney, C. R. P. Hartmann, and J. M. DeFaria, Jr., “Application of information theory to sequential fault diagnosis,” *IEEE Trans. Comput.*, vol. C-31, no. 2, pp. 164–170, Feb. 1982.

[14] A. Martelli and U. Montanari, “Additive and/or graphs,” in *Proc. 3rd Int. Conf. Artif. Intell.*, Stanford, CA, Aug. 1973, pp. 1–11.

[15] M. Shakeri, V. Raghavan, K. R. Pattipati, and A. Patterson-Hine, “Sequential testing algorithms for multiple fault diagnosis,” *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 29, no. 2, pp. 153–163, Mar. 1999.

[16] F. Tu and K. R. Pattipati, “Rollout strategies for sequential fault diagnosis,” *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 33, no. 1, pp. 86–99, Jan. 2003.

[17] T. Ünlüyurt, “Sequential testing of complex systems: A review,” *Discrete Appl. Math.*, vol. 142, no. 1–3, pp. 189–205, 2004.

O. Erhun Kundakcioglu received the B.S. degree from Bilkent University, Ankara, Turkey, in 2002 and the M.S. degree from Sabancı University, Istanbul, Turkey, in 2004, both in industrial engineering.

He is currently a Graduate Student/Research Assistant with the Industrial and Systems Engineering Department, University of Florida, Gainesville. His primary research interests are in the areas of data mining, sequential fault diagnosis, reliability analysis, and wireless communications.

Tonguç Ünlüyurt received the B.S. degree in industrial engineering from Bilkent University, Ankara, Turkey, in 1994 and the Ph.D. degree in operations research from Rutgers University, New Brunswick, NJ, in 1999.

He is currently an Assistant Professor with Sabancı University, Istanbul, Turkey. His research interests include applications of combinatorial optimization in telecommunications, logistics, and manufacturing in addition to sequential fault diagnosis.