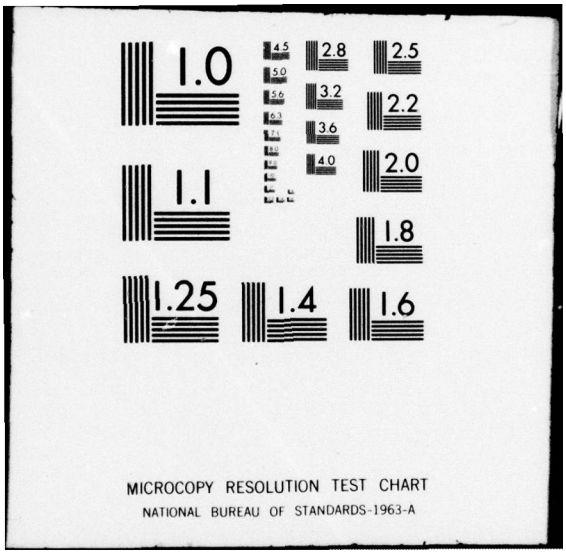
AD-A057 952 CALIFORNIA UNIV BERKELEY OPERATIONS RESEARCH CENTER F/G 12/2
OPTIMAL TESTING PROCEDURES FOR COHERENT SYSTEMS.(U)
SEP 77 Y BEN-DOV N00014-75-C-0781
UNCLASSIFIED ORC-77-23 NL

END
DATE FILMED
10-78
DDC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
6.3
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

LEVEL II

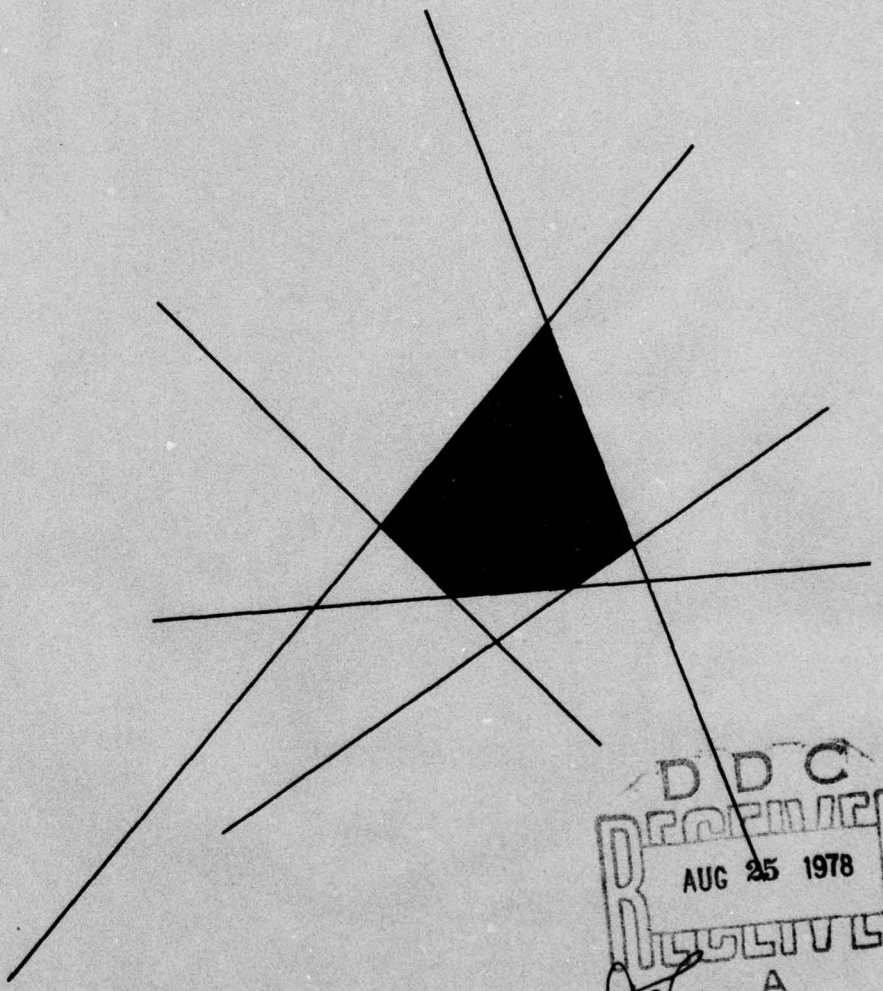
ORC 77-23
SEPTEMBER 1977

(12)

OPTIMAL TESTING PROCEDURES FOR COHERENT SYSTEMS

by

YOSI BEN-DOV



OPERATIONS
RESEARCH
CENTER

UNIVERSITY OF CALIFORNIA · BERKELEY

OPTIMAL TESTING PROCEDURES FOR COHERENT SYSTEMS

by

Yosi Ben-Dov
Operations Research Center
University of California, Berkeley

SEPTEMBER 1977 ORC 77-23

This research has been partially supported by the Office of Naval Research under Contract N00014-75-C-0781 and the Air Force Office of Scientific Research (AFSC), USAF, under Grant AFOSR-77-3179 with the University of California. Reproduction in whole or in part is permitted for any purpose of the United States Government.

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
(14) REPORT NUMBER: ORC-77-23	2. GOVT ACCESSION NO. / 3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle): (6) OPTIMAL TESTING PROCEDURES FOR COHERENT SYSTEMS	5. TYPE OF REPORT & PERIOD COVERED: (9) Research Report / 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): (10) Yosi Ben-Dov	8. CONTRACT OR GRANT NUMBER(s): AFOSR-77-3179
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Operations Research Center, University of California, Berkeley, California 94720	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: (16) 2304/A5 (17) A5
11. CONTROLLING OFFICE NAME AND ADDRESS: United States Air Force, Air Force Office of Scientific Research, Bolling AFB, D.C. 20332	(11) 12. REPORT DATE: September 1977 / 13. NUMBER OF PAGES: 25
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office): 61102F	15. SECURITY CLASS. (of this report): Unclassified / 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report): (12) 26p. Approved for public release; distribution unlimited.	
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)	
(15) N00014-75-C-0781, AFOSR-77-3179	
18. SUPPLEMENTARY NOTES: Also supported by the Office of Naval Research under Contract N00014-75-C-0781.	
19. KEY WORDS (Continue on reverse side if necessary and identify by block number): Coherent Systems; Optimal Testing Procedures; Branch and Bound	
20. ABSTRACT (Continue on reverse side if necessary and identify by block number): (SEE ABSTRACT)	

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

S/N 0102-LF-014-6601

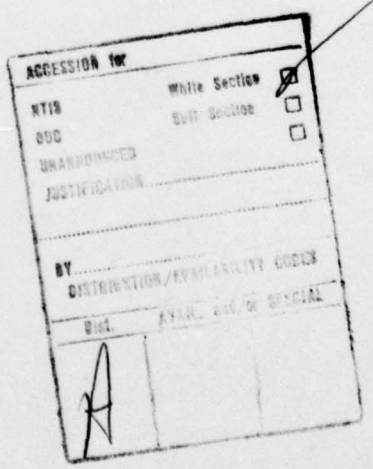
Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

270 750

ACKNOWLEDGEMENT

The author is grateful to Professor R. E. Barlow for all the help he rendered to make this work possible.



ABSTRACT

THIS REPORT DEALS WITH

We are interested in the problem of minimizing the expected cost of testing a coherent system. The concept of the Importance of Components is used to develop a branch and bound algorithm which determines the optimal testing policy for any coherent system.

OPTIMAL TESTING PROCEDURES FOR COHERENT SYSTEMS

by

Yosi Ben-Dov

1. INTRODUCTION

We are interested in the problem of minimizing the expected cost of testing a coherent system. Our system is composed of n components that either work or fail, and its structure function is monotonically increasing in each argument. By "testing" a system, we mean determining its state which can either be "functioning" or "failed." We do the actual testing by checking the components of the system in an optimal sequence. Components can be individually tested and tests give perfect information. Associated with each component is a cost for testing it $(c_i , i = 1,2, \ldots, n)$ and a priori probability that it is functioning $(p_i , i = 1,2, \ldots, n)$. (The a priori probability that a component is failed is given by $q_i = 1 - p_i$, $i = 1,2, \ldots, n$.) Components are assumed to function or fail independently of each other. The problem is, therefore, to find the optimal testing policy (which can be represented as a decision tree).

R. Butterworth [3] solved this problem for the special cases of series and parallel systems, and gave a sufficient condition for the optimality of his procedure for k-out-of-n systems. (Such a system works if and only if at least k of its n components work.) J. Halpern [4] presented an optimal sequential procedure for the k-out-of-n system with equal testing costs for all components. H. Lambert [5] dealt with the problem of minimizing the expected time of testing a failed system. He suggested a criteria for solving the problem for general systems, and showed how to use it specifically for the cases of series and parallel systems.

In Section 2, we define the concept of Importance of Components which is used as the decision criterion for a Branch and Bound algorithm. This algorithm is based on a paper by Little et al. [6], who presented a branch and bound algorithm for the Travelling Salesman problem, and on the paper by Reinwald and Soland [7]. In Section 3, we explain the rationale of our algorithm, and prove its optimality. Finally, in Section 4, we describe the algorithm itself, and give an example of its operation.

2. RELIABILITY IMPORTANCE OF COMPONENTS

The measure of Reliability Importance takes into account the structural importance of the components as well as their failure probabilities. This measure tells us how much we can improve the system reliability by improving the reliability of any single component. Specifically, we shall use Birnbaum's definition of importance [2].

Definition:

The Reliability Importance I_j of component j is given by:

$$I_j = \frac{\partial h(\underline{P})}{\partial p_j} \tag{2.1}$$

where $h(\underline{P})$ is the reliability function of the system, and $\underline{P}$ is the vector of probabilities: $\underline{P} = (p_1, \ldots, p_j, \ldots, p_n)$.

Using the well known identity which holds for the reliability function: $h(\underline{P}) = p_i h(1_i, \underline{P}) + q_i h(0_i, \underline{P})$ for $i = 1, \ldots, n$, we can get a definition equivalent to (2.1):

$$I_j = h(1_j, \underline{P}) - h(0_j, \underline{P}) \tag{2.2}$$

or more explicitly:

$$I_j = E[\phi(1_j, \underline{X}) - \phi(0_j, \underline{X})] = P\{[\phi(1_j, \underline{X}) - \phi(0_j, \underline{X})] = 1\} \tag{2.3}$$

where

$$x_i = \begin{cases} 0 & \text{when the } i^{th} \text{ component is failed} \\ 1 & \text{when the } i^{th} \text{ component is functioning} \end{cases}$$

and $\phi(\underline{X})$ is the structure function of the system which satisfies:

$$\phi(\underline{X}) = \begin{cases} 0 & \text{when the system is failed} \\ 1 & \text{when the system is functioning.} \end{cases}$$

Remark:

If $p_i = \frac{1}{2}$ for every i, then: $I_j = \left.\frac{\partial h(\underline{P})}{\partial p_j}\right|_{\underline{P}=(\frac{1}{2},\ldots,\frac{1}{2})}$ which gives us Birnbaum's definition of Structural Importance of the components, i.e.: $I_\phi(j) = \frac{1}{2^{n-1}} \sum_{\{\underline{X}|x_j=1\}} [\phi(1_j,\underline{X}) - \phi(0_j,\underline{X})]$. Note that $\sum_{\{\underline{X}|x_j=1\}} [\phi(1_j,\underline{X}) - \phi(0_j,\underline{X})]$ is the total number of state vectors in which component j is critical to the system, in the sense that for those vectors the system will be functioning if and only if component j will be functioning.

Examples:

For the following examples, assume that the components of the system have been labeled such that their probabilities are ordered as:

$$p_1 \leq p_2 \leq \cdots \leq p_n .$$

1. Series System:

The reliability function of a series system is given by:

$h(\underline{P}) = \prod_{i=1}^{n} p_i$ and thus: $I_j = \prod_{i\neq j} p_i$. Therefore: $I_1 \geq I_2 \geq \cdots \geq I_n$, i.e., the least reliable component is the most important in the system. A series system is functioning if and only if all of its components are functioning, so that the failure of any component causes a system failure

and the system is no better than its weakest component. Hence, the weakest component in the system is the most important component.

2. Parallel System:

$h(\underline{P}) = 1 - \prod_{i=1}^{n} (1 - p_i)$ thus: $I_j = \prod_{i \neq j} (1 - p_i)$. Therefore:

$I_1 \leq I_2 \leq \ldots \leq I_n$, i.e., the most reliable component is the most important in the system. This, again, is very intuitive, since a parallel system functions if at least one of its components functions. Hence, the strongest component in the system is the most important.

3. 2-out-of-3 System:

The reliability function of a 2-out-of-3 system is given by:

$h(\underline{P}) = p_1p_2 + p_1p_3 + p_2p_3 - 2p_1p_2p_3$, *and thus:*

$$I_1 = p_2 + p_3 - 2p_2p_3$$

$$I_2 = p_1 + p_3 - 2p_1p_3$$

$$I_3 = p_1 + p_2 - 2p_1p_2 \ .$$

If $p_i = \frac{1}{2}$ for $i = 1 , 2 , 3$, then: $I_1 = I_2 = I_3$, since all the components are equally important as far as the structure of the system is concerned.

If $p_i \leq \frac{1}{2}$ for $i = 1 , 2 , 3$, then: $I_1 \geq I_2 \geq I_3$, i.e., the weakest component is the most important for the system.

If $p_i \geq \frac{1}{2}$ for $i = 1 , 2 , 3$, then: $I_1 \leq I_2 \leq I_3$, i.e., the most important component is the one with the highest reliability.

Otherwise, $I_2 \geq \max \{I_1, I_3\}$, i.e., the most important component is the one whose probability of working is neither the highest nor the smallest among the three components.

3. THE REASONING BEHIND THE ALGORITHM

Let us first define a measure of "unimportance" of component j :

$$\bar{I}_j = 1 - I_j \tag{3.1}$$

and investigate its meaning. Suppose we are given two state vectors: $\underline{X}^1$ and $\underline{X}^2$ such that:

(i) $\phi(\underline{X}^1) = \phi(\underline{X}^2)$ and

(ii) $x_i^1 = x_i^2$ for every $i \neq i_o$, but $x_{i_o}^1 \neq x_{i_o}^2$.

We want to find the value of $\phi(\underline{X}^1)$ (or, $\phi(\underline{X}^2)$) , and to do so by testing the individual components. One way of doing this would be to test all the components in an arbitrary order and thus find the value of $\phi(\underline{X}^1)$. This would definitely be unnecessarily costly, since the value of $\phi(\underline{X}^1)$ is independent of the value of x_{i_o} , and we therefore need not check this component.

The measure of unimportance which we defined can be described as follows: we find the total number of vectors for which the checking of a specific component is unnecessary, and then look at the probability of the occurrence of those vectors, i.e.: Prob {j is unimportant} = $\bar{I}_j$ = $1 - I_j = P\{[\phi(1_j,\underline{X}) - \phi(0_j,\underline{X})] = 0\}$.

Next we define:

$$d_j = c_j \bar{I}_j \tag{3.2}$$

which is a measure of the expected extra cost due to testing component j even though it is unnecessary to know its state. Thus: $e_j = c_j I_j$ is a

measure of the expected necessary expense due to testing component j. This is equal to the cost of testing component j minus the expected cost of the extra expense, i.e., $e_j = c_j I_j = c_j - d_j$. We define the total expected necessary expense to be:

$$(3.3) \quad I = \sum_k c_k I_k = \sum_k c_k (1 - \bar{I}_k) = \sum_k c_k - \sum_k c_k \bar{I}_k = \sum_k c_k - \sum_k d_k .$$

Finally, we introduce some additional notation:

$$(3.4) \quad I_j(i) = h(1_j, 1_i, \underline{P}) - h(0_j, 1_i, \underline{P})$$

$$(3.5) \quad I_j(i') = h(1_j, 0_i, \underline{P}) - h(0_j, 0_i, \underline{P})$$

where $I_j(i)$ and $I_j(i')$ are the measures of the importance of component j when component i is given to be functioning or failed respectively. Also, let:

$$(3.6) \quad d_j(i) = c_j \bar{I}_j(i)$$

and

$$(3.7) \quad d_j(i') = c_j \bar{I}_j(i') .$$

In general, we define:

$$(3.8) \quad I_j(\underline{Z}, s) = h(1_j, \underline{Z}, 1_s, \underline{P}) - h(0_j, \underline{Z}, 1_s, \underline{P})$$

$$(3.9) \quad I_j(\underline{Z}, s') = h(1_j, \underline{Z}, 0_s, \underline{P}) - h(0_j, \underline{Z}, 0_s, \underline{P})$$

where $I_j(\underline{Z}, s)$ and $I_j(\underline{Z}, s')$ are the measures of the importance of component j, given the vector of known component states $\underline{Z}$, and component

s is given to be functioning or failed respectively. Also, let:

$$d_j(\underline{Z},s) = c_j \bar{I}_j(\underline{Z},s) \tag{3.10}$$

and

$$d_j(\underline{Z},s') = c_j \bar{I}_j(\underline{Z},s') \ . \tag{3.11}$$

Following the above discussion, one can suggest the following algorithm: start the testing procedure by checking component i, which minimizes d_k for $k = 1,2, \ldots, n$. Then continue to develop the testing tree by choosing the components that minimize $d_j(i)$ and $d_k(i')$ for $k, j = 1,2, \ldots, n$; $k, j \neq i$, and continue in this manner. This method, however, does not necessarily give an optimal testing procedure. Actually, we can show by induction (see Theorem 1) that $I + d_i$ is a lower bound for the expected cost of all testing procedures which start by testing component i. As a corollary, it is easy to see that $I + \min_i \{d_i\}$ is the minimum expected cost of *all* testing procedures. Therefore, we must modify the "naive" algorithm suggested above in order to be able to consider testing procedures which have a higher lower bound, but yet may give a smaller expected cost.

Before we prove Theorem 1, we present a lemma which states that testing component i first does not change the expected extra cost of testing any of the other components.

Lemma:

$$\sum_{k \neq i} d_k = p_i \sum_k d_k(i) + q_i \sum_k d_k(i') \ .$$

Proof:

$$(3.12)\qquad \sum_{k\neq i} d_k = \sum_{k\neq i} c_k \bar{I}_k = \sum_{k\neq i} c_k\{1 - [h(1_k,\underline{P}) - h(0_k,\underline{P})]\} .$$

$$\sum_k d_k(i) = \sum_{k\neq i} c_k \bar{I}_k(i) = \sum_{k\neq i} c_k\{1 - [h(1_k,1_i,\underline{P}) - h(0_k,1_i,\underline{P})]\}$$

$$\sum_k d_k(i') = \sum_{k\neq i} c_k \bar{I}_k(i') = \sum_{k\neq i} c_k\{1 - [h(1_k,0_i,\underline{P}) - h(0_k,0_i,\underline{P})]\}$$

$$p_i \sum_k d_k(i) + q_i \sum_k d_k(i') = \sum_{k\neq i} c_k - \sum_{k\neq i} c_k\{p_i[h(1_k,1_i,\underline{P}) - h(0_k,1_i,\underline{P})]$$

$$+ q_i[h(1_k,0_i,\underline{P}) - h(0_k,0_i,\underline{P})]\}$$

$$= \sum_{k\neq i} c_k - \sum_{k\neq i} c_k\{[p_i h(1_k,1_i,\underline{P}) + q_i h(1_k,0_i,\underline{P})]$$

$$(3.13)\qquad - [p_i h(0_k,1_i,\underline{P}) + q_i h(0_k,0_i,\underline{P})]\}$$

but, recalling that:

$$h(\underline{P}) = p_i h(1_i,\underline{P}) + q_i h(0_i,\underline{P})$$

we also have:

$$h(1_k,\underline{P}) = p_i h(1_k,1_i,\underline{P}) - q_i h(1_k,0_i,\underline{P})$$

and:

$$h(0_k,\underline{P}) = p_i h(0_k,1_i,\underline{P}) - q_i h(0_k,0_i,\underline{P})$$

$$\therefore\quad (3.13) = \sum_{k\neq i} c_k - \sum_{k\neq i} c_k[h(1_k,\underline{P}) - h(0_k,\underline{P})]$$

$$= \sum_{k\neq i} c_k\{1 - [h(1_k,\underline{P}) - h(0_k,\underline{P})]\} = (3.12).\blacksquare$$

Theorem 1:

A lower bound for the expected cost of all testing procedures which start by testing component i, is: $I + d_i$.

Proof:

We use an induction proof, where the induction is based on n, the number of components in the system.

For $n = 1$, the proof is trivial. For $n = 2$, there are two possible systems with two components: a series system and a parallel system. We study them in turn.

1. Series System:

$$h(\underline{P}) = p_1 p_2$$

$$d_1 = c_1(1 - I_1) = c_1\{1 - [h(1_1,p_2) - h(0_1,p_2)]\} = c_1(1 - p_2 + 0) = c_1 q_2$$

$$d_2 = c_2(1 - I_2) = c_2\{1 - [h(p_1,1_2) - h(p_1,0_2)]\} = c_2(1 - p_1 + 0) = c_2 q_1$$

$$\therefore \quad I = \sum_{i=1}^{2} c_i - \sum_{i=1}^{2} d_i = c_1 + c_2 - c_1 q_2 - c_2 q_1 = c_1 p_2 + c_2 p_1 .$$

Finally:

$$I + d_1 = c_1 p_2 + c_2 p_1 + c_1 q_2 = c_1 + p_1 c_2$$

$$I + d_2 = c_1 p_2 + c_2 p_1 + c_2 q_1 = c_2 + p_2 c_1 .$$

Those lower bounds are actually equal to the expected cost of testing the system, when we start by checking component 1 or 2 respectively.

2. Parallel System:

$$h(\underline{P}) = p_1p_2 - p_1p_2$$

$$d_1 = c_1(1 - I_1) = c_1\{1 - [h(1_1,p_2) - h(0_1,p_2)]\} = c_1(1 - 1 + p_2) = c_1p_2$$

$$d_2 = c_2(1 - I_2) = c_2\{1 - [h(p_1,1_2) - h(p_1,0_2)]\} = c_2(1 - 1 + p_1) = c_2p_1$$

$$\therefore \quad I = \sum_{i=1}^{2} c_i - \sum_{i=1}^{2} d_i = c_1 + c_2 - c_1p_2 - c_2p_1 = c_1q_2 + c_2q_1 \ .$$

Therefore:

$$I + d_1 = c_1q_2 + c_2q_1 + c_1p_2 = c_1 + q_1c_2$$

$$I + d_2 = c_1q_2 + c_2q_1 + c_2p_1 = c_2 + q_2c_1 \ .$$

Here, again, the lower bounds are equal to the expected costs.

Now we assume that the theorem is true for systems with $(n - 1)$ components, and show it is true for systems with n components. Assume that the first component to be tested is i, and let C_i be the optimal expected cost of testing the system, if we start the testing procedure by checking component i first. i can be found to be functioning or failing and we denote by $C(i)$ and $C(i')$ the optimal costs of testing the subtrees that can be developed respectively. Then we have the equation:

(3.14) $$C_i = c_i + p_iC(i) + q_iC(i') \ .$$

We want to show that: $C_i \geq I + d_i$ for every $i = 1,2, \ldots, n$. By the induction hypothesis, we have:

$$C(i) \geq I(i) = \sum_{k \neq i} c_k - \sum_k d_k(i)$$

$$C(i') \geq I(i') = \sum_{k \neq i} c_k - \sum_k d_k(i')$$

$$\therefore \quad C_i \geq c_i + p_i\left[\sum_{k \neq i} c_k - \sum_k d_k(i)\right] + q_i\left[\sum_{k \neq i} c_k - \sum_k d_k(i')\right]$$

$$= \sum_{k \neq i} c_k - \left[p_i \sum_k d_k(i) + q_i \sum_k d_k(i')\right]$$

$$= \sum_k c_k - \sum_{k \neq i} d_k \quad \text{(by the lemma)}$$

$$= \sum_k c_k - \sum_k d_k + d_i$$

$$= I + d_i \ . \blacksquare$$

Corollary:

A lower bound for the expected cost of all the testing procedures is given by: $I + \min_i \{d_i\}$.

The next theorem allows us to speed up the calculations of the lower bounds in each iteration of our algorithm.

Theorem 2:

A lower bound for the expected cost of testing the system--given that we start the testing procedure by checking component i , and then testing components α or β depending on whether $x_i = 1$ or $x_i = 0$ respectively--is given by:

(3.15) $$L_i = I + d_i + p_i d_\alpha(i) + q_i d_\beta(i') \ .$$

Proof:

$$L_i = c_i + p_i\left[\sum_{k\neq i} c_k - \sum_k d_k(i) + d_\alpha(i)\right] + q_i\left[\sum_{k\neq i} c_k - \sum_k d_k(i') + d_\beta(i')\right]$$

$$= \sum_k c_k - \left[p_i \sum_k d_k(i) + q_i \sum_k d_k(i')\right] + p_i d_\alpha(i) + q_i d_\beta(i')$$

$$= \sum_k c_k - \sum_{k\neq i} d_k + p_i d_\alpha(i) + q_i d_\beta(i')$$

$$= I + d_i + p_i d_\alpha(i) + q_i d_\beta(i') \quad . \blacksquare$$

A branch and bound algorithm seems to be suitable for our problem, and we use the measure of unimportance that we defined in (3.1) as the decision criterion for its operation. The algorithm is described in the next section.

4. BRANCH AND BOUND ALGORITHM

To establish the optimal testing procedure we need to develop an optimal tree that will tell us which component to test first, and according to the result of the first test, or any other subsequent tests where to continue in our testing procedure. The algorithm works by developing subtrees, such that at each iteration the subtree on hand is the one which gives the lowest bound of the type that we defined earlier.

We start the algorithm with n such subtrees, which are the original components, and pick the one which minimizes $I + d_i$ over all $i = 1,2, \ldots, n$. With this component k, say, we continue by considering the $(n - 1)^2$ subtrees which consist of the component k branching to each combination of two out of the $(n - 1)$ remaining components. For each of these subtrees, which correspond to different possible decision rules, the left-hand branch goes to the component which is to be tested if k is found failed, and the right-hand branch goes to the component which is to be tested if k is found working.

After computing the bounds for each of these new subtrees, we find the minimum of all the bounds that we have computed so far, and continue to expand the corresponding subtree in a similar manner. This subtree may, at this stage, be either a single component, or a subtree which we have constructed.

Eventually, we will find a subtree which is a complete tree (i.e., it need not be expanded any more, and thus it uniquely describes a testing procedure for the given system) such that its expected cost is less than or equal to the lower bound for any other subtree. This will be the optimal testing tree.

The algorithm is described schematically in Figure 1.

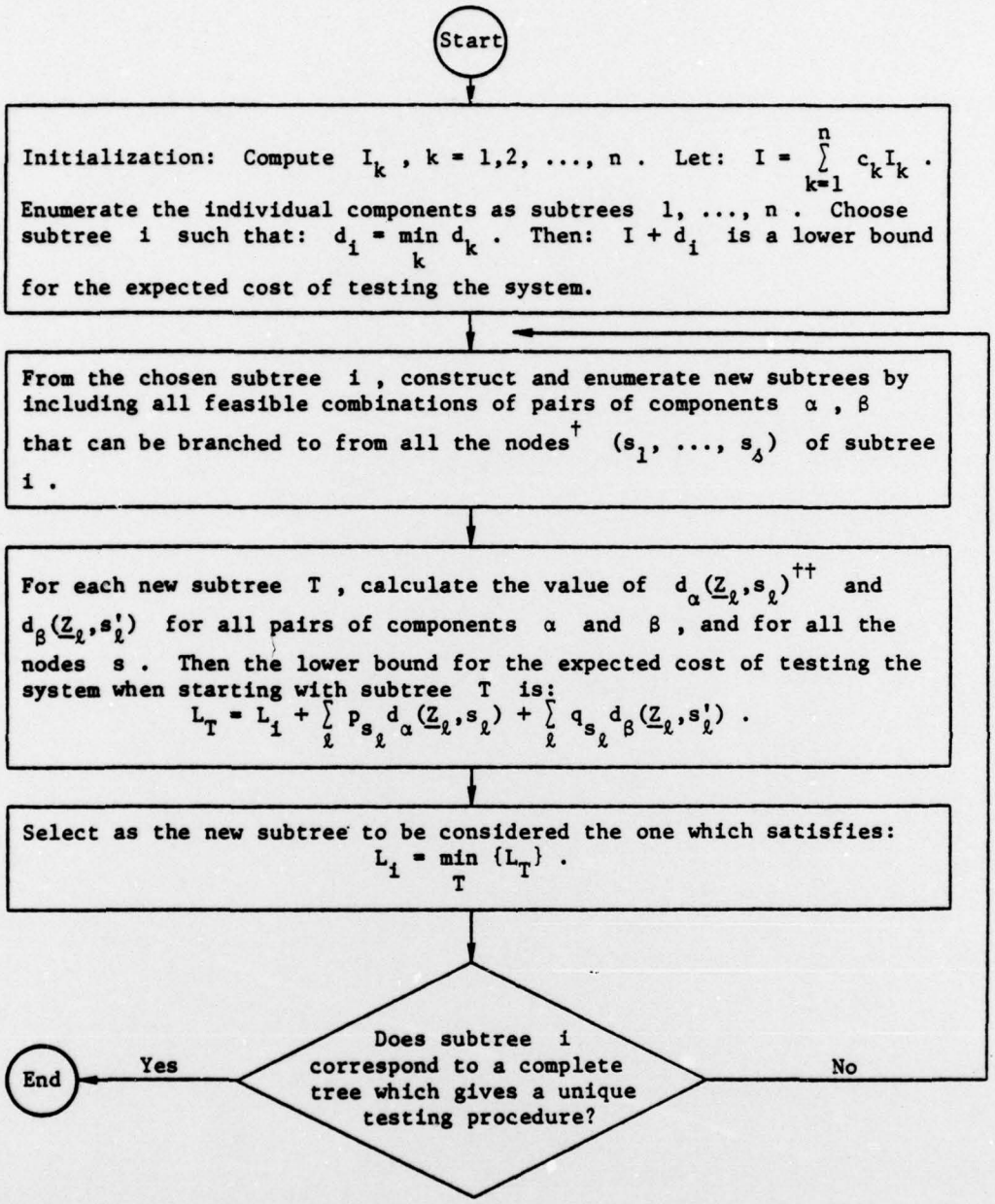


FIGURE 1

---

†Nodes are the lowest level components in a subtree.

††$\underline{Z}_\ell$ is the vector of component states which lead to node s_ℓ in subtree i .

Example:

Consider a 2-out-of-3 system (a k-out-of-n system works if and only if at least k out of its n components work), with the reliability function:

$$h(\underline{P}) = p_1p_2 + p_1p_3 + p_2p_3 - 2p_1p_2p_3$$

and the following data:

i	p_i	c_i
1	.6	5
2	.8	2
3	.9	4

Iteration #0 (Initialization):

We compute the Importance Measures for all the components:

$$I_1 = p_2 + p_3 - 2p_2p_3 = 1.7 - 1.44 = .26$$

$$I_2 = p_1 + p_3 - 2p_1p_3 = 1.5 - 1.08 = .42$$

$$I_3 = p_1 + p_2 - 2p_1p_2 = 1.4 - .96 = .44$$

$$I = \sum_{k=1}^{3} c_k I_k = 3.9 .$$

Iteration #1:

Compute the lower bounds as defined in Theorem 1, for all the given components:

$$L_1 = I + d_1 = I + c_1(1 - I_1) = 3.9 + 3.7 = 7.6$$

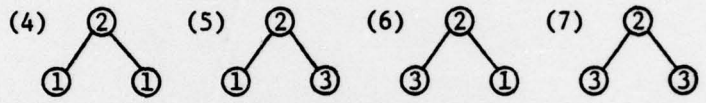
$$L_2 = I + d_2 = I + c_2(1 - I_2) = 3.9 + 1.16 = 5.06$$

$$L_3 = I + d_3 = I + c_3(1 - I_3) = 3.9 + 2.24 = 6.14 .$$

Since $L_2 = \min \{L_i\}$, we continue the algorithm by developing a subtree whose root is component 2.

Iteration #2:

Compute the lower bounds, as defined by (3.15), for the following trees:



First, we have to compute $d_i(2)$ and $d_i(2')$ for $i = 1, 3$, as given by (3.6) and (3.7):

$$d_1(2) = c_1(1 - 1 + p_3) = p_3 c_1 = 4.5$$

$$d_1(2') = c_1(1 - p_3) = q_3 c_1 = .5$$

$$d_3(2) = c_3(1 - 1 + p_1) = p_1 c_3 = 2.4$$

$$d_3(2') = c_3(1 - p_1) = q_1 c_3 = 1.6 .$$

The new set of lower bounds will be:

$$L_4 = L_2 + p_2 d_1(2) + q_2 d_1(2') = 5.06 + 3.7 = 8.76$$

$$L_5 = L_2 + p_2 d_3(2) + q_2 d_1(2') = 5.06 + 2.02 = 7.08$$

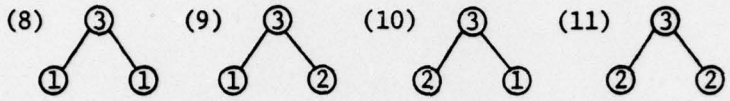
$$L_6 = L_2 + p_2 d_1(2) + q_2 d_3(2') = 5.06 + 3.92 = 8.98$$

$$L_7 = L_2 + p_2 d_3(2) + q_2 d_3(2') = 5.06 + 2.24 = 7.3 .$$

Now we see that $L_3 = \min_{i \neq 2} \{L_i\}$, and we have to develop all the trees that can be originated by component 3.

Iteration #3 (Termination):

Compute the lower bounds for the following trees:



We compute the new $d_i(3)$ and $d_i(3')$ for $i = 1, 2$:

$$d_1(3) = c_1(1 - 1 + p_2) = c_1p_2 = 4$$

$$d_1(3') = c_1(1 - p_2) = c_1q_2 = 2$$

$$d_2(3) = c_2(1 - 1 + p_1) = c_2p_1 = 1.2$$

$$d_2(3') = c_2(1 - p_1) = c_2q_1 = .8 .$$

The new set of lower bounds:

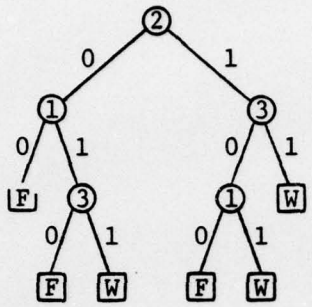
$$L_8 = L_3 + p_3d_1(3) + q_3d_1(3') = 6.14 + 3.8 = 9.94$$

$$L_9 = L_3 + p_3d_2(3) + q_3d_1(3') = 6.14 + 1.28 = 7.42$$

$$L_{10} = L_3 + p_3d_1(3) + q_3d_2(3') = 6.14 + 3.68 = 9.82$$

$$L_{11} = L_3 + p_3d_2(3) + q_3d_2(3') = 6.14 + 1.16 = 7.3 .$$

Finally, $L_5 = \min_{\substack{i=1,\ldots,11 \\ i\neq 2,3}} \{L_i\}$, and the tree given in (5) is equivalent to a complete tree for our problem. The complete testing procedure which is uniquely determined by (5) is:



where: F means the system fails, W means the system works and the minimum expected cost of testing the system is given by:

$$C = c_2 + (p_2 + q_2p_1)c_3 + (q_2 + p_2q_3)c_1 = 2 + 3.68 + 1.4 = 7.08 .$$

REFERENCES

[1] Barlow, R. E. and F. Proschan, STATISTICAL THEORY OF RELIABILITY AND LIFE TESTING, Holt, Rinehart and Winston, Inc., (1975).

[2] Birnbaum, Z. W., "On the Importance of Different Components in a Multi-Component System," MULTIVARIATE ANALYSIS-II, P. R. Krishnaiah, ed., Academic Press, pp. 581-592.

[3] Butterworth, R., "Some Reliability Fault-Testing Models," Operations Research, Vol. 20, pp. 335-343, (1972).

[4] Halpern, J., "Fault Testing for a k-out-of-n System," Operations Research, Vol. 22, pp. 1267-1271, (1974).

[5] Lambert, H. E., "Fault Trees for Decision Making in System Analysis," UCRL-51829, Lawrence Livermore Laboratory, (October 1975).

[6] Little, J. D. C., K. G. Murty, D. W. Sweeney and C. Karel, "An Algorithm for the Travelling Salesman Problem," Operations Research, Vol. 11, pp. 972-989, (1963).

[7] Reinwald, L. T. and R. M. Soland, "Conversion of Limited-Entry Decision Tables to Optimal Computer Programs I: Minimum Average Processing Time," Journal of the Association for Computing Machinery, Vol. 13, pp. 339-358, (1966).