AD-A099 999 OKLAHOMA UNIV NORMAN SCHOOL OF INDUSTRIAL ENGINEERING F/G 9/2
OPTIMUM TESTING PROCEDURES FOR SYSTEM DIAGNOSIS AND FAULT ISOLA--ETC(U)
MAR 81 A A ALY AFOSR-80-0139
UNCLASSIFIED 81-1 NL

1 OF 1
AD
A099999

END
DATE
FILMED
6 81
DTIC

AFOSR-TR- 81-0470

(3)

AD A099999

OPTIMUM TESTING PROCEDURES
FOR SYSTEM DIAGNOSIS
AND FAULT ISOLATION*

LEVEL

by

Adel A. Aly
Principal Investigator

DTIC
ELECTE
JUN 1 0 1981
D
C

Mar[redacted]81

School of Industrial Engineering
University of Oklahoma
Norman, Oklahoma 73019

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

This research was supported in part by the U.S. Air Force Office of Scientific Research, Bolling AFB, Wash., C.D. 20332 under grant number AFOSR-80-0139

*Paper presented at the "International Conference on Production Engineering, Design and Control," Alexandria, Egypt, December 27-29, 1980.

DTIC FILE COPY

81 6 10 040

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFOSR-TR- 81-0470	2. GOVT ACCESSION NO. AD-A099999	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) "OPTIMUM TESTING PROCEDURES FOR SYSTEM DIAGNOSIS AND FAULT ISOLATION"		5. TYPE OF REPORT & PERIOD COVERED Final 2/1/80 - 1/31/81
		6. PERFORMING ORG. REPORT NUMBER 81-1
7. AUTHOR(s) Dr. Adel A. Aly		8. CONTRACT OR GRANT NUMBER(s) AFOSR-80-0139
9. PERFORMING ORGANIZATION NAME AND ADDRESS University of Oklahoma School of Industrial Engineering 202 W. Boyd, Suite 124 Norman, Oklahoma 73019		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 61102F 2304/D9
11. CONTROLLING OFFICE NAME AND ADDRESS Air Force Office of Scientific Research/NM Building 410, Bolling AFB, Wash. D.C. 20332		12. REPORT DATE March 31, 1981
		13. NUMBER OF PAGES 92
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

N/A

18. SUPPLEMENTARY NOTES

Paper presented at the "International Conference on Production Engineering, Design and Control," Alexandria, Egypt, December 27-29, 1980.

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Fault detection and isolation
Built-in-test
Optimum sequenceof testing
Branch-and Bound

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Even though a great deal of work has been done in developing models in the field of designing diagnostic tests for fault isolation in digital systems, there is still a lack of efficient and fast procedures.

Two approaches to the cost-effective design of fault isolation procedures were presented here. They were oriented specifically toward built-in-test (BIT) diagnostic subsystems for modular electronic equipment.

A branch and bound solution approach was used in order to find the optimal

20. (cont'd)

sequence of tests to be executed by the BIT to isolate a single malfunctioned unit among a group of line replaceable units. Computational results were presented and discussed. A computer program listing of the solution technique was included.

ABSTRACT

Even though a great deal of work has been done in developing models in the field of designing diagnostic tests for fault isolation in digital systems, there is still a lack of efficient and fast procedures.

Two approaches to the cost-effective design of fault isolation procedures were presented here. They were oriented specifically toward built-in-test (BIT) diagnostic subsystems for modular electronic equipment.

A branch and bound solution approach was used in order to find the optimal sequence of tests to be executed by the BIT to isolate a single malfunctioned unit among a group of line replaceable units. Computational results were presented and discussed. A computer program listing of the solution technique was included.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Introduction

In recent years, interest has grown in the development and use of automatic devices to test and checkout physical systems of all types. Most of the attention in this field has been oriented toward built-in-test (BIT) diagnostic subsystems for modular military electronic equipment, mainly in airborne and ground electronic equipment. BIT diagnostics have the advantage of allowing fewer and less qualified maintenance personnel and fewer pieces of external test equipment, which are generally quite expensive.

A primary equipment is composed of modular line replaceable units (LRUs), all of which operate independently. Associated with each unit is an a priori probability of being in failure, and it is assumed that the probabilities of multiple failures are negligible.

Whenever the equipment malfunctions, a single LRU is assumed to have failed, and two types of diagnostic tests should be used for the primary and the secondary isolations. The primary isolation tests will be automatically executed by the BIT in order to identify the group of LRUs which contains the faulty unit. After the execution of the automatic BIT, secondary isolation will be performed by semi-automatic or manual means which incur time and extra equipment costs to locate the single failed unit within a group of LRUs.

1.2 Statement of the Problem

Assume that equipment consists of n mutually exclusive groups of LRUs. Associated with each LRU_i is an a priori probability p_i, which is the probability before any diagnosis that the malfunction of the equipment is caused by the failure of LRU_i. Whenever the equipment fails, the BIT automatically executes a sequence of primary diagnostic tests to isolate the group which contains the single faulty LRU.

Each LRU could be either good or bad, therefore a set of 2^n tests is required to constitute a complete set of all possible binary tests. However, if a test that checks a subset of LRUs is passed, the test that checks the complement of this subset must be a failure, and conversely. Therefore, such a pair of tests is redundant, and in the quest for least-expected-cost procedures the more expensive of the pair can be ignored. By this argument the number of possible different tests which can exist for n LRUs is $(2^{n-1} - 1)$ after excluding the two tests which examine all or none of the n LRUs.

Associated with each test, T_k which is included in the primary diagnostic, there is a known cost, $\bar{C}_k$. The total cost of locating a particular faulty element is the sum of the costs of the tests along the path which leads from the initial state, in which no LRU is known to be good or bad, to the final state representing the group containing the faulty unit, plus the cost of secondary isolation of that group.

The problem is to design minimum-expected-cost test procedures to be executed by the BIT. These can be described by tree structures, with nodes and twigs. Each node of a tree can be interpreted as a state of ambiguity subset. The ambiguity subset at each node consists of the twigs that are descendant from the node. The test applied at a node serves to partition the associated ambiguity subset, thus reducing the ambiguity. The root node, or full subset, corresponds to a state of complete ambiguity, while at the twigs, which correspond to unit subsets and hence where the outcome is determined, there is no further ambiguity.

1.3 Determination of States Following a Test

The following notation will be used throughout this research.

T_k: Test k. A test is represented by an (n-bit) number containing only the binary digits 0 and 1. A 0 is assigned in position i of a test if LRU_i must be good in order for the test to pass. A 1 is placed in position i of a test if LRU_i is not tested.

$\hat{S}$: State of the equipment prior to performing the test T_k. A state is represented by an (n-bit) number containing only the binary digits 0 and 1. The n bits in the designation of a state ocrrespond, sequentially from left to right, to LRU_1, LRU_2 . . . LRU_n. A 0 is assigned in position i of a state if LRU_i is known to be good. A 1 is assigned in position i of a state if LRU_i is not yet tested. In the initial state there are 1's in all positions since none of the LRUs have been tested.

$S(\hat{S},T_k)$: State of the equipment if test T_k passes. This state is computed by multiplying $\hat{S}$ and T_k bit by bit with no carry.

$\bar{S}(\hat{S},\bar{T}_k)$: State of the equipment if test T_k fails. This state is computed by multiplying $\hat{S}$ and $\bar{T}_k$, the complement of T_k, bit by bit without carry.

$n(S)$: Number of remaining untested LRUs at state S.

$\bar{C}_k$: Cost of test T_k.

$T(S)$: Set of all possible tests which could be used at state S.

$T(\hat{S},S)$: Set of all tests which could be used to reach state $S(\hat{S},T_k)$ from state $\hat{S}$.

$N(S)$: A node representing state S.

$I(S)$: Set of indices of the n(S) remaining untested LRUs at node N(S).

$b(\hat{S},S)$: A branch leading from node $N(\hat{S})$ to node N(S).

$\overset{*}{C}(S)$: Minimum expected cost of a sequence of tests, given that the current state is S.

$C(\hat{S},S)$: Expected cost of testing branch $b(\hat{S},S)$.

E_i: Expected cost for secondary isolation of LRU_i.

The basic structure of a sequential testing diagram is illustrated in Figure 1.1.

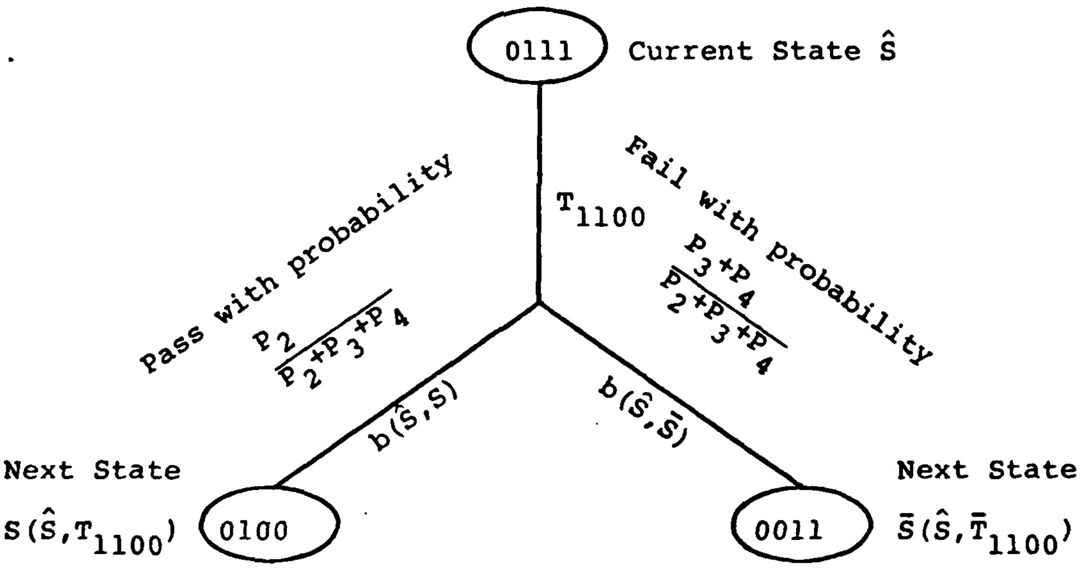


Figure 1.1. Sequential testing diagram for example of 4-LRUs.

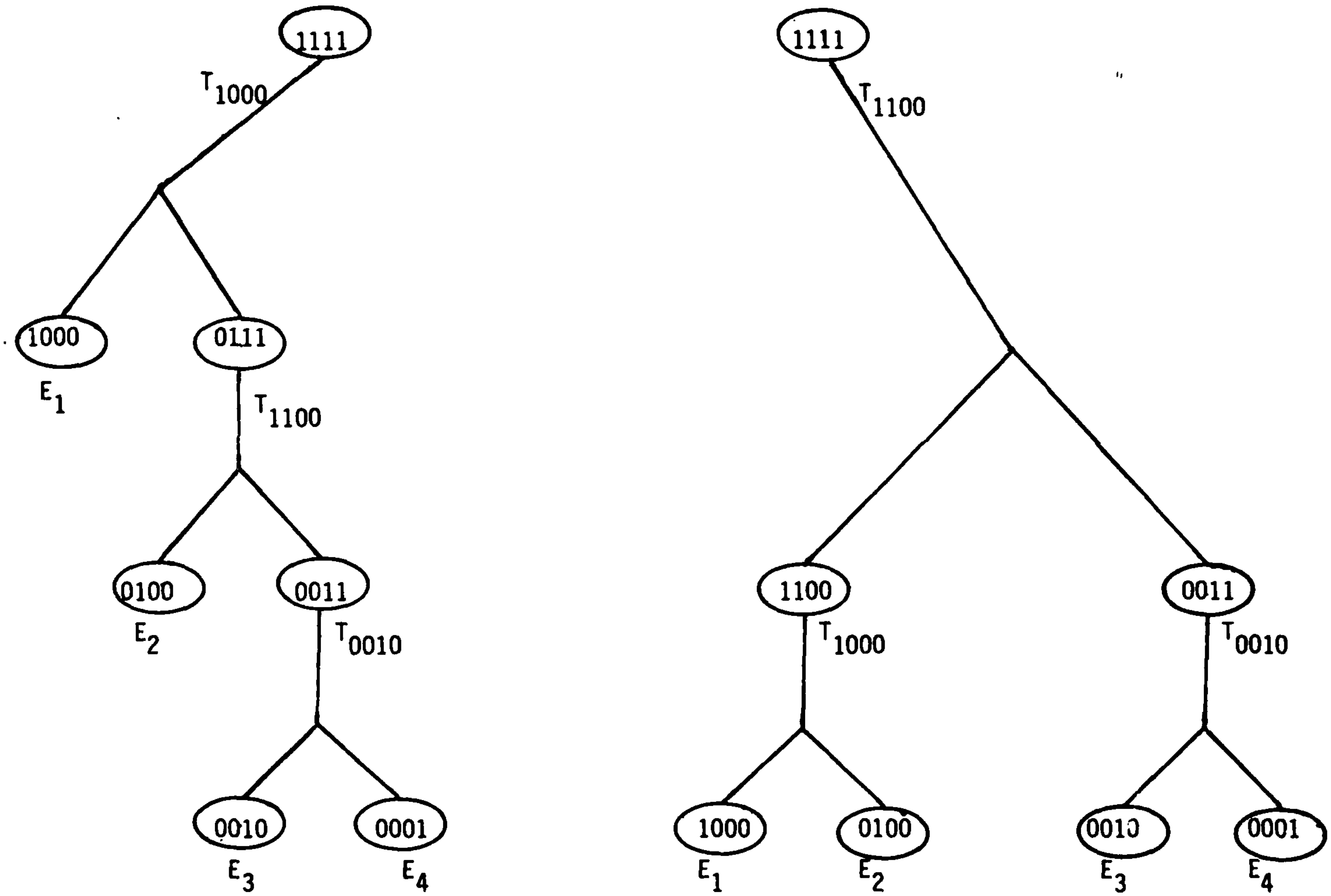
In order to explain how the expected cost of any sequential test procedure is computed, two feasible solutions to the example problem defined in Table 1.1 are shown in Figure 1.2.

TABLE 1.1

EXAMPLE PROBLEM

LRU_i	1	2	3	4
P_i	.45	.30	.20	.05
E_i	6	3	5	1

T_k	Binary Designation of Test				C_k
T_1	1	0	0	0	$ 1
T_2	0	1	0	0	6
T_3	0	0	1	0	2
T_4	0	0	0	1	7
T_5	1	1	0	0	3
T_6	1	0	1	0	5
T_7	1	0	0	1	4



(a) (b)

Figure 1.2 Two Feasible Testing Diagrams (a) and (b)

Let C_1 be the expected cost of the feasible test procedure $[T_{1000}, T_{1100}, T_{0010}]$ represented by tree (a) in Figure 1.2.

$$
\begin{aligned}
C_1 &= \overset{*}{C}(1111) \\
&= C_{1000} + p_1\,\overset{*}{C}(1000) + (p_2+p_3+p_4)\,\overset{*}{C}(0111) \\
&= C_{1000} + p_1\,\overset{*}{C}(1000) + (p_2+p_3+p_4)\,\Big(C_{1100} + \frac{p_2}{p_2+p_3+p_4}\overset{*}{C}(0100) \\
&\qquad + \frac{(p_3+p_4)}{p_2+p_3+p_4}\,\overset{*}{C}(0011)\Big) \\
&= C_{1000} + p_1\,\overset{*}{C}(1000) + p_2\,\overset{*}{C}(0100) + (p_2+p_3+p_4)\Big(C_{1100} \\
&\qquad + \frac{(p_3+p_4)}{p_2+p_3+p_4}\,\Big(C_{0010} + \frac{p_3}{p_3+p_4}\,\overset{*}{C}(0010) + \frac{p_4}{p_3+p_4}\overset{*}{C}(0001)\Big)\Big) \\
&= C_{1000} + p_1\,E_1 + p_2\,E_2 + p_3\,E_3 + p_4\,E_4 \\
&\qquad + (p_2+p_3+p_4)\,C_{1100} + (p_3+p_4)\,C_{0010} \\
&= 7.8
\end{aligned}
$$

Let C_2 be the expected cost of the feasible test procedure $[(T_{1100}, T_{1000}), \text{ or } (T_{1100}, T_{0010})]$ represented by tree (b) in Figure 1.2

$$
\begin{aligned}
C_2 &= \overset{*}{C}(1111) \\
&= C_{1100} + (p_1+p_2)\,\overset{*}{C}(1100) + (p_3+p_4)\,\overset{*}{C}(0011)
\end{aligned}
$$

$$= C_{1100} + (p_1+p_2)(C_{1000}+ \frac{p_1}{p_1+p_2} \overset{*}{C}(1000)+ \frac{p_2}{p_1+p_2}\overset{*}{C}(0100))$$

$$+ (p_3+p_4)(C_{0010}+ \frac{p_3}{p_3+p_4}\overset{*}{C}(0010)+ \frac{p_4}{p_3+p_4}\overset{*}{C}(0001))$$

$$= C_{1100} + p_1 E_1 + p_2 E_2 + p_3 E_3 + p_4 E_4$$

$$+ (p_1+p_2)C_{1000} + (p_3+p_4)C_{0010}$$

$$= 8.9$$

This example shows that the first sequential testing procedure [T_{1000}, T_{1100}, T_{0010}] is more economical than the second one [(T_{1100}, T_{1000}), or (T_{1100}, T_{0010})].

CHAPTER 2

LITERATURE REVIEW

All works which have been done in the area of the optimization of fault detection and isolation procedures are directed toward solving two basic problems.

1. Generating a least expected cost testing sequence to be executed by the automatic BIT diagnostic (Primary Isolation).

2. Determining a troubleshooting sequence which minimizes the expected cost of secondary isolation to locate the single failed unit within a group of LRUs identified by the BIT primary diagnostic.

The only other problem which has been treated in the literature is the one which restricts the repertoire of tests to those which test only single elements and without even assigning any probabilities to these tests. In this case, the solution consists of deciding which test to omit and in what sequence to perform the remaining tests. This problem can be solved as a machine setup problem as the one in Glassey [6].

2.1 Primary Isolation

Johnson, et al. [9] proposed using the information-gain figure-of-merit in order to find a sequence of tests that can be executed by an automatic diagnostic.

In spite of the fact that this method is easy to use, it fails to guarantee optimum cost sequence.

Chang [3] used the distinguishability criterion to produce a low expected cost testing sequence which is not necessarily an optimal sequence.

Cohn and Ott [4] presented a recursive algorithm which is based on the concept of dynamic programming. They used set notation to design a test tree. For every possible ambiguity subset, they assigned an evaluation, consisting of the least expected cost of resolving that ambiguity. The evaluation of the subset of complete ignorance is the cost of the optimal tree. This evaluation function is computed by a recursion on the number of elements in the ambiguity subsets.

By treating the equipment states as stages in a sequential decision process, Sheskin [11] applied probabilistic dynamic programming to determine a minimum expected cost testing diagram. Using the recursive relationship, the solution procedure moves backwards stage by stage. The solution procedure begins by equating the expected values of the terminal states, which corresponds to the groups into which the equipment is partitioned, to the expected costs of secondary isolation for these groups. At each state, a set of possible decisions consists of all of the tests which can be performed is considered and the optimal testing sequence at this state is found, until it finds the optimal testing diagram when starting at the initial state.

Aly [1] constructed the problem as a search tree, and presented a branch and bound algorithm to find the optimal testing sequence.

2.2 Secondary Isolation

Gluss [7] solved the problem of having a fault develop in a system consisting of n modules where each one has several elements, and that it is required to dictate a search strategy that will optimize the search in some fashion by minimizing a stipulated cost function. He developed a model, which assumes that over-all-tests of each module may be performed, and individual item tests within modules; also, the search is subject to the constraint that before conducting item tests the faulty module must first be determined by module tests.

Firstman and Gluss [5] extended the work in the previous model of Gluss, in which the estimation of the probabilities of faults lying in respective modules or elements is performed in a different way from that in Gluss' paper: they are computed from element reliability data by manipulation of the element failure rate. Furthermore, consideration is given to fault symptoms that are supplied by weighting the probabilities according to the symptoms information.

All the previous search models allow for the possibility of a test not indicating the true state of the component tested. However when Butterworth [2] tackled the problem, he used tests that always give the correct answer. He developed several rules to find the optimal sequential policies for series, and parallel systems of independent LRUs. For a series system, he indicated that the expected cost for secondary isolation of the failed unit, given that an equipment fault has been isolated to this unit by primary diagnostic will be minimized by removing and replacing the LRUs in a nonincreasing sequence of the values of the ratio of their probability of failure to the average time of removing and replacing them.

Butterworth's rules fail to identify an optimal policy for the simple system where the testing costs are identical for all components. In this case the condition implies that all the components have the same failure probabilities. However, Halpern [8] presented a simple adaptive sequential testing procedure for the k-out-of-n system with equal cost of all tests. This procedure covers the deficiency of Butterworth's rules.

2.3 Scope of the Research

From the above section, it is noticed that the only approaches which guarantee an optimum testing sequence are the recursive procedure by Cohn and Ott [4], the dynamic programming by Sheskin [1], and the branch and bound by Aly [1].

Capitalizing on Aly's approach, it seems very promising to formulate the problem as a search tree and to find the optimal testing sequence using a branch and bound approach. Using this approach efficiently could save a lot of work in comparison with using the two methods previously mentioned because of the savings in the solution space achieved by using strong dominance rules instead of finding the optimal solutions among all possible solutions at each node in the solution by dynamic programming for instance, as shown in Figure 2.1 for a four LRUs example which uses many arcs. The same problem could be formulated as the search tree depicted in Figure 2.2. However, all the arcs which lead to any state with only one untested LRU and which resulted from applying tests that remove the ambiguity of exactly one LRU are omitted in order to simplify both the network and the tree.

Also, by using a good lower bound at each node, most of the active nodes could be fathomed and the optimal solution could be found as fast as possible by using a strong branching rule.

The efficiency of the solution by using branch-and-bound approach depends upon the strength of the bounds, the dominance and the branching rules. Consequently, the main effort will be directed toward finding the dominance rules which minimize the number of branches as much as possible, finding the branching rules which concentrate the search only in the very promising branches, finding a lower bound at each node which helps in fathoming the maximum number of nodes, and constructing a sound and efficient branch-and-bound algorithm.

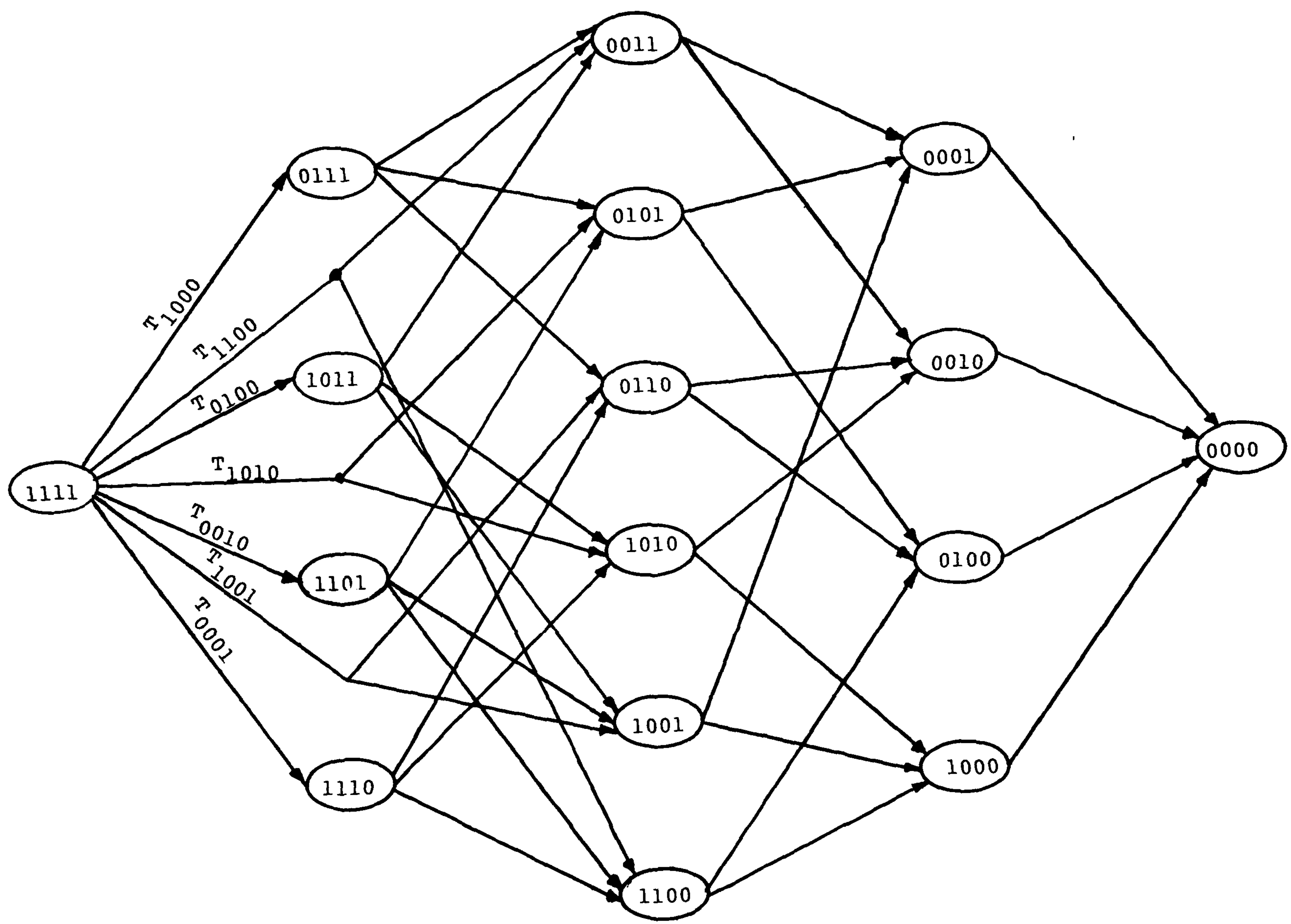


Figure 2.1 A Directed Network for a 4-LRUs Example

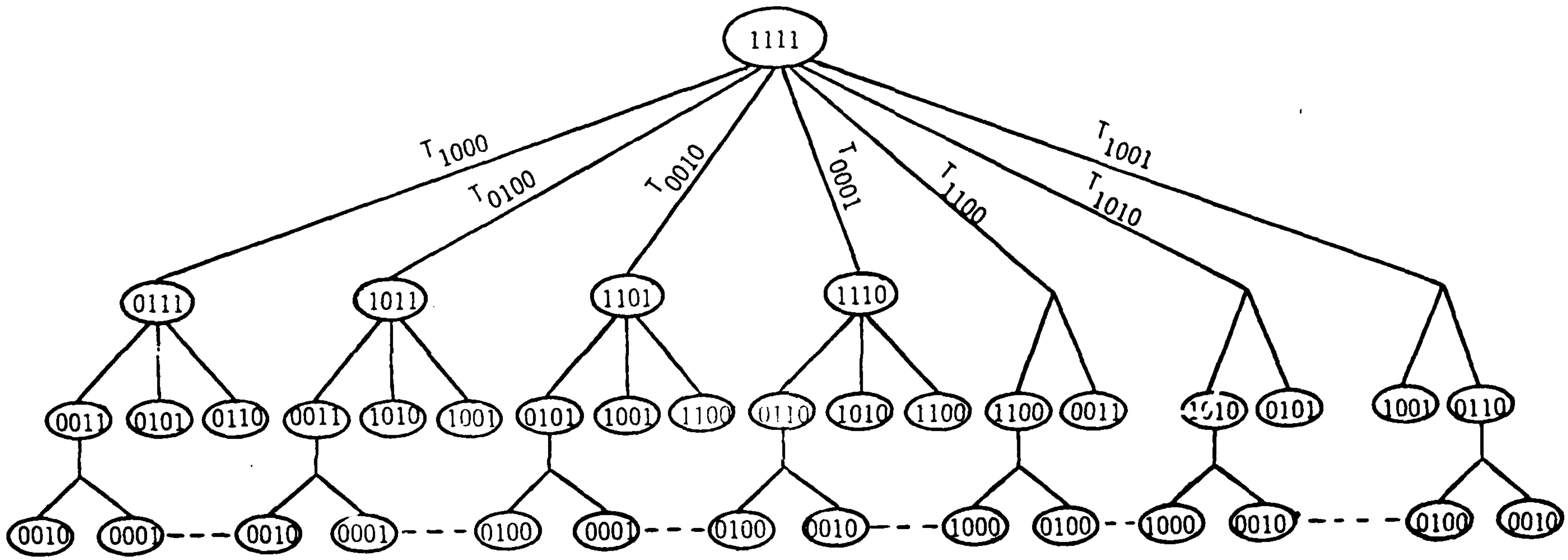


Figure 2.2 A Tree Diagram for a 4-LRUs Example

CHAPTER 3

A BRANCH AND BOUND APPROACH

In this chapter a branch and bound algorithm is developed to find the optimal sequence of diagnostic tests to be executed automatically to isolate the group of modules which contains the faulty unit.

3.1 Concept of Branch and Bound Techniques

As stated by Lawler and Wood [10], branch and bound is a method of controlled search of the space of all feasible solutions. The space of all feasible solutions is repeatedly partitioned into smaller and smaller subsets, and a lower bound (in the case of minimization) is calculated for the value of the objective function over the solutions within the subsets. If a known feasible solution is available (an upper bound), then after each partitioning those subsets with a lower bound exceeding the current upper bound are excluded from further consideration. Partitioning continues until a feasible solution is found such that its cost is not greater than the lower bound for any subset.

Branch and bound algorithms have two main characteristics; the branching and bounding characteristics. The branching characteristic guarantees that an optimal solution will eventually be obtained. The bounding characteristic furnishes the possibility of recognizing an optimal solution prior to complete enumeration.

Therefore, any branch and bound algorithm needs to define a set of rules for (1) branching from nodes to new nodes, (2) determining lower bounds for the new nodes, (3) choosing an intermediate node from which to branch next, (4) recognizing when a node contains only infeasible or non-optimal solutions, and (5) recognizing when a final node contains an optimal solution.

In order to use the branch and bound technique to find the optimal sequence of tests to be used in detecting and isolating the malfunctioned unit, the search tree is constructed as the one in Figure 2.2 with a few modifications. There is no need for all the nodes in the last level, which have states including only one untested LRU, since their status can be found once the search reaches any node with state of having exactly two untested LRUs regardless of its level. Consequently, savings can be made in both time and storage required for the solution. Also, at any node if a test which does not remove the ambiguity of exactly one unit (in other words it decreases the number of untested LRUs by more than one) is applied, the state of this node will be changed to another two states with more than one level difference between them and the given node. Therefore, a dummy or fictitious node will be added after the given node in order to keep track of the two new branches since the expected cost of applying this test should include the expected costs of both branches.

The modified search tree which is depicted in Figure 3.1 represents a four LRUs example ($n = 4$), with a maximum number of levels of $(n - 1)$. At any state S with $n(S)$ remaining untested LRUs there are $(2^{n(S)-1} - 1)$ branches emanating from this state. Each branch represents a set of possible tests which could be used to remove the same ambiguity, and consequently leads to the same new states at another level down the tree. Since our objective is to minimize the expected cost of search tests, always, the test with the minimum cost among all possible tests at every branch will be considered.

In order to save in the storage and time requirment, which are the main problems in this kind of combinatorial problem, the nodes (which represents the states) of the tree will not be generated in advance, but they will be generated one by one as the algorithm proceeds. This will not only save the number of nodes but also the size of information to be stored at each node. Once a feasible solution is found, all the remaining branches which are emanated from all active nodes should be checked. At the last generated node, all the previous branches and nodes which emanated from this node and which are already examined, fathomed, or had a feasible solution (which is to be stored) are to be cancelled and the search is to proceed in a new active branch. By repeating this procedure,

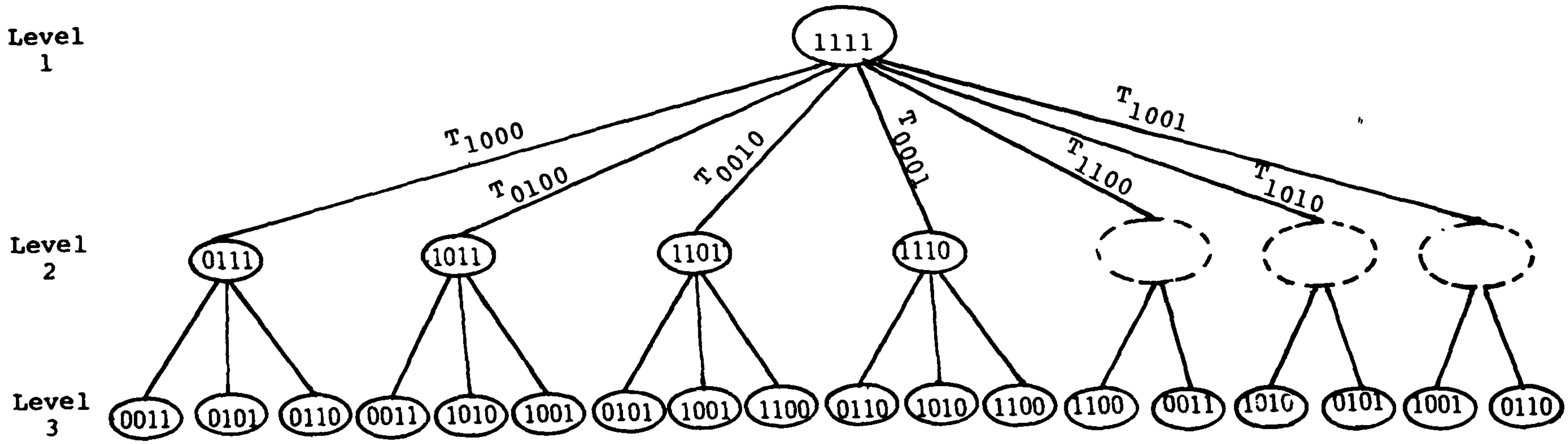


Figure 3.1 A Modified Search Tree for a 4-LRUs Example

the number of nodes stored at any time is minimum and relatively very small.

In the algorithm, the search starts by finding a feasible solution as quickly as possible by moving directly down the tree using the branching rules from the initial node at the first level to another node in a successive level, to a third one in a successive level, etc., until finding a state of having only two untested LRUs. Proceed upward in the same branches in order to update the values of the lower bounds at the fictitious nodes and, consequently, find the actual values of the lower bounds in the other branches of all fictitious nodes. This procedure guarantees finding a fast and good feasible solution which enables us to fathom efficiently many nodes, especially since the search proceeds in the most promising branch at each node according to the branching rules after applying the dominance rules, which eliminates as many branches as possible.

After finding a feasible solution, the algorithm proceeds by moving to the last created node and starting branching and bounding as usual until fathoming all nodes emanating from it; then going to the second from the last created node and so on until fathoming the first node in the tree. In this case the last solution corresponding to the last value of the upper bound is optimal.

3.2 Upper and Lower Bounds

3.2.1 Upper Bound at the First Node

On a minimization problem—like the problem presented here—developing a reasonable initial upper bound on the objective function value is important because it might help in fathoming nodes before even computing the first objective function value associated with a feasible search procedure generated by the tree. In this case the objective function value associated with any feasible procedure may serve as an upper bound.

Since the maximum number of tests required to find the malfunctioned unit among n LRUs is (n - 1), using the (n - 1) tests that have the minimum costs among all tests, which isolate only one LRU at the initial node, is sufficient to find the malfunctioned unit and consequently presents a feasible search scheme.

Noting that the cost of the tests in the objective function should be multiplied by the probabilities of the untested LRU's at each node in order to find the expected cost, neglecting the values of these probabilities (which are less than one), and taking into consideration the expected cost of the secondary isolation of all n LRUs, results in a value of a possible and reasonable upper bound.

This initial upper bound, U, is defined as

$$U = \sum_{k \in t} \bar{C}_k + \sum_{i=1}^{n} P_i \cdot E_i \tag{3.1}$$

where

$\bar{C}_k$ = cost of test k.

P_i = prior probability of failure of LRU_i.

E_i = expected cost for secondary isolation of LRU_i.

t = set of tests with the minimum (n - 1) costs among the n possible tests which isolate only one LRU if they are used at the first node (i.e., tests with (n - 1) zeros and only a single one in the n bits such as tests T_{1000}, T_{0100}, T_{0010}, T_{0001} in case of having only four LRUs).

3.2.2 A Lower Bound for Each Node

At each node in the tree a lower bound is computed based on the actual value of the expected costs of all tests used prior to reaching this node, as well as an estimate of the minimum expected cost of tests required to remove the ambiguity of all the remaining untested LRUs at this node.

At any node $N(\hat{S})$ applying test T_k will generate two nodes $N(S)$ and $N(\bar{S})$ corresponding to states $S(\hat{S},T_k)$ and $\bar{S}(\hat{S},\bar{T}_k)$ respectively which arises two cases according to the number of remaining untested LRUs at each node.

<u>Case 1</u> $\mathrm{Min}[n(S), n(\bar{S})] = 1$

In this case, there is no meaning of generating the node corresponding to the state of having only one remaining untested LRU and, consequently, there is only one branch to be searched, assuming it is the one starting with node N(S). Since the remaining untested LRUs at this node are n(S) therefore, at most (n(S) - 1) tests could be used to remove their ambiguity, and the cost of these tests should be multiplied by the sum of the probabilities of the untested LRUs at each of the (n(S) - 1) nodes.

Since $2 \leq n(S) \leq n$, then the minimum possible sum of probabilities to be multiplied by any cost is the sum of the minimum two probabilities of the remaining n(S) LRUs.

Let I(S) be the set of indices of the n(S) remaining untested LRUs at node N(S), the prior probabilities $\bar{p}_1, \bar{p}_2, \ldots, \bar{p}_{n(S)}$ of these LRUs are arranged in an ascending order such that $\bar{p}_1 < \bar{p}_2 \ldots < \bar{p}_{n(S)}$, and C(S) is defined as a lower bound of the minimum expected cost of tests required to remove the ambiguity of the n(S) untested LRUs at node N(S), then

$$C(S) = (\bar{p}_1 + \bar{p}_2) \cdot \sum_{j \varepsilon t_{n-1}} \bar{c}_j$$

where t_n = set of the n(S) tests with the minimum n(S) costs among all possible tests which could be used at this node.

Let $T(\hat{S},S)$ be the set of all tests which could be used to reach the state of node N(S) from the state of node $N(\hat{S})$

and $C(\hat{S},S)$ be the minimum expected cost of the test required to reach the state of node $N(S)$ from that of node $N(\hat{S})$, then

$$C(\hat{S},S) = \min_{k\varepsilon T(\hat{S},S)}[\bar{C}_k] \cdot \sum_{i\varepsilon I(\hat{S})} p_i$$

Based on the above discussion, a lower bound $L(S)$ at node $N(S)$ representing state $S(\hat{S},T_k)$ could be found by computing the expression

$$L(S) = L(\hat{S}) - C(\hat{S}) + C(S) + C(\hat{S},S) \qquad (3.2)$$

If $N(S)$ is the first node in the tree with state S having n untested LRUs then,

$$L(S) = C(S) + \sum_{i=1}^{n} p_i \cdot E_i \qquad (3.3)$$

<u>Case 2</u> $\mathrm{Min}(n(S),n(\bar{S})] > 1$

In this case, applying test T_k at node $N(\hat{S})$ will generate two nodes which should be both searched. Instead, a fictitious node $N(\hat{S}_f)$ corresponding to a dummy state $\hat{S}_f$ will be assumed to have resulted from applying test T_k at $N(\hat{S})$ and will be inserted after node $N(\hat{S})$. Then, the two nodes will be emanated from $N(\hat{S}_f)$ and generate the two branches $b(\hat{S}_f,S)$ and $b(\hat{S}_f,\bar{S})$ as shown in Figure 3.2.

Finding the lower bound at any fictitious node $N(\hat{S}_f)$ is slightly different from finding it at any other node, since it should include $C(S)$ if the search is moving downward in

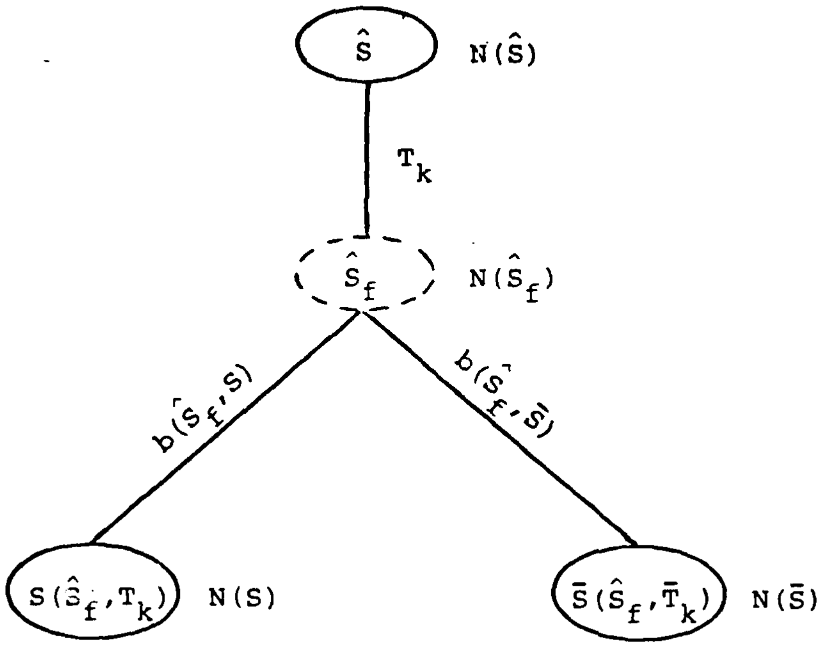


Figure 3.2. Sequential testing diagram using a fictitious node

branch $b(\hat{S}_f,\bar{S})$, and includes $C(\bar{S})$ if the search is moving downward in branch $b(\hat{S}_f,S)$. Arbitrary in the algorithm the search will move first to the node with the maximum number of remaining untested LRUs among nodes $N(S)$ and $N(\bar{S})$. Even though both branches should be searched, this procedure will minimize backtracking because it increases the possibility of fathoming more branches. So, if $n(S) > n(\bar{S})$ the search will move downward in branch $b(\hat{S}_f,S)$. Consequently, the lower bound at the fictitious node $N(\hat{S}_f)$ could be computed using the expression

$$L(\hat{S}_f) = L(\hat{S}) - C(\hat{S}) + C(\bar{S}) + C(\hat{S},\hat{S}_f) \qquad (3.4)$$

Since there are no tests required to change the state of node $N(\hat{S}_f)$ to the states of nodes $N(S)$ and $N(\bar{S})$, then $C(\hat{S}_f)$, $C(\hat{S}_f,S)$, and $C(\hat{S}_f,\bar{S})$ are all equal to zero.

A final word about the lower bounds. If the search reaches node $N(S)$ where $n(S) = 2$, then a feasible solution could be obtained. Let C be the expected cost of this feasible test sequence, then

$$C = L(S) - C(S) + \min_{j\varepsilon T(S)}[\bar{C}_j] \cdot \sum_{i\varepsilon I(S)} p_i \qquad (3.5)$$

C is the value of the actual expected cost of a feasible solution unless it results from any branch emanated from a fictitious node, in this case it is only a lower bound of the actual value of the expected cost of a feasible solution. To find the actual value the search should go upward the tree to the fictitious node and update its lower bound by substituting the last value of C instead of $C(\bar{S})$ in equation 3.4. Then, moving downward in the other branch $b(\hat{S}_f, \bar{T}_k)$, as in Figure 3.2, until finding a node with a state having only two untested LRUs. At this moment computing C using equation 3.5 results in the value of the actual expected cost of a feasible solution because in this case the cost of the two branches (emanated from a fictitious node) has been taken into consideration.

3.3 The Branching Rule

The branching rule is the criterion used at each node N(S) to proceed the search in one of the possible $(2^{n(S)-1} - 1)$ branches where each branch represents a set of tests which could be used to change the state of ambiguity at this node to another state in another level down the tree. The more effective the branching rules are, the faster a feasible solution could be reached and consequently the less the time and speed required.

The branching rule used in the branch and bound algorithm was proposed by Johnson, et al. [9] as a method for constructing a good but not necessarily optimum sequence of tests that can be executed by an automatic diagnostic. Using this rule will improve the efficiency of the branch and bound algorithm because it will guarantee finding a good feasible solution as fast as possible.

This rule uses the information-gain figure-of-merit, F_k, which is the ratio of the ambiguity removed by a test T_k to the test cost, $\bar{C}_k$. This rule is defined as follows:

At any node $N(\hat{S})$ with a state having $n(\hat{S})$ untested LRUs, by applying test T_k, which has a cost $\bar{C}_k$, either state $S(\hat{S},T_k)$ of node $N(S)$ could be reached if the test passes, or state $\bar{S}(\hat{S},\bar{T}_k)$ of node $N(\bar{S})$ will be reached if it fails. Then,

$$F_k = -[\rho \log_2 \rho + (1-\rho) \log_2 (1-\rho)]/\bar{C}_k \tag{3.6}$$

where $\rho = \sum_{j\varepsilon I(S)} p_j / \sum_{j\varepsilon I(\hat{S})} p_j$

Rank all tests at node $N(\hat{S})$ in a decreasing order according to the values of their F. According to this order the tests will be chosen at this node.

3.4 The Dominance Rules

Dominance rules could play a very important part in determining the size of the solution space and consequently the size of the search tree, especially if it works at the root of the tree. Therefore, attention should be made in order to come up with strong dominance rules.

At any node $N(\hat{S})$ by applying test T_k two nodes could be reached; either node of state $S(\hat{S},T_k)$ or node of state $\bar{S}(\hat{S},\bar{T}_k)$ with $n(S)$ and $n(\bar{S})$ remaining untested LRUs respectively. Divide the set of all possible tests $T(\hat{S})$ at node $N(\hat{S})$ into two subsets τ, and $\bar{\tau}$ where;

$\tau = [T_k | T_k \ \varepsilon \ T(\hat{S}),\ n(S) \neq n(\bar{S})]$

and

$\bar{\tau} = [T_k | T_k \ \varepsilon \ T(\hat{S}),\ n(S) = n(\bar{S})]$

Assume further that state S* is the state with the minimum number of remaining untested LRUs among states S and $\bar{S}$.

$$\text{Let} \quad \rho(T_k) = \begin{cases} \sum\limits_{j \varepsilon I(S^*)} p_j & ,\text{if } n(S) \neq n(\bar{S}) \\ \min[\sum\limits_{j \varepsilon I(S)} p_j,\ \sum\limits_{j \varepsilon I(\bar{S})} p_j] & ,\text{if } n(S) = n(\bar{S}) \end{cases}$$

The following theorems explain the dominance rules which will be used in the algorithm. The detailed structural proofs of all these theorems are presented in Appendix A.

Theorem 3.4.1

At any node N(S), any branch generated by a test T_k such that $T_k \ \varepsilon \ \tau$ dominates any other branch generated by a test T_m such that $T_m \ \varepsilon \ \tau$ if:

$$\bar{C}_k = \min_{i \varepsilon T(S)} [\bar{C}_i]$$

and $\qquad \bar{C}_k \leq \bar{C}_m \cdot \rho(T_k)$

Theorem 3.4.2

At any node N(S), any branch generated by test T_k such that $T_k \ \varepsilon \ \tau$ dominates any other branch generated by test T_m such that $T_m \ \varepsilon \ \bar{\tau}$ if

$$\bar{C}_k = \min_{i\varepsilon T(S)} [\bar{C}_i]$$

and $\bar{C}_k \leq \bar{C}_m \cdot \rho(T_k)$

Corollary 3.4.1

Theorem 3.4.2 could also be applied in the opposite case, i.e., any branch generated by test T_k such that $T_k \varepsilon \bar{\tau}$ dominates any other branch generated by a test T_m such that $T_m \varepsilon \tau$

if $\bar{C}_k = \min_{i\varepsilon T(S)} [\bar{C}_j\}$

and $\bar{C}_k \leq \bar{C}_m \cdot \rho[T_k]$

Theorem 3.4.3

At any node N(S) with a state having at most four remaining untested LRUs, if test T_k such that $T_k \varepsilon \bar{\tau}$ has the minimum cost among all tests which can be used at N(S), then the branch generated by T_k dominates all branches which are generated by any other test T_m such that $T_m \varepsilon \bar{\tau}$.

A summary of the dominance rules is presented in Table 3.1 which summarizes the condition required to make a branch generated by test T_k at node N(S) dominates another branch generated by test T_m, where $\bar{C}_k = \min_{i\varepsilon T(S)} (\bar{C}_i)$.

TABLE 3.1

DOMINANCE RULES

T_k \ T_m	$T_m \varepsilon \tau$	$T_m \varepsilon \bar{\tau}$
$T_k \varepsilon \tau$	$\bar{C}_k \leq \bar{C}_m \cdot \rho(T_k)$	$\bar{C}_k \leq \bar{C}_m \cdot \rho(T_k)$
$T_k \varepsilon \bar{\tau}$	$\bar{C}_k \leq \bar{C}_m \cdot \rho(T_k)$	If $n(S) \leq 4$ no other condition if required If $n(S) > 4$ no general rule is founded

3.5 The Branch and Bound Algorithm

In this section the complete branch and bound algorithm for determining the sequence of diagnostic tests to be executed automatically by the BIT to isolate the group of modules (LRUs) which contains the faulty unit is given.

The input parameters are:

n = Total number of LRUs

T = Set of all tests which could be used

p_i = Prior probability of failure of LRU_i, $i = 1,2,\ldots,n$

E_i = Expected cost for secondary isolation of the failed unit in LRU_i

$\bar{C}_k$ = Cost associated with test T_k

Values of the objective function are:

UB $=$ Upper bound on expected total cost

$L(S)$ $=$ Lower bound on expected total cost at state S

C $=$ Expected cost of a feasible test sequence

The parameters for creating, fathoming nodes and branching are:

ND $=$ Current node number

n^* $=$ Counter for nodes created

$S(\hat{S},T_k)$ $=$ State S generated by applying test T_k at previous state $\hat{S}$

$n(S)$ $=$ Number of the remaining untested LRUs at node S

$N(S)$ $=$ Node corresponding to state S

$T(S)$ $=$ Set of the $2^{n(S)-1} - 1$ possible tests at state S

$I(S)$ $=$ Set of the $n(S)$ remaining untested LRUs at state S

$C(S)$ $=$ Lower bound of the minimum expected cost of tests required to remove the ambiguity of the $n(S)$ untested LRUs at state S

$Y(S)$ $=$ Set of the remaining feasible branches after applying the *dominance* rules at node $N(S)$ (each branch could be generated by at least one test).

$\ell(S)$ $=$ Level of node $N(S)$

Step 0 Initialize the input parameter, let S be the initial state of node $N(S)$, $n(S) = n$, $\ell(S) = 1$, $ND = 1$, and $n^* = 1$. Compute UB using equation 3.1 and $L(S)$ using equation 3.3.

Step 1 Apply a stopping test based on the secondary isolation costs. If $C(S) + \sum_{i=1}^{n} p_i \cdot E_i \geq \sum_{i=1}^{n} E_i$. Use the secondary isolation for all LRUs, stop. Otherwise, go to 2.

Step 2 Use the dominance rules to find Y(S).

Step 3 Find the information-gain figure-of-merit F_k for each branch or test $T_k \varepsilon Y(S)$ using equation 3.6. Rank them in a decreasing order according to the values of their F.

Step 4 Start branching using branch of test T_k with the maximum F among all tests in Y(S) and remove this branch (test) from Y(S).

Step 5 Generate the new two possible nodes by using T_k at node S, denote them $N(S_1)$ and $N(S_2)$. Find $n(S_1)$ and $n(S_2)$.

Step 6 If min $[n(S_1), n(S_2)] = 1$, let node number $n^* + 1$ be the node with max $[n(S_1), n(S_2)]$, go to 7. Otherwise, let node number $n^* + 1$ be a fictitious node, go to 8.

Step 7 Let state S be the state of node number $n^* + 1$, let $ND = n^* + 1$, $\ell(S) = \ell(S) + 1$, go to 11.

Step 8 Let state S be the state of node number $n^* + 1$, let $ND = n^* + 1$, $\ell(S) = \ell(S) + 1$.

Find L(S) of the fictitious node N(S) using equation 3.4.

Step 9 If $n(S_2) > n(S_1)$, let node number ND + 1 be $N(S_1)$, and node number ND + 2 be $N(S_2)$. Otherwise, let node number ND + 1 be $N(S_2)$ and node number ND + 2 be $N(S_1)$.

Step 10 $\ell(S_1) = \ell(S) + 1$ and $\ell(S_2) = \ell(S) + 1$, let S be the state of node number ND + 2. Let n* = ND + 2.

Step 11 Compute L(S) using equation 3.2.

Step 12 Apply the secondary isolation stopping test. If $C(S) + \sum_{i \varepsilon I(S)} P_i \cdot E_i \geq \sum_{i \varepsilon I(S)} E_i$. Go to 15. Otherwise, go to 13.

Step 13 If L(S) $\geq$ UB, fathom node N(S). Go to 21. Otherwise, generate T(S), go to 14.

Step 14 If n(S) = 2, compute C using equation 3.5, go to 16. Otherwise, go to 2.

Step 15 Compute $C = L(S) = C(S) - \sum_{i \varepsilon I(S)} P_i \cdot E_i + \sum_{i \varepsilon I(S)} E_i$, Y(S) is empty.

Step 16 If C $\geq$ UB, fathom node N(S), go to 21. Otherwise, go to 17.

Step 17 If $\ell(S) = 2$, the last solution is feasible. Let UB = C, and state S be the state of node number n*, go to 22. Otherwise, go to 18.

Step 18 If node N(S) is branched directly from a fictitious node, go to 19. Otherwise, go upward the same branch to the next node, let S be the state of this node, with number ND.

Step 19 If node number ND-1 is fictitious, let S be its state and let ND = ND-1, go to 17. Otherwise, let S be the state of node number ND-2 and let ND = ND-2, go to 20.

Step 20 Update the lower bound at the fictitious node N(S) by substituting the last value of C instead of $C(\bar{S})$ in equation 3.4. Let S be state of node number ND + 1, let ND = ND + 1. Compute L(S), go to 12.

Step 21 Let ND be the number of the node N(S). If N(S) is branched from a fictitious node, fathom also node number ND-1, and let S be the state of node number (ND-2) and let its number be ND. Otherwise, let S be the state of node number (ND-1) and let its number be ND.

Step 22 If $\ell(S) = 1$, go to 30. Otherwise go to 23.

Step 23 If $\ell(S) = 2$, go to 28. Otherwise go to 24.

Step 24 If N(S) is fictitious, or $n(S) = 2$, let state S be the state of node number ND-1, and its number is ND, go to 22. Otherwise go to 25.

Step 25 If Y(S) is empty, go to 26. Otherwise go to 31.

Step 26 If N(S) is branched from a fictitious node, go to 27. Otherwise, let state S be the state of node number ND-1 and let its number be ND, go to 22.

Step 27 If the lower bound at the fictitious node has been previously updated, let state S be state of node number ND-1 and let its number be ND, go to 22. Otherwise, go upward this branch to the next node let its state be S and its number ND, go to 22.

Step 28 If N(S) is fictitious, or n(S) = 2. Let S be the initial state, go to 3-. Otherwise go to 29.

Step 29 If Y(S) is empty. Let S be the initial state, go to 30. Otherwise go to 31.

Step 30 If Y(S) is empty, stop, go to 32. Otherwise go to 31.

Step 31 Let n* be the number of node N(S), go to 4.

Step 32 The optimal sequence of tests is the one associated with the last value of the upper bound UB.

3.6 Verification of the Algorithm

The algorithm of Section 3.5 was coded in FORTRAN IV. The code was verified using the example problem used in [11] and shown in Table 1.1.

The search tree used in solving this problem by branch and bound algorithm is presented in Figure 3.3. The same optimal sequence of tests has been obtained. Either sequence of tests T_{1000}, T_{1100}, and T_{0010} or T_{1000}, T_{0010}, and T_{1100} produced the same optimal solution.

From the tree presented in Figure 3.3 it is noticed that the dominance rules and lower bounds worked efficiently to reduce the size of the tree to include only seven nodes compared with the original possible tree for four LRUs, which has 23 nodes as well as the dynamic programming network which has 16 nodes.

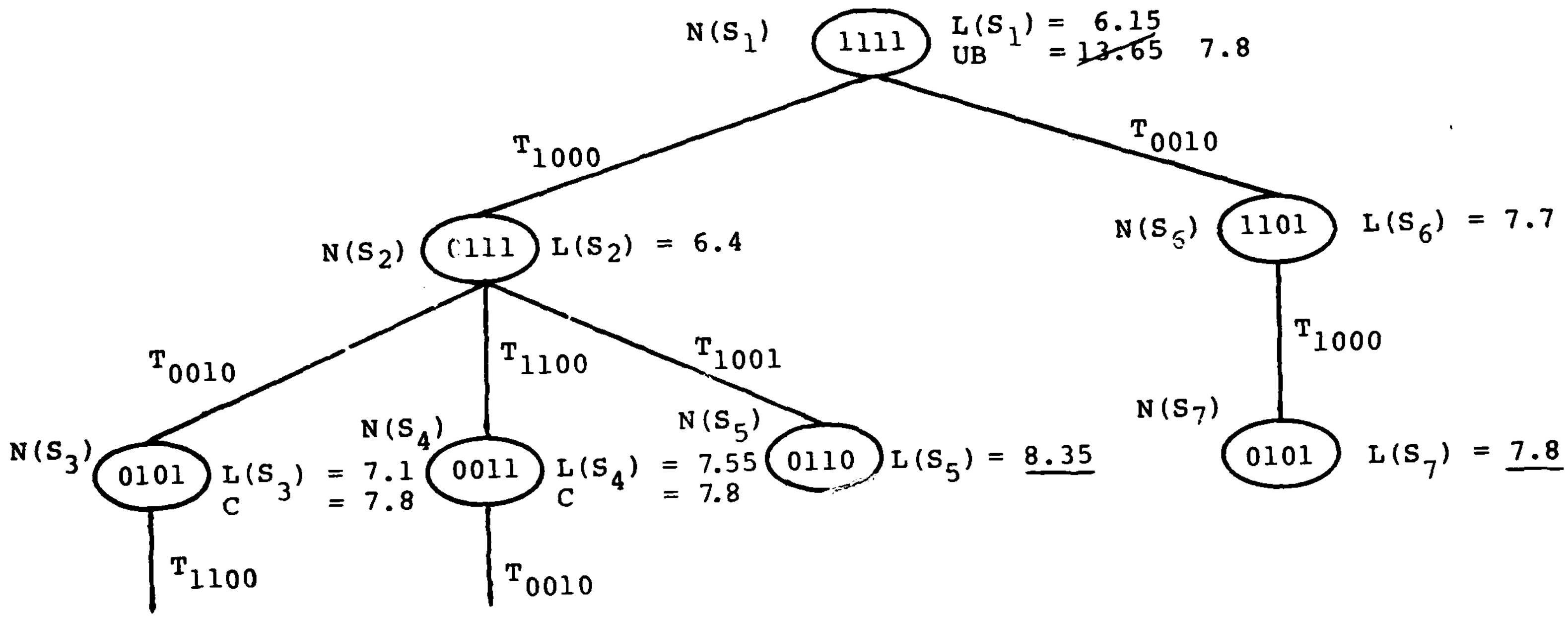


Figure 3.3 Search Tree of the Example Problem

It is noticed, also that even though the total number of nodes generated during the execution of the algorithm was seven, the maximum number of nodes stored at any time was only three, which is relatively small and reasonable. Also, the first feasible solution happened to be the optimal solution which shows the strength of the branching rules and its effectiveness in helping fathoming the remaining active nodes.

Thus, in a simple test example, the efficiency of the algorithm was verified.

3.7 The Heuristic Algorithm

The branch and bound algorithm explained in section 3.5 finds efficiently the optimal solution. However, the size of the problems which could be solved by this algorithm is relatively small because of the storage burden and time requirement, which is inherented in most combinatorial problems.

This heuristic algorithm is simply the same branch and bound algorithm explained in the previous section with two more stopping tests which stop the search for optimality by stopping the search either directly after finding the second feasible solution or after generating a limited number of nodes based on the maximum number of nodes required to find a feasible solution. By experiment it was found that the best results happened when the search stopped after generating a number of nodes equals to fifteen times the maximum number of nodes required to find a feasible solution. These tests were developed from the computational results of the branch and bound algorithm which showed that most of the search time was consumed in proving optimality not in finding the optimal solution itself.

The stopping test based on the second feasible solution can be added in step 17 in the branch and bound algorithm. While the stopping test based on the total number of nodes could be added before step 11.

The value of the objective function obtained by the heuristic algorithm was found to be on the average, 99.244% or more of the values of the optimal solution for all test problems. The details of the computational experience are presented in Chapter 4.

CHAPTER 4

COMPUTATIONAL RESULTS

In this chapter the computational experience with both the branch and bound and heuristic algorithms presented in Chapter 3 is demonstrated and analyzed. The test problems were randomly generated from uniform distribution. All probabilities of failure of the n LRUs were generated from a uniform (0-1) distribution. The costs of all tests were generated from a uniform (1-20), while the expected costs for secondary isolation of all LRUs were generated from a uniform (1-10) distribution. All problems were run on the University of Oklahoma IBM 370/158J computer. The results are summarized in Table 4.1.

As in all combinatorial problems, the required computational time is a function of the size of the problem as well as the number of active nodes. As depicted in Figure 4.1 the case of $n \geq 8$ LRUs is the critical case where the time starts increasing exponentionally from 10.6 seconds in case of $n = 7$ to 160.345 seconds in case of $n = 8$.

Table 4.2 displays a comparison between the branch and bound algorithm and the heuristic one. The savings in computation time by using the heuristic algorithm is obvious, especially when the number of LRUs increases. However, the

TABLE 4.1

COMPUTATIONAL RESULTS FOR ALL TEST PROBLEMS

No. of LRUs	No. of Problems	Average CPU Time Sec.	Average no. of Nodes Created	Max. no. of Nodes Created	Max no. of Active Nodes	Average Optimal Node
3	50	.2786	2.16	4	2	1.74
4	50	.3316	10.48	26	4	4.96
5	50	.5264	32.2	113	5	9.12
6	50	1.7656	89.8	432	7	15.06
7	50	10.6	213.66	666	8	24.32
8	20	160.345	968	2812	10	50.4
9	2	1253.715	1947	1959	11	13

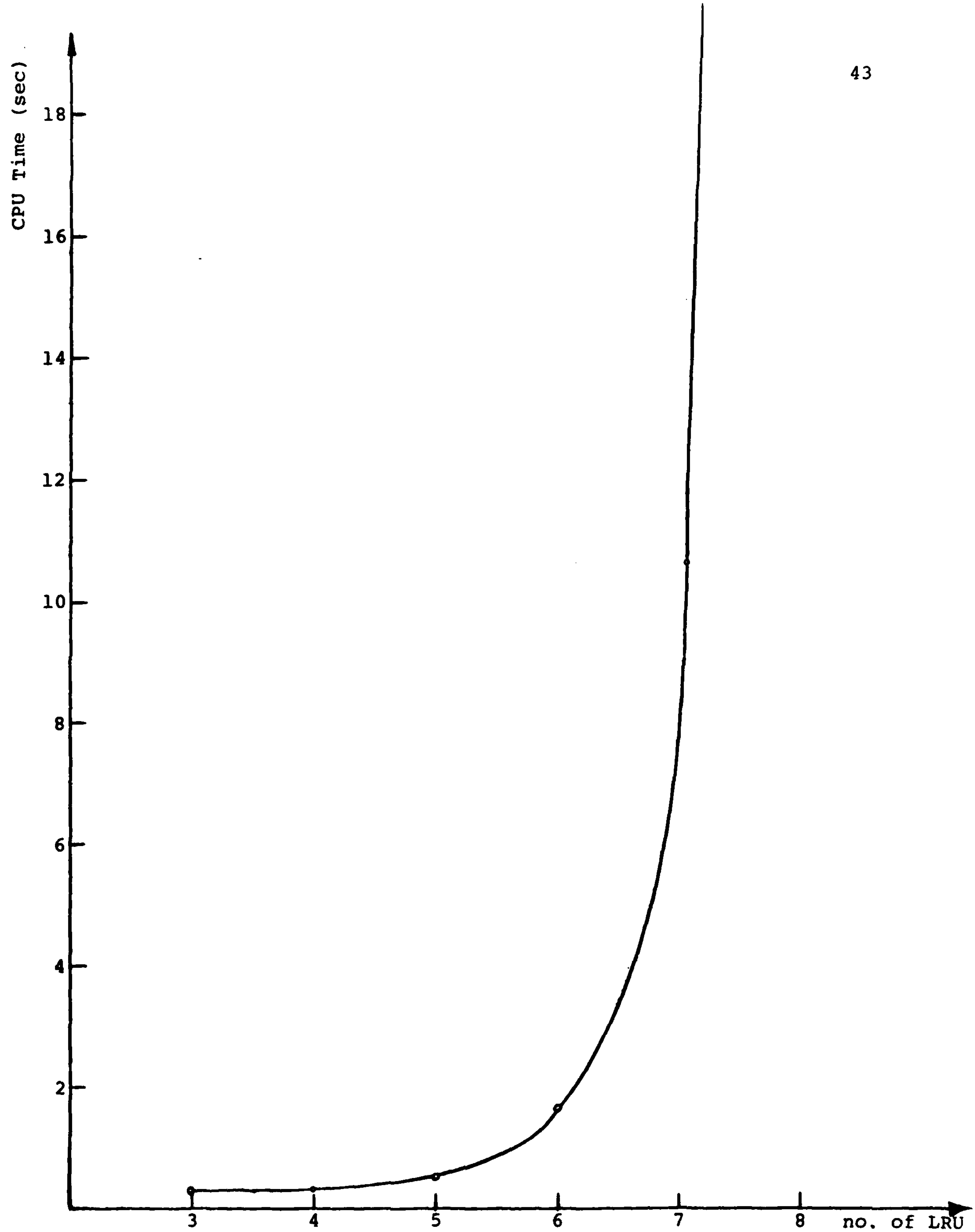


Figure 4.1. A plot of computational times for selected problems

TABLE 4.2

A COMPARISON BETWEEN THE RESULTS OF BRANCH AND BOUND AND THE HEURISTIC ALGORITHMS

LRU	Average CPU Time Sec		%ge of time saved by using heuristic	%ge of difference in the objective function between the optimal and heuristic	
	B&B	Heuristic		Average	Maximum
4	.331	.318	3.86%	.4418%	3.91%
5	.526	.426	18.95%	.756%	7.25%
6	1.765	1.032	42.17%	.349%	3.37%
7	10.6	6.113	42.45%	.666%	3.4%
8	160.345	23.579	85.3%	.0124%	.0749%
9	1253.715	74.137	94.00%	0	0
10	>3600	280.135	>92.2%	optimal solution is not known	

sacrifice in the optimal value of the objective function is less than 7.25% of the optimal, and on the average it is less than 0.756%. Also, Figure 4.2 shows that the heuristic algorithm reached the optimal solution in more than 82% of the problems tested which shows the effectiveness of this algorithm.

Table 4.3 shows a comparison between the branch and bound algorithm and the dynamic programming approach used in [11]. This comparison is based on the maximum number of nodes created by

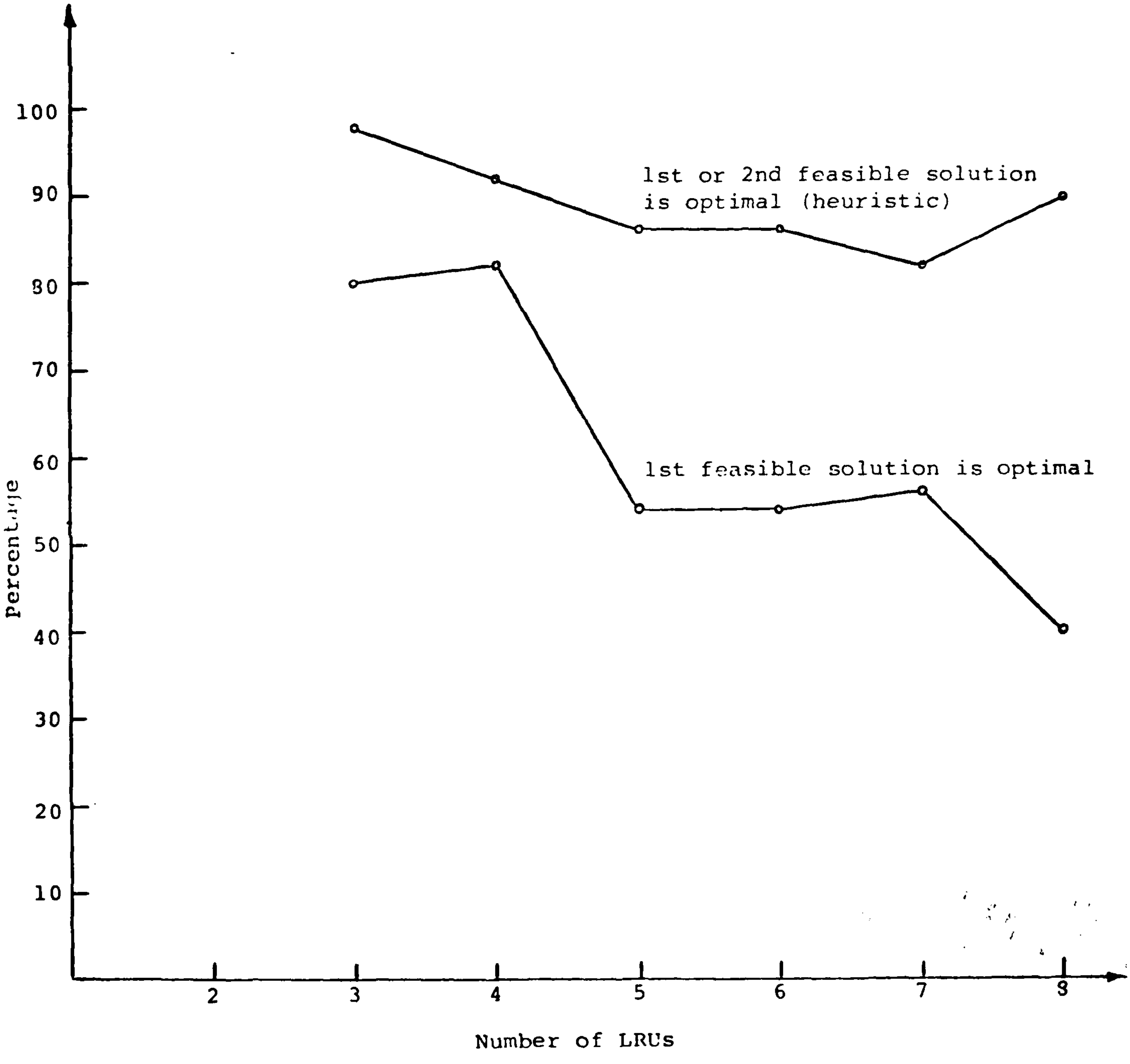


Figure 4.2. A plot of the percentage of time the optimal solution was reached under two conditions

TABLE 4.3

A COMPARISON BETWEEN THE BRANCH AND BOUND AND DYNAMIC PROGRAMMING ALGORITHMS

LRU / Max. no. of nodes created	3	4	5	6	7	8	9
Branch and Bound	4	26	113	432	666	2812	1959
Dynamic Programming	12	77	39	1767	7560	31369	128010

using both algorithms. The comparison indicated a dramatic difference in the number of nodes created especially for $n \geq 7$ LRUs. A comparison in the computation time would have been rather more important. However, no computational results were reported in case of using dynamic programming, only the upper bound on the number of states generated by dynamic programming.

CHAPTER 5

SUMMARY AND CONCLUSIONS

5.1 Summary

Two approaches to the cost effective design of fault isolation procedures were investigated. The problem was formulated as a search tree in which the optimal search procedure could be found using a branch and bound approach. Dominance and branching rules were developed, then a branch and bound algorithm was presented.

Having studied the computational results, another heuristic algorithm was developed which proved to be efficient and fast. An example problem was solved to illustrate the efficiency of the branch and bound algorithm and was compared with a previous dynamic programming algorithm.

Computational results indicated that the heuristic algorithm was faster than the branch and bound one with a very slim sacrifice in optimality.

Computational results were reported and compared to the available results of other algorithms.

5.2 Conclusions

Several conclusions can be drawn from this research regarding the consideration of new approaches for fault isolation problems. They are:

1. The branch and bound approach could be used successfully to tackle the problem of designing a cost effective fault isolation procedure. Because of the branching and dominance rules, many of the nonoptimal solutions would be eliminated early in the solution procedure which could efficiently reduce the size of the required search tree, as well as the time and storage needed to find the optimal solution.

2. The branch and bound algorithm proved to be more efficient than the dynamic programming scheme which has been used in previous works to seek optimal procedures.

3. The heuristic algorithm presented in section 3.7 proved to be a good compromise between the ultimate goal of optimality and the problem of time requirement to achieve this goal. This algorithm has the advantage of finding a near optimal solution in a very short time compared to other methods.

4. Even though the size of problems solved efficiently by the two algorithms are limited to nine LRUs, this size is still greater than any problem reported to be solved in any previous work.

5.3 Future Work

Recommendations for further research in the cost effective design of fault isolation procedures would be:

1. Developing a technique to minimize and control the number of possible tests which could be used in the search because of the dramatic increase of the possible number of tests with the increase of LRUs.

2. More investigation in developing branching and dominance rules and more work in designing test procedures using branch and bound approaches.

3. Investigating how to partition the equipment into optimum groups of modules.

4. Considering the problem without neglecting the possibility of multiple failures of two or more LRUs at the same time.

5. Studying the effect of imperfect information on the optimum test procedures and how to modify the solution according to that (sensitivity analysis).

6. Determining an optimum procedure which minimizes the expected cost of secondary isolation to locate the single failed unit within the group of LRUs identified by the BIT primary diagnostic.

REFERENCES

1. Aly, A. A., "Optimum Design of Built-in-Test Diagnostic Systems," A Report submitted to AFOSR, 1979.
2. Butterworth, R., "Some Reliability Fault-Testing Models," Operations Research, Vol. 20 (1972), pp. 335-343.
3. Chang, H. Y., "A Distinguishability Criterion for Selecting Efficient Diagnostic Test," AFIPS Proceedings of Spring Joint Computer Conference, Vol. 32 (1968), pp. 529-534.
4. Cohn, H. Y. and Ott, G., "Design of Adaptive Procedures for Fault Detection and Isolation," IEEE Transactions Reliability, Vol. R-20 (1971), pp. 7-10.
5. Firstman, S. I. and Gluss, B., "Optimum Search Routines for Automatic Fault Isolation," Operations Research, Vol. 8 (1960), pp. 512-523.
6. Glassey, C. R., "Minimum Change Over Scheduling of Several Products on One Machine," Operations Research, Vol. 16 (1968), pp. 342-352.
7. Gluss, B., "An Optimum Policy for Detecting a Fault in a Complex System," Operation Research, Vol. 7 (1959), pp. 467-477.
8. Halpern, J., "Fault Testing for a k-out-of-n System," Operation Research, Vol. 22 (1974), pp. 1267-1271.

9. Johnson, R. A., Kletsky, E. J. and Brule, J. D., "Diagnosis of Equipment Failures," SURI, Report No. EE, 557-594T1 (1959).

10. Lawler, E. L. and Wood, D. E., "Branch and Bound Methods: A Survey," Operations Research, Vol. 14 (1966), pp. 699-717.

11. Sheskin, T. J., "Sequencing of Diagnostic Tests for Fault Isolation by Dynamic Programming," IEEE Transactions on Reliability, Vol. R-27 (1978), pp. 353-358.

APPENDIX A

DOMINANCE RULES THEOREMS

The proofs of all theorems which have been used to determine the dominance rules in Section 3.4 are presented here.

In order to simplify the proofs, the cost $\bar{C}_k$ of a test T_k which has 1's in positions i,j,...,z will also be identified as $C_{i,j,\ldots,z}$.

Theorem 3.4.1

At any node N(S), any branch generated by a test T_k such that $T_k \in \tau$ dominates any other branch generated by a test T_m such that $T_m \in \tau$ if:

$$\bar{C}_k = \min_{i \in T(S)} [\bar{C}_i]$$

and $$\bar{C}_k \leq \bar{C}_m \cdot \rho(T_k)$$

Proof

Referring to Figure A.1 and Figure A.2 which represent two branches from the search tree of a problem of 5 LRUs, both

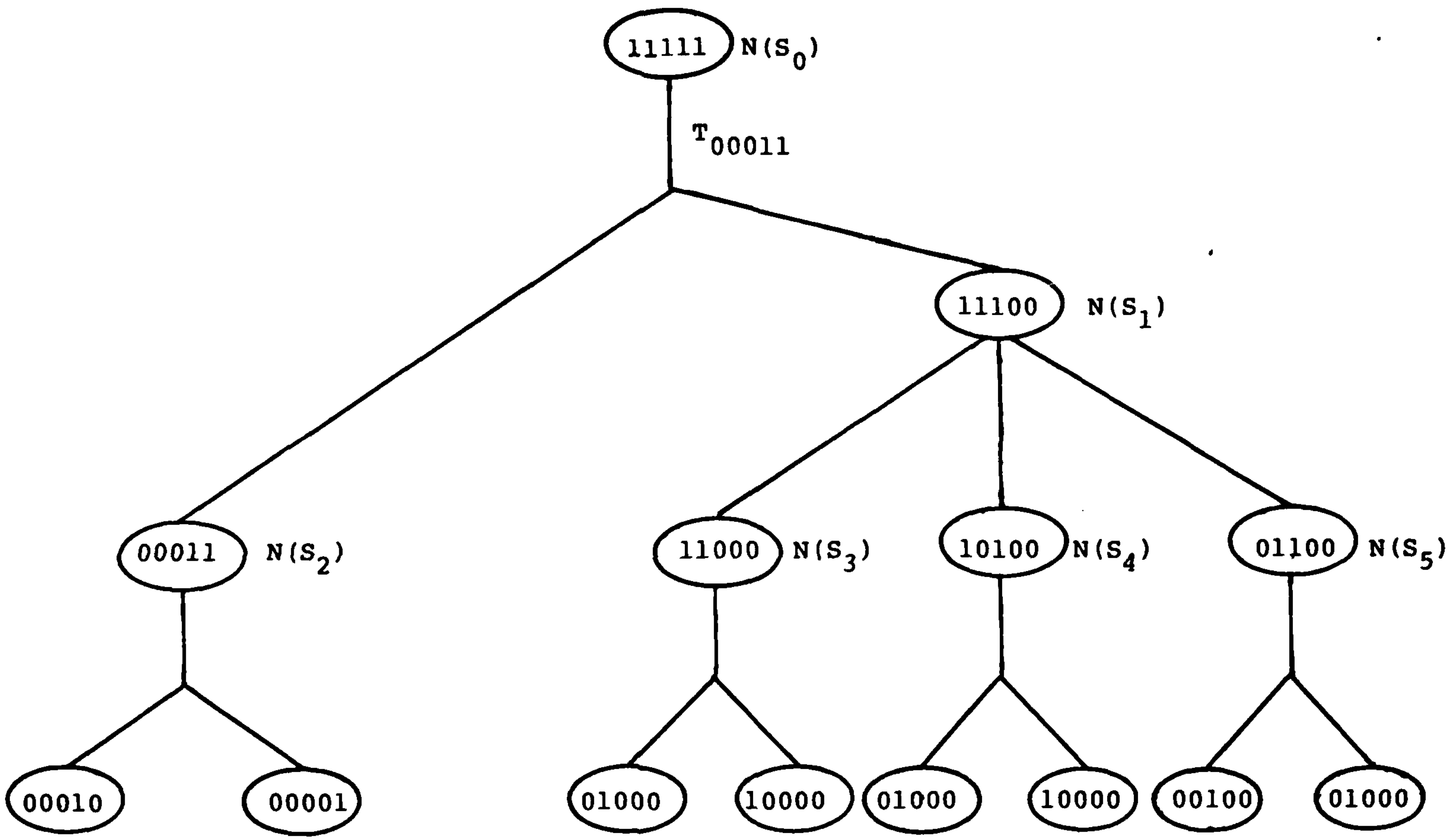


Figure A.1 A Search Tree for a 5-LRUs Example

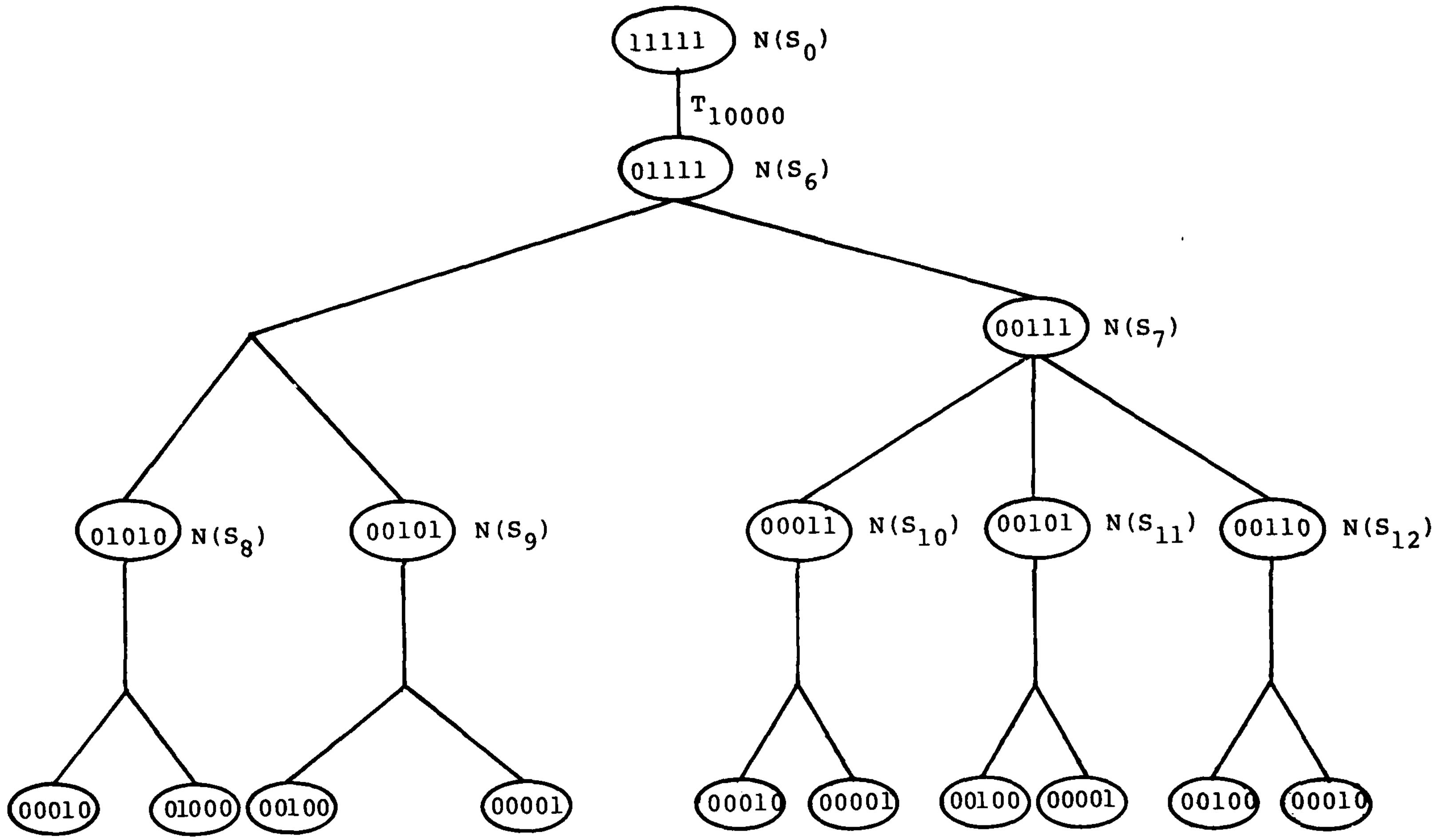


Figure A.2 A Search Tree for a 5-LRUs Example

branches are emanated from the first node and by using tests belong to set τ. All tests which could be used at any branch are presented in Table A.1.

Let the cost of branch $b_1(S_0,S_1,S_5)$ which pass through nodes $N(S_0)$, $N(S_1)$ and $N(S_5)$ be $C_{b_1}(S_0,S_1,S_5)$, and let the cost of branch $b_2(S_0,S_6,S_7,S_{10})$ which pass through nodes $N(S_0)$, $N(S_6)$, $N(S_7)$, $N(S_{10})$ be $C_{b_2}(S_0,S_6,S_7,S_{10})$, then,

$$\begin{aligned} C_{b_1}(S_0,S_1,S_5) = {} & C_{4,5}+(p_4+p_5)\cdot\min[C_5,C_4,C_{1,4},C_{3,4},C_{1,5},C_{2,4},\\ & C_{2,5},C_{3,5}]+(p_1+p_2+p_3)\cdot\min[C_1,C_{1,4},C_{2,3},C_{1,5}]+\\ & (p_2+p_3)\cdot\min[C_{1,2},C_2,C_3,C_{2,4},C_{2,5},C_{1,3},C_{3,4},\\ & C_{3,5}] + \sum_{i=1}^{5} p_i\cdot E_i. \end{aligned}$$

and

$$\begin{aligned} C_{b_2}(S_0,S_6,S_7,S_{10}) = {} & C_1+(p_2+p_3+p_4+p_5)\cdot\min[C_{1,2},C_2] + (p_3+p_4+p_5)\\ & \cdot\min[C_{2,3},C_{1,3},C_3,C_{4,5}]+(p_4+p_5)\cdot\min[C_5,\\ & C_4,C_{1,4},C_{3,4},C_{1,5},C_{2,4},C_{2,5},C_{3,5}]+\sum_{i=1}^{5} p_i\cdot E_i. \end{aligned}$$

Branch $b_1(S_0,S_1,S_5)$ generated by test T_{00011} dominates branch $b_2(S_0,S_6,S_7,S_{10})$ generated by test T_{10000} if

$$C_{4,5} + (p_1+p_2+p_3)\cdot\min[C_1,\ldots] + (p_2+p_3)\cdot\min[C_{1,2},C_2,\ldots]$$
$$\le C_1 + (p_2+p_3+p_4+p_5)\cdot\min[C_{1,2},C_2] + (p_3+p_4+p_5)\cdot\min[C_{4,5},\ldots]$$

But if $C_{4,5} = \min\limits_{i\varepsilon T(S_0)} [\bar{C}_i]$

and since $\sum_{i=1}^{5} p_i = 1$, therefore ...

TABLE A.1

TESTS WHICH COULD BE USED TO REACH STATE S FROM STATE $\hat{S}$ IN THE 5-LRUs EXAMPLE

$\hat{S}$	S	T ($\hat{S}$, S)
11100	11000	T_{00100}, T_{11000}, T_{00110}, T_{00101}
11100	10100	T_{01000}, T_{10100}, T_{01001}, T_{01010}
11100	01100	T_{10000}, T_{10010}, T_{01100}, T_{10001}
01111	00111	T_{11000}, T_{01000}
01111	(01010, 00101)	T_{01010}, T_{00101}
01010	(00010, 01000)	T_{01000}, T_{00010}, T_{11000}, T_{10010}, T_{01100}, T_{00110}, T_{01001}, T_{00011}
00011	(00010, 00001)	T_{00001}, T_{00010}, T_{10010}, T_{00110}, T_{10001}, T_{01010}, T_{01001}, T_{00101}
00101	(00100, 00001)	T_{00100}, T_{00001}, T_{10100}, T_{10001}, T_{01100}, T_{01001}, T_{00110}, T_{00011}
00110	(00100, 00010)	T_{00100}, T_{00010}, T_{10100}, T_{10010}, T_{01100}, T_{01010}, T_{00101}, T_{00011}
11000	(10000, 01000)	T_{10000}, T_{01000}, T_{10100}, T_{10010}, T_{10001}, T_{01100}, T_{01010}, T_{01001}
10100	(10000, 00100)	T_{10000}, T_{00100}, T_{11000}, T_{10010}, T_{10001}, T_{01100}, T_{00110}, T_{00101}
01100	(01000, 00100)	T_{01000}, T_{00100}, T_{11000}, T_{01010}, T_{01001}, T_{10100}, T_{00110}, T_{00101}

branch $b_1(S_0,S_1,S_5)$ dominates branch $b_2(S_0,S_6,S_7,S_{10})$ if

$- (p_4+p_5)C_1 \leq (p_4+p_5) \cdot \min[C_{1,2},C_2] - (p_1+p_2)C_{4,5}$

i.e. $(p_4+p_5)C_1 \geq (p_1+p_2)C_{4,5} - (p_4+p_5) \cdot \min[C_{1,2},C_2]$

which could be satisfied if $C_{4,5} \leq (p_4+p_5)C_1$.

This proof is valid even if the cost of branches $b_3(S_0,S_1,S_3)$ and $b_4(S_0,S_1,S_4)$ are less than that of $b_1(S_0,S_1,S_5)$ because in this case they dominate branch $b_1(S_0,S_1,S_5)$ and consequently dominate branch $b_2(S_0,S_6,S_7,S_{10})$. However, it is not the same for other branches $b_5(S_0,S_6,S_7,S_{12})$ and $b_2(S_0,S_6,S_7,S_{10})$. Therefore, it should be proved that branch $b_1(S_0,S_1,S_5)$ dominates branch $b_2(S_0,S_6,S_7,S_{10})$ and any other branch generated by test T_{10000}, whatever the branch emanating from node $N(S_6)$ with minimum cost is.

<u>Case 1</u> If branch $b_5(S_0,S_6,S_7,S_{12})$ is the optimal branch generated by test T_{10000}

Let the cost of branch $b_5(S_0,S_6,S_7,S_{12})$ be $C_{b_5}(S_0,S_6,S_7,S_{12})$

$$C_{b_5}(S_0,S_6,S_7,S_{12}) = C_1 + (p_2+p_3+p_4+p_5) \cdot \min[C_{1,2},C_2] + (p_3+p_4+p_5)\cdot\min[C_5,C_{2,5},C_{1,5},C_{3,4}] + (p_3+p_4)\cdot\min[C_3,C_4,C_{1,3},C_{1,4},C_{2,3},C_{2,4},C_{3,5},C_{4,5}] + \sum_{i=1}^{5} p_i \cdot E_i.$$

So, if $C_{4,5} \leq \min\limits_{i\varepsilon T(S_0^1)}[\bar{C}_i]$

then branch $b_1(S_0,S_1,S_5)$ dominates branch $b_5(S_0,S_6,S_7,S_{12})$ if

$C_{4,5} + (p_4+p_5) \cdot \min[C_5,C_4,C_{1,4},C_{3,4},C_{1,5},C_{2,4},C_{2,5},C_{3,5}] +$

$(p_1+p_2+p_3) \cdot \min[C_1, C_{1,4}, C_{2,3}, C_{1,5}] + (p_2+p_3) \cdot \min[C_{1,2}, C_2, C_3,$

$C_{2,4}, C_{2,5}, C_{1,3}, C_{3,4}, C_{3,5}] \leq C_1 + (p_2+p_3+p_4+p_5) \cdot \min[C_{1,2}, C_2]$

$+ (p_3+p_4+p_5) \cdot \min[C_5, C_{2,5}, C_{1,5}, C_{3,4}] + (p_3+p_4)\, C_{4,5}$

or $-(p_4+p_5)C_1 \leq -(p_1+p_2+p_5)C_{4,5} + (p_4+p_5) \cdot \min[C_{1,2}, C_2]$

$+ p_3 \cdot \min[C_5, \ldots]$

or $(p_4+p_5)C_1 \geq (p_1+p_2+p_5)C_{4,5} - (p_4+p_5) \cdot \min[C_{1,2}, \ldots]$

$- p_3 \cdot \min[C_5, \ldots]$

which could be satisfied if

$$C_{4,5} \leq (p_4+p_5)C_1$$

under the condition that

$$C_{4,5} = \min_{i\varepsilon T(S_0)}[\bar{C}_i]$$

By the same procedure it could be proved that branch $b_1(S_0, S_1, S_5)$ dominates branch $b_6(S_0, S_6, S_7, S_{11})$ and any other branch generated by test T_{10000} and a test T_k such that $k\varepsilon\tau$ at node S_6.

<u>Case 2</u> If branch $b_7(S_0, S_6, S_8, S_9)$ is the optimal branch generated by T_{10000}

Let the cost of branch $b_7(S_0, S_6, S_8, S_9)$ be $C_{b_7}(S_0, S_6, S_8, S_9)$

$$C_{b_7}(S_0, S_6, S_8, S_9) = C_1 + (p_2+p_3+P_4+p_5) \cdot \min[C_{2,4}, C_{3,5}] + (p_2+p_4) \cdot \min[C_{4,5}, \ldots] + (p_3+p_5) \cdot \min[C_{4,5}, \ldots]$$

So if $C_{4,5} = \min\limits_{i\varepsilon T(S_0)}[\bar{C}_i]$

$\therefore$ branch $b_1(S_0,S_1,S_5)$ dominates branch $b_7(S_0,S_6,S_8,S_9)$

if $-(p_4+p_5)C_1 \leq -p_1 C_{4,5}$

i.e. $(p_4+p_5)C_1 \geq p_1 C_{4,5}$

which could be satisfied if

$$C_{4,5} \leq (p_4+p_5)C_1$$

Therefore, in any event the branch generated by T_{00011} such that $T_{00011} \varepsilon \tau$ dominates any other branch which is generated by T_{10000} such that $T_{10000} \varepsilon \tau$ if

$$C_{4,5} = \min_{i\varepsilon T(S_0)} [\bar{C}_i]$$

and $C_{4,5} \leq C_1 \cdot \rho(T_{00011})$

Theorem 3.4.2

At any node N(S), any branch generated by test T_k such that $T_k \varepsilon \tau$ dominates any other branch generated by test T_m such that $T_m \varepsilon \bar{\tau}$ if

$$\bar{C}_k = \min_{i\varepsilon T(S)} [\bar{C}_i]$$

and $\bar{C}_k \leq \bar{C}_m \cdot \rho(T_k)$

Proof

Referring to Figures A.3 and A.4 which represent two branches in a search tree of problems having 6-LRUs, the first branch in Figure A.3 is generated by T_{100000} where $T_{100000} \varepsilon \tau$ and the second branch in Figure A.4 is generated by T_{111000} where $T_{111000} \varepsilon \bar{\tau}$. All tests which could be used at any branch

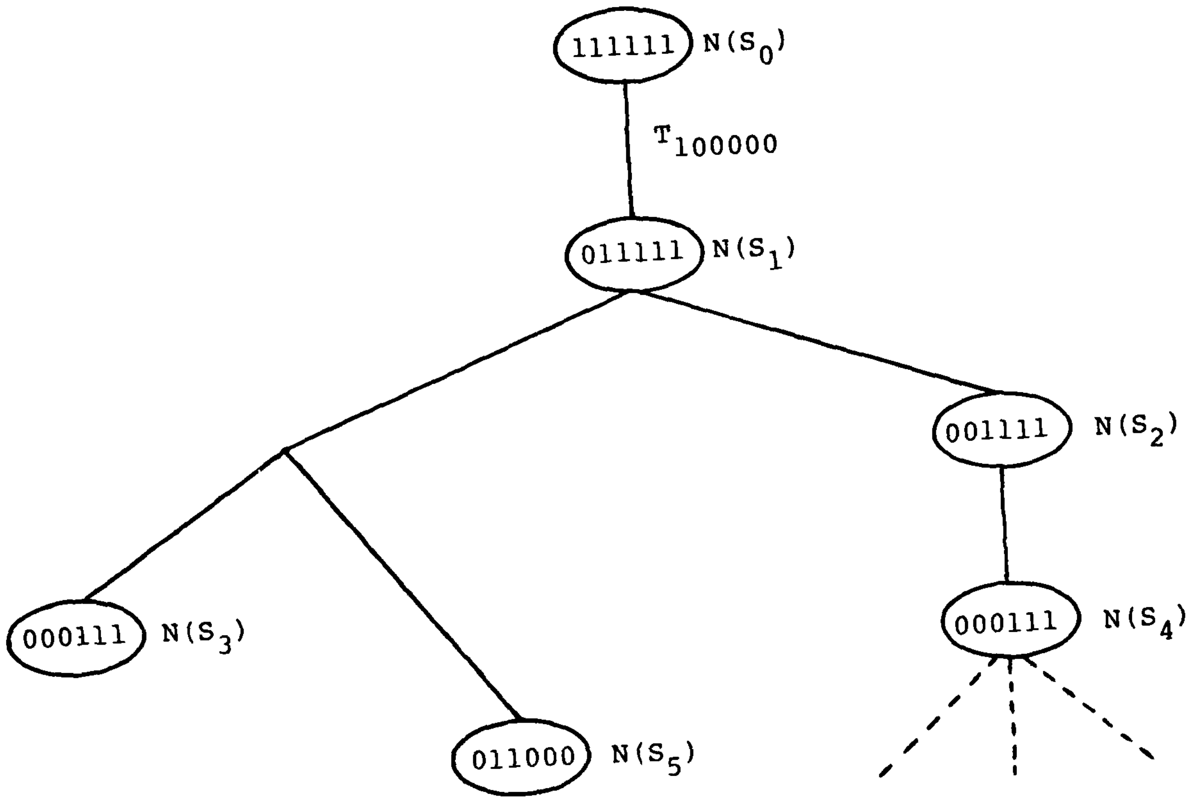


Figure A.3 A Search Tree for a 6-LRUs Example

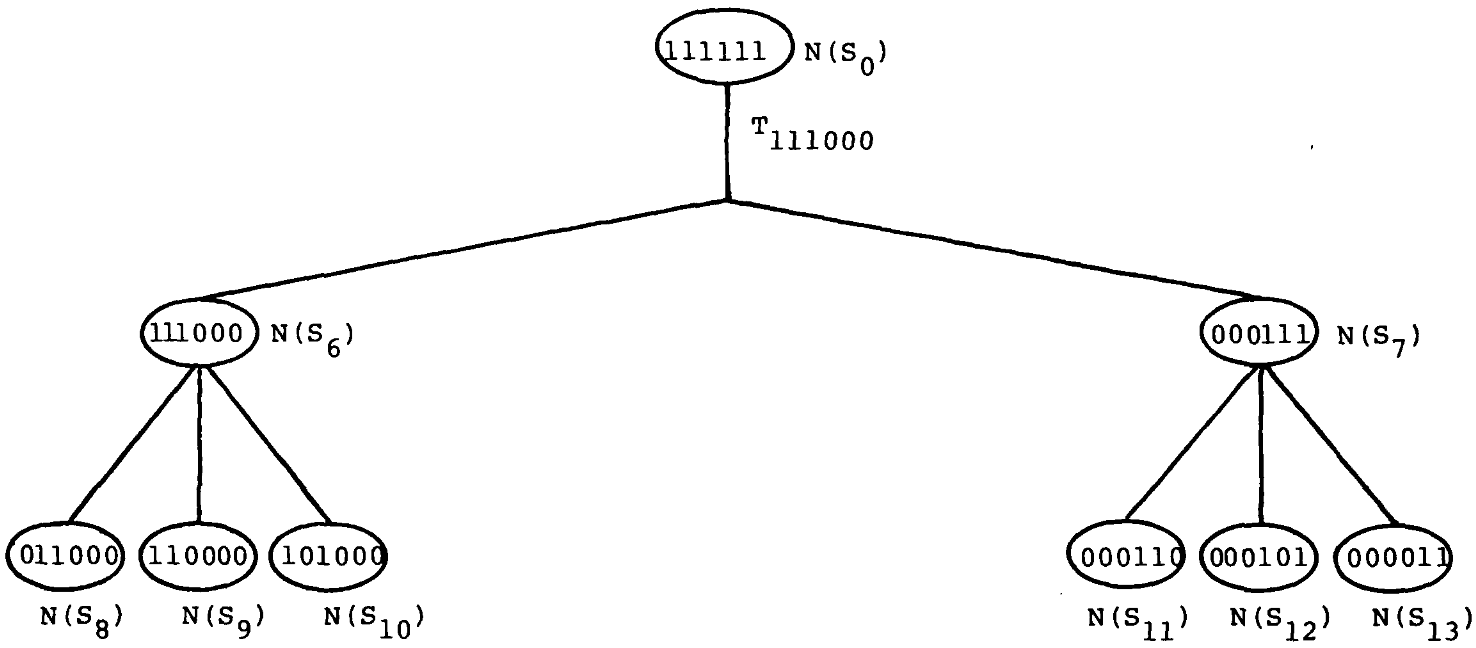


Figure A.4 A Search Tree for a 6-LRUs Example

are presented in Table A.2.

Let cost of branch $b_1(S_0,S_1,S_3,S_5)$ be $C_{b_1}(S_0,S_1,S_3,S_5)$

$$C_{b_1}(S_0,S_1,S_3,S_5) = C_1 + (p_2+p_3+p_4+p_5+p_6) \cdot \min[C_{2,3},C_{1,2,3}]$$
$$+ (p_2+p_3) \cdot \min[C_2,C_3,\ldots,C_{1,3,6}] + \text{cost of optimal search starting from node } N(S_3) + \sum_{i=1}^{6} p_i \cdot E_i.$$

Let cost of branch $b_2(S_0,S_6,S_8)$ be $C_{b_2}(S_0,S_6,S_8)$

$$C_{b_2}(S_0,S_6,S_8) = C_{1,2,3} + (p_1+p_2+p_3) \cdot \min[C_1, \ldots] + (p_2+p_3) \cdot$$
$$\min[C_2,C_3,\ldots,C_{1,3,6}] + \text{cost of optimal search starting from node } N(S_7) + \sum_{i=1}^{6} p_i \cdot E_i.$$

So if $C_1 = \min\limits_{i\varepsilon T(S_0)}[\bar{C}_i]$

then branch b_1 (S_0,S_1,S_3,S_5) which is generated by test T_{100000} where T_{100000} ε τ dominates branch b_2 (S_0,S_6,S_8) which is generated by test T_{111000} where T_{111000} ε $\bar{\tau}$ if

$$C_1 + (p_2+p_3+p_4+p_5+p_6) \cdot \min\ (C_{2,3},\ C_{1,2,3}) \leq C_{1,2,3} +$$
$$(p_1+p_2+p_3) \cdot \min\ [C_1,\ \ldots]$$

or if $C_1 + C_{1,2,3}\ (p_2+p_3+p_4+p_5+p_6) \leq C_{1,2,3} + (P_1+p_2+p_3)C_1$

$$-p_1\ C_{1,2,3} \leq -\ (p_4+p_5+p_6)C_1$$

$$\therefore\ p_1\ C_{1,2,3} \geq (p_4+p_5+p_6) \cdot C_1$$

which is still satisfied if

$$C_1 \leq p_1\ C_{1,2,3}$$

TABLE A.2

TESTS WHICH COULD BE USED TO REACH STATE S FROM STATE $\hat{S}$ IN THE 6-LRUs EXAMPLE

$\hat{S}$	S	$T(\hat{S}, S)$
000111	000110	T_{000001}, T_{100001}, T_{000110}, T_{010001}, T_{001001}, T_{110001}, T_{101001}, T_{100110}
000111	000101	T_{000010}, T_{000101}, T_{100010}, T_{010010}, T_{001010}, T_{110010}, T_{101010}, T_{100101}
000111	000011	T_{000100}, T_{000011}, T_{100100}, T_{010100}, T_{001100}, T_{110100}, T_{101100}, T_{100011}
111000	011000	T_{100000}, T_{100100}, T_{100010}, T_{100001}, T_{011000}, T_{100110}, T_{100101}, T_{100011}
111000	110000	T_{001000}, T_{110000}, T_{001100}, T_{001010}, T_{001001}, T_{110001}, T_{110100}, T_{110010}
111000	101000	T_{010000}, T_{101000}, T_{010100}, T_{010010}, T_{010001}, T_{101001}, T_{101100}, T_{101010}
011111	001111	T_{010000}, T_{110000}
011111	011000, 000111	T_{011000}, T_{111000}
001111	000111	T_{111000}, T_{001000}, T_{011000}, T_{101000}
000110	(000100, 000010)	T_{000100}, T_{000010}, T_{100010}, T_{010100}, T_{010010}, T_{001100}, T_{001010}, T_{000101}, T_{000011}, T_{100100}, T_{110100}, T_{110010}, T_{101100}, T_{101010}, T_{100101}, T_{100011}
000101	(000100, 000001)	T_{000100}, T_{000001}, T_{100100}, T_{100001}, T_{010100}, T_{010001}, T_{001100}, T_{001001}, T_{000110}, T_{000011}, T_{110100}, T_{110001}, T_{101100}, T_{101001}, T_{100110}, T_{100011}
011000	(010000, 001000)	T_{010000}, T_{001000}, T_{110000}, T_{101000}, T_{010100}, T_{010010}, T_{010001}, T_{001100}, T_{001010}, T_{001001}, T_{110100}, T_{110010}, T_{110001}, T_{101100}, T_{101010}, T_{101001}
110000	(100000, 010000)	T_{100000}, T_{010000}, T_{101000}, T_{100100}, T_{100010}, T_{100001}, T_{011000}, T_{010100}, T_{010010}, T_{010001}, T_{101100}, T_{101010}, T_{101001}, T_{100110}, T_{100101}, T_{100011}
101000	(100000, 001000)	T_{100000}, T_{001000}, T_{110000}, T_{100100}, T_{100010}, T_{100001}, T_{011000}, T_{001100}, T_{001010}, T_{001001}, T_{110100}, T_{110010}, T_{110001}, T_{100110}, T_{100101}, T_{100011}

This proof is valid whatever the optimal branch from node $N(S_1)$ is because if any other branch dominates branch $b_1(S_0,S_1,S_3,S_5)$, it will also dominate branch $b_2(S_0,S_6,S_8)$. However, to check the validity of the proof if any other branch from node $N(S_6)$ is optimal, let the cost of branch $b_3(S_0,S_6,S_9)$ be $C_{b_3}(S_0,S_6,S_9)$.

$$C_{b_3}(S_0,S_6,S_9) = C_{1,2,3} + (p_1+p_2+p_3) \cdot \min [C_3,C_{1,2},C_{3,4},C_{3,5},C_{3,6}, C_{1,2,6},C_{1,2,4},C_{1,2}]+(p_1+p_2)\cdot\min[C_1,\ldots]+ \text{cost of optimal search starting from node } N(S_7)+ \sum_{i=1}^{6} p_i \cdot E_i.$$

So if $C_1 = \min\limits_{i\varepsilon T(S_0)}[\bar{C}_i]$

Then branch $b_1(S_0,S_1,S_3,S_5)$ dominates branch $b_3(S_0, S_6,S_9)$ if $C_1 - p_1\, C_{1,2,3} + (p_2+p_3) \cdot \min [C_3,C_{1,2}, C_{3,4}, C_{3,5}, C_{3,6},C_{1,2,6},C_{1,2,4},C_{1,2,5}, \ldots] \leq (p_1+p_2)C_1 + (p_1+p_2+p_3) \cdot \min [C_3,C_{1,2},C_{3,4},C_{3,5},C_{3,6},C_{1,2,6},C_{1,2,4},C_{1,2,5}]$

or $- p_1\, C_{1,2,3} \leq - (p_3+p_4+_5+p_6)C_1 + p_1 \cdot \min [C_3, \ldots]$

or $p_1\, C_{1,2,3} \geq (p_3+p_4+p_5+p_6)C_1 - p_1 \cdot \min [C_3, \ldots]$

which could be satisfied if

$C_1 \leq p_1\, C_{1,2,3}$

By using the same procedure, it could be concluded that a branch generated by test T_{100000} where $T_{100000}\ \varepsilon\ \tau$ dominates any other branch which is generated by test T_{111000} where $T_{111000}\ \varepsilon\ \bar{\tau}$ if

$$C_1 = \min_{i\varepsilon T(S_0)} [\bar{C}_i]$$

and $$C_1 \leq C_{1,2,3} \cdot \rho(T_{100000}).$$

Corollary 3.4.1

Theorem 3.4.2 could also be applied in the opposite case, i.e., any branch generated by test T_k such that $T_k \varepsilon \bar{\tau}$ dominates any other branch generated by a test T_m such that $T_m \varepsilon \tau$

if $$\bar{C}_k = \min_{i\varepsilon T(S)} [\bar{C}_i]$$

and $$\bar{C}_k \leq \bar{C}_m \cdot \rho[T_k]$$

Proof

Referring to the proof of theorem 3.4.2, and Figures 3.5 and 3.6

if $$C_{1,2,3} \leq \min_{i\varepsilon T(S_0)} [\bar{C}_i]$$

Branch $b_2(S_0,S_6,S_8)$ which is generated by test T_{111000} where $T_{111000} \varepsilon \bar{\tau}$ dominates branch $b_1(S_0,S_1,S_3,S_5)$ which is generated by test T_{100000} where $T_{100000} \varepsilon \tau$ if

$$-(p_4+p_5+p_6)\, C_1 \leq -\, p_1 C_{1,2,3}$$

or $$(p_4+p_5+p_6)C_1 \geq p_1 C_{1,2,3}$$

which could be satisfied if $C_{1,2,3} \leq (p_4+p_5+p_6)C_1$.

This proof is valid whatever the optimal branch from node $N(S_6)$ is. However, to check the validity of the proof if any other branch from node $N(S_1)$ is optimal, let the cost of branch $b_4(S_0,S_1,S_2,S_4)$ be $C_{b_4}(S_0,S_1,S_2,S_4)$.

$$C_{b4}(S_0,S_1,S_2,S_4) = C_1 + \sum_{i=2}^{6} p_i \cdot \min[C_2,C_{1,2}] + \sum_{i=3}^{6} p_i \cdot \min[C_3, C_{1,3},C_{1,2,3},C_{2,3}] + [\text{cost of the optimal search starting from node having the state 000111}] + \sum_{i=1}^{6} p_i \cdot E_i.$$

So if $C_{1,2,3} = \min\limits_{i\varepsilon T(S_0)} [\bar{C}_i]$

Branch $b_2(S_0,S_6,S_8)$ dominates branch $b_4(S_0,S_1,S_2,S_4)$

if $C_{1,2,3}, + \sum_{i=1}^{3} p_i \cdot \min[C_1, \ldots] + (p_2+p_3) \cdot \min[C_2,C_{1,2},.]$

$$\leq C_1 + \sum_{i=2}^{6} p_i \cdot \min[C_2,C_{1,2}] + \sum_{i=3}^{6} p_i \cdot \min[C_{1,2,3}, \ldots]$$

or $-(p_4+p_5+p_6) \cdot C_1 \leq -(p_1+p_2+p_3)C_{1,2,3}$

$$(p_4+p_5+p_6)C_1 \geq (p_1+p_2+p_3)C_{1,2,3}$$

which could be satisfied if

$$C_{1,2,3} \leq (p_4+p_5+p_6)C_1$$

which is still satisfied if

$$C_{1,2,3} \leq \min[(p_1+p_2+p_3), (p_4+p_5+p_6)] \cdot C_1$$

Therefore, in any event a branch generated by test T_{111000} where $T_{111000} \varepsilon \bar{\tau}$ dominates any branch which is generated by test T_{100000} where $T_{100000} \varepsilon \tau$ if

$$C_{1,2,3} = \min_{i\varepsilon T(S_0)} [\bar{C}_i]$$

and

$$C_{1,2,3} \leq C_1 \cdot \rho(T_{111000})$$

Theorem 3.4.3

At any node N(S) with a state having at most four remaining untested LRUs, if test T_k such that $T_k \varepsilon \bar{\tau}$ has the minimum cost among all tests which can be used at N(S), then the branch generated by T_k dominates all branches which are generated by any other test T_m such that $T_m \varepsilon \bar{\tau}$.

Proof

With reference to Figure A.5, depicting a search tree for a four LRUs example, all tests which could be used at any node in this tree are presented in Table A.3

Branches $b_1(S_0,S_1,S_2)$, $b_2(S_0,S_3,S_4)$, and $b_3(S_0,S_5,S_6)$ emanated from the first node $N(S_0)$ by tests T_{1010}, T_{1100}, T_{1001} respectively, let the costs of these branches be $C_{b_1}(S_0,S_1,S_2)$, $C_{b_2}(S_0,S_3,S_4)$, and $C_{b_3}(S_0,S_5,S_6)$ respectively where all the tests belong to set $\bar{\tau}$

$$C_{b_1}(S_0,S_1,S_2) = \sum_{i=1}^{4} p_i \cdot E_i + C_{1,3} + (p_2+p_4) \cdot \min[C_4,C_2,C_{1,4},$$

$$C_{1,2}] + (p_1+p_3) \cdot \min[C_1,C_3,C_{1,4},C_{1,2}]$$

$$C_{b_2}(S_0,S_3,S_4) = \sum_{i=1}^{4} p_i \cdot E_i + C_{1,2} + (p_1+p_2) \cdot \min[C_1,C_2,$$

$$C_{1,3},C_{1,4}] + (p_3+p_4) \cdot \min[C_4,C_3,C_{1,3},C_{1,4}]$$

$$C_{b_3}(S_0,S_5,S_6) = \sum_{i=1}^{4} p_i \cdot E_i + C_{1,4} + (p_2+p_3) \cdot \min[C_3,C_2,$$

$$C_{1,2},C_{1,3}] + (p_1+p_4) \cdot \min[C_1,C_4,C_{1,2},C_{1,3}]$$

So, if $C_{1,3} = \min\limits_{i \varepsilon T(S_0)} [\bar{C}_i]$, then

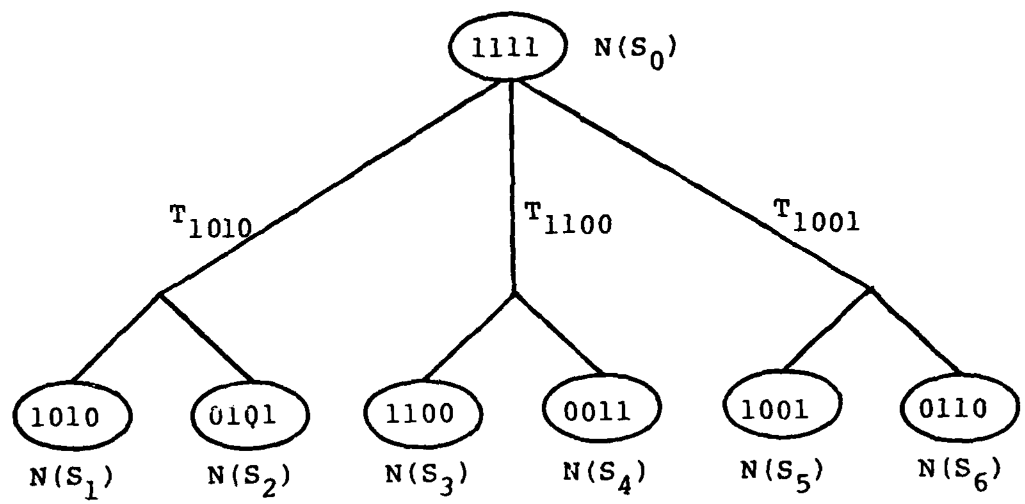


Figure A.5 A Search Tree for a 4-LRUs Example

TABLE A.3

TESTS WHICH COULD BE USED AT NODE S IN THE 4-LRUs EXAMPLE

S	T(S)
1010	T_{1000}, T_{0010}, T_{1001}, T_{1100}
0101	T_{0001}, T_{0100}, T_{1001}, T_{1100}
1100	T_{1000}, T_{0100}, T_{1010}, T_{1001}
0011	T_{0001}, T_{0010}, T_{1010}, T_{1001}
0110	T_{0010}, T_{0100}, T_{1100}, T_{1010}
1001	T_{1000}, T_{0001}, T_{1100}, T_{1010}
1111	T_{1000}, T_{0100}, T_{0010}, T_{0001}, T_{1100}, T_{1010}, T_{1001}

$$C_{1,3} = \min[C_1, C_2, C_3, C_4, C_{1,2}, C_{4,5}]$$ and branch $b_1(S_0, S_1, S_2)$ dominates branch $b_2(S_0, S_3, S_4)$

if $C_{1,3} + p_2 \cdot \min[C_4, C_2, C_{1,4}, C_{1,2}] + p_4 \cdot \min[C_4, C_2, C_{1,4}, C_{1,2}]$

$+ p_1 \cdot \min[C_1, C_3, C_{1,4}, C_{1,2}] + p_3 \cdot \min[C_1, C_3, C_{1,4}, C_{1,2}]$

$\leq C_{1,2} + p_2 \cdot \min[C_1, C_2, C_{1,3}, C_{1,4}] + p_1 \cdot \min[C_1, C_2, C_{1,3}, C_{1,4}]$

$+ p_3 \cdot \min[C_4, C_3, C_{1,3}, C_{1,4}] + p_4 \cdot \min\ [C_4, C_3, C_{1,3}, C_{1,4}]$

or $C_{1,3} + p_2 \cdot \min[C_4, C_{1,2}] + p_4 \cdot \min[C_2, C_{1,2}] + p_1 \cdot \min[C_3, C_{1,2}]$

$+ p_3 \cdot \min[C_1, C_{1,2}] \leq C_{1,2} + p_2 \cdot \min[C_1, C_{1,3}] + p_1 \cdot \min[C_2, C_{1,3}]$

$+ p_3 \cdot \min[C_4, C_{1,3}] + p_4 \cdot \min[C_3, C_{1,3}]$

or $[C_{1,3} < \text{cost} \leq C_{1,2}] < C_{1,2}$

which is satisfied only because $C_{1,3} = \min\limits_{i \epsilon T(S_0)} [\bar{C}_i]$

which implies that this is the only condition required to guarantee that branch $b_1(S_0, S_1, S_2)$ dominates branch $b_2(S_0, S_3, S_4)$.

By the same procedure, it could be proved that branch $b_1(S_0, S_1, S_2)$ dominates also branch $b_3(S_0, S_5, S_6)$ only if

$$C_{1,3} = \min_{i \epsilon T(S_0)} [\bar{C}_i]$$

Branch $b_1(S_0, S_1, S_2)$ dominates br[illegible]ch [illegible](S_0, S_5, S_6) if

$C_{1,3} + p_2 \cdot \min[C_4, C_{1,4}] + p_4 \cdot \min[C_2, C_{1,4}] + p_1 \cdot \min[C_3, C_{1,4}]$

$+ p_3 \cdot \min[C_1, C_{1,4}] \leq C_{1,4} + p_2 \cdot \min[C_3, C_{1,3}] + p_3 \cdot \min[C_2, C_{1,3}]$

$+ p_1 \cdot \min[C_4, C_{1,3}] + p_4 \cdot \min\ [C_1, C_{1,3}]$

or if $[C_{1,3} < \text{cost} \leq C_{1,4}] < C_{1,4}$

which is satisfied only because $C_{1,3}$ < cost of any other test.

Therefore, the branch which is generated by test T_{1010} where $T_{1010} \in \bar{\tau}$ which has the minimum cost at node $N(S_0)$ dominates any other branch generated by tests belong also to set $\bar{\tau}$.

Eventhough all the dominance rules are proved in case of having five or six LRUs, they could be considered reasonably as general cases. However, because of the dramatic increase in the number of possible tests $(2^{(n-1)} - 1)$ in case of having more than six LRUs, the proofs in these cases are omitted here.

```
C                               APPENDIX B


C       * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *


C                       M A I N   P R O G R A M


C       * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

C          PURPOSE
C     THIS PROGRAM USES A BRANCH AND BOUND ALGORITHM TO FIND THE OPTIMAL
C     SEQUENCE OF TESTS REQUIRED TO LOCATE A MALFUNCTIONED UNIT IN A
C     SYSTEM OF N LRU'S

C     INPUT
C            CONTROL CARDS
C            DATA CARDS


C     CONTROL CARDS  (A)
C     COLUMNS
C     1-5     .....N......NUMBER OF LRU'S
C     6-10    .....M......NUMBER OF TESTS

C     CONTROL CARDS   (B)...(ONE PER EACH TEST)
C     1-10  IXX(1,I,J)..TEST I DESCRIBED BY THE N BITS (EACH BIT IS
C     REPRESENTED BY J )


C     DATA CARDS   (A)...7F10.7
C     COLUMNS
C     1-10 ..........61-70...C(I) ..COST OF TEST I

C     DATA CARDS   (B)...7F10.7
C     COLUMNS
C     1-10 ..........61-70...P(J)...PROBABILITY OF FAILURE OF LRU J
```

```
C   DATA CARDS    (C)...7110
C   COLUMNS
C   1-10 ..........61-70...MH(J)...SECONDARY ISOLATION COST OF LRU J


C   PARAMETERS
C   N=NUMBER OF L.R.U.
C   M=NUMBER OF TESTS
C   P(J)=PROBABILITY OF FAILURE OF L.R.U. J
C   MH(J)=SECONDARY ISOLATION COST
C   C(I)=COST OF TEST I
C   IS= NODE NUMBER
C   IY(IS,J)=STATE OF NODE IS
C   IXX(IS,I,J)= TEST OF MINIMUM COST AT BRANCH I FROM NODE IS
C   IXX1(IS,I,J)=COMPLEMENT OF TEST IXX(IS,I,J)
C   CA(IS,I)=COST OF TEST OF MINIMUM COST AT BRANCH I FROM NODE IS
C   L(IS)=LEVEL OF NODE IS
C   ISS(IS)=PREVIOUS NODE TO NODE IS
C   K(IS)= NUMBER OF UNTESTED L.R.U.'S AT NODE IS
C   ALB(IS)=LOWER BOUND AT NODE IS
C   ID(IS).....IF ID(IS)=1 ...NODE IS IS DUMMY
C   INDEX(IS,I)=1  ......BRANCH I FROM NODE IS COULD BE USED
C   INDEX(IS,I)=0  ......BRANCH I FROM NODE IS COULD NOT BE USED
C   F(IS,I)=VALUE OF FIGURE OF MERIT TO BE USED INBRANCHING RULES FOR
C   BRANCH I AT NODE IS
C   JUPD(IS)=1    ...IFTHE DUMMY NODE IS IS UPDATED,ZERO OTHERWISE
      DIMENSION       LYS1(9),CRR(255),LYS2(9),L(12),JUPD(4)
      COMMON/ABC/IXX(11,255,9)
      COMMON/BBC/CA(11,255)
      COMMON/DEF/IXX1(11,255,9)
      COMMON/LMN/IY(11,9)
      COMMON/XYZ/ISS(11)
      COMMON/UVW/F(11,255)
      COMMON/GHI/R(11),ID(11)
      COMMON/ENT/K(11)
      COMMON/JHK/P(9)
      COMMON/BAB/INDEX(11,255)
      COMMON/KAL/ALB(11)
      COMMON/MAG/C(255)
      COMMON/ANZ/PR(9)
      COMMON/SEC/MH(9)
      COMMON/ANA/CR(255)
      CALL DATA(N,M)
C   AT THE FIRST NODE
      IS=1
      NUMBER=1
      NODE=1
      ISMAX=1
      L(IS)=0
      K(IS)=N
      ID(IS)=0
C   FIND UBBER BOUND  UB
      DO 162 J=1,N
 162  IY(IS,J)=1
      SECC=0.0
      DO 29 J=1,N
 29   SECC=SECC+P(J)*MH(J)
      SUMB=0.0
      DO 30 I=1,M
      CA(IS,I)=C(I)
      CRR(I)=C(I)
      ITT=0
      DO 31 J=1,N
 31   ITT=ITT+IXX(1,I,J)
      IF(ITT.EQ.1)GO TO 32
      CRR(I)=0.0
 32   SUMB=SUMB+CRR(I)
```

```
 30     CONTINUE
        CMAXI=CRR(1)
        DO 35 I=2,M
        IF(CMAXI-CRR(I))33,35,35
 33     CMAXI=CRR(I)
 35     CONTINUE
        SUMCC=SUMB-CMAXI
        UB=SUMCC+SECC
C   FIND LOWER BOUND
        CALL REXP(IS,N,M)
        ALB(IS)=R(IS)+SECC
C   STOPPING TEST USING SECONDARY ISOLATION
        HSUM=0.
        DO 37 J=1,N
 37     HSUM=HSUM+MH(J)
        IF((R(IS)+SECC).LT.HSUM) GO TO 34
        UB=HSUM
        WRITE(6,36)
 36     FORMAT(///,5X,'USE THE SECONDARY ISOLATION FOR ALL L.R.U.')
        GO TO 2752
 34     WRITE(6,859)IS,ALB(IS)
 859    FORMAT(//,10X,'IS=',I3,2X,'ALB(IS)=',F10.3)
        WRITE(6,222)UB
 222    FORMAT(//,10X,'UB=',F10.3)
 1000   CALL DOMINA(IS,N)
        CALL FINDF(IS,N)
 2000   CALL BRANCH(IS,IOPT)
        WRITE(6,380)
 380    FORMAT(///,10X,'IS',20X,'NODE',25X,'TEST')
        WRITE(6,381)IS,((IY(IS,J),J=1,N),(IXX(IS,IOPT,J),J=1,N))
 381    FORMAT(10X,I1,21X,9I1,20X,9I1)
C   FIND THE TWO POSSIBLE STATES FROM THE MOST PROMISING BRANCH
        MR=IS
        KS1=0
        KS2=0
        DO 1 J=1,N
        LYS1(J)=IY(IS,J)*IXX(IS,IOPT,J)
        LYS2(J)=IY(IS,J)*IXX1(IS,IOPT,J)
        IF(LYS1(J)-1)9,4,9
 4      KS1=KS1+1
 9      IF(LYS2(J).EQ.0) GO TO 1
        KS2=KS2+1
 1      CONTINUE
        IS=ISMAX+1
        IF(KS1.EQ.1) GO TO 6
        IF(KS2.EQ.1) GO TO 8
C   USE DUMMY NODE
        WRITE(6,383) IS
 383    FORMAT(//,5X,'NODE',2X,I4,2X,'IS DUMMY')
        ID(IS)=1
        JUPD(IS)=0
        L(IS)=L(MR)+1
        R(IS)=0.
        ISS(IS)=MR
C   FIND THE PARAMETERS OF THE TWO NODES BRANCHED FROM THE DUMMY NODE
        IF(KS2.GT.KS1) GO TO 7
        IS=IS+1
        DO 500 J=1,N
 500    IY(IS,J)=LYS2(J)
        L(IS)=L(MR)+2
        K(IS)=KS2
        ID(IS)=0
        ISS(IS)=IS-1
        CALL REXP(IS,N,M)
        CALL LOWERB(IS-1,N,IOPT)
        IS=IS+1
        DO 600 J=1,N
 600    IY(IS,J)=LYS1(J)
```

```
      K(IS)=KS1
      ISS(IS)=IS-2
      L(IS)=L(MR)+2
      ID(IS)=0
      NODE=NODE+3
      ISMAX=ISMAX+3
      CALL REXP(IS,N,M)
      CALL LOWERB(IS,N,IOPT)
      GO TO 800
 7    IS=IS+1
      DO 3 J=1,N
 3    IY(IS,J)=LYS1(J)
      L(IS)=L(MR)+2
      K(IS)=KS1
      ID(IS)=0
      ISS(IS)=IS-1
      CALL REXP(IS,N,M)
      CALL LOWERB(IS-1,N,IOPT)
      IS=IS+1
      DO 400 J=1,N
 400  IY(IS,J)=LYS2(J)
      K(IS)=KS2
      ISS(IS)=IS-2
      L(IS)=L(MR)+2
      ID(IS)=0
      NODE=NODE+3
      ISMAX=ISMAX+3
      CALL REXP(IS,N,M)
      CALL LOWERB(IS,N,IOPT)
      GO TO 800
C  FIND THE PARAMETERS OF THE NEW NODE
 8    DO 700 J=1,N
 700  IY(IS,J)=LYS1(J)
      K(IS)=KS1
      GO TO 10
 6    DO 900 J=1,N
 900  IY(IS,J)=LYS2(J)
      K(IS)=KS2
 10   ISS(IS)=MR
      L(IS)=L(MR)+1
      ID(IS)=0
      CALL REXP(IS,N,M)
      CALL LOWERB(IS,N,IOPT)
      NODE=NODE+1
      ISMAX=ISMAX+1
C  STOPPING TEST USING SECONDARY ISOLATION
 800  SUMPMH=0.
      SUMMH=0.
      DO 140 J=1,N
      IF(IY(IS,J).EQ.0) GO TO 140
      SUMPMH=SUMPMH+P(J)*MH(J)
      SUMMH=SUMMH+MH(J)
 140  CONTINUE
      IF(R(IS)-SUMPMH-SUMMH)165,166,166
 166  BLB=ALB(IS)-R(IS)-SUMPMH+SUMMH
      IF(BLB-UB)4000,168,168
 4000 WRITE(6,976)(IY(IS,J),J=1,N)
 976  FORMAT(///,10X,'USE SECONDARY ISOLATION FOR UNTESTED L.R.U.
     2AT NODE',2X,20I1)
      WRITE(6,870)BLB
 870  FORMAT(//,10X,'BLB=',2X,F6.3)
      M1=(2**(K(IS)-1))-1
      DO 169 I=1,M1
 169  F(IS,I)=0.
      GO TO 167
 165  IF(ALB(IS)-UB)333,168,168
 333  CALL GENBRA(IS,N,M)
      IF(K(IS)-2)1000,172,1000
```

```
C   LAST NODE IN THE BRANCH
 172   SUMP3=0.
       DO 173 J=1,N
       IF(IY(IS,J).EQ.0)   GO TO 173
       SUMP3=SUMP3+P(J)
 173   CONTINUE
C   FIND THE ACTUAL EXPECTED COST OF THIS BRANCH
       BN=(SUMP3)*CA(IS,1)
       BLB=ALB(IS)-R(IS)+BN
       WRITE(6,380)
       WRITE(6,381)IS,((IY(IS,J),J=1,N),(IXX(IS,1,J),J=1,N))
       WRITE(6,808)BLB
 808   FORMAT(//,10X,'BLB=',F10.3)
       IF(BLB-UB)167,168,168
C   GO TO OTHER BRANCH OF THE DUMMY NODE TO ADJUST THE LOWER BOUND
 167   IF(L(IS)-1)174,175,174
 174   IF(ID(ISS(IS)).NE.1) GO TO 178
       IF(ID(IS-1).EQ.1) GO TO 176
       GO TO 177
 178   IS=ISS(IS)
       GO TO 167
 176   IS=IS-1
       GO TO 167
 177   IS=IS-2
       JUPD(IS)=1
       ALB(IS)=BLB-R(IS+1)
       IS=IS+1
       CALL LOWERB(IS,N,IOPT)
       GO TO 800
C   FATHOM THIS NODE
 168   M1=(2**(K(IS)-1))-1
       DO 180 I=1,M1
 180   F(IS,I)=0.
       WRITE(6,330)
 330   FORMAT(//,10X,'FATHOM LAST NODE  AND CANCEL LAST TEST ')
       IF(ID(ISS(IS))-1)201,181,201
 666   IS=ISS(IS)
       GO TO 170
 181   M1=(2**(K(IS)-1))-1
       DO 182 I=1,M1
 182   F(IS-1,I)=0.
       IS=IS-2
       GO TO 170
C   A FEASIBLE SOLUTION
 175   UB=BLB
       WRITE(6,222)UB
       WRITE(6,392)NODE
 392   FORMAT(//,3X,'    NUMBER OF ACTIVE NODES IS',I8)
       NUMBER=NUMBER+1
       IS=ISMAX
C   BACKTRACK
 170   IF(L(IS).EQ.0) GO TO 299
       IF(L(IS).EQ.1) GO TO 200
       IF(ID(IS).EQ.1) GO TO 201
       IF(K(IS).EQ.2) GO TO 201
       M1=(2**(K(IS)-1))-1
       DO 202 I=1,M1
       IF(F(IS,I).NE.0) GO TO 300
 202   CONTINUE
       IF(ID(ISS(IS)).NE.1) GO TO 201
       IF(JUPD(ISS(IS))-1)666,201,666
 201   IS=IS-1
       GO TO 170
 200   IF(ID(IS).EQ.1) GO TO  204
       IF(K(IS).EQ.2) GO TO 204
       M1=(2**(K(IS)-1))-1
       DO 205 I=1,M1
       IF(F(IS,I).NE.0.) GO TO 300
```

```
205   CONTINUE
204   IS=1
299   DO 206 I=1,M
      IF(F(IS,I).NE.0.) GO TO 300
206   CONTINUE
      GO TO 2752
300   ISMAX=IS
      GO TO 2000
2752  WRITE(6,391)UB
391   FORMAT(///,5X,'THE OPTIMUM EXPECTED COST IS ',F10.3)
      WRITE(6,392)NODE
      STOP
      END
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

C                    S U B R O U T I N E   D A T A

C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

C     PURPOSE
C  DATA IS USED TO READ BOTH CONTROL AND DATA CARDS
      SUBROUTINE DATA(N,M)
      COMMON/ABC/IXX(11,255,9)
      COMMON/DEF/IXX1(11,255,9)
      COMMON/ANZ/PR(9)
      COMMON/MAG/C(255)
      COMMON/JHK/P(9)
      COMMON/SEC/MH(9)
      READ (5,190)N,M
 190  FORMAT(2I5)
      WRITE(6,191)N,M
 191  FORMAT(1H1,//,1X,'NUMBER OF L.R.U. IS',I2,10X,'NUMBER OF TESTS
     1IS ',I4)
      WRITE(6,301)
 301  FORMAT(///,6X,'I',10X,'IXX(1,I,J)')
      DO 304 I=1,M
      READ(5,302)(IXX(1,I,J),J=1,N)
 302  FORMAT(10I1)
      WRITE(6,977)I,(IXX(1,I,J),J=1,N)
 977  FORMAT(5X,I3,10X,10I1)
 304  CONTINUE
C  FIND IXX1(1,I,J)
      DO 160 I=1,M
      DO 161 J=1,N
      IF(IXX(1,I,J).EQ.1) GO TO 163
      IXX1(1,I,J)=1
      GO TO 161
 163  IXX1(1,I,J)=0
 161  CONTINUE
 160  CONTINUE
      WRITE(6,195)
 195  FORMAT(1H1,//,20X,'COST OF TESTS',//)
      DO 194 I=1,M
      READ(5,6000)C(I)
 6000 FORMAT(7F10.7)
      WRITE(6,193)I,C(I)
 193  FORMAT(10X,'COST OF TEST ',I4,2X,'IS',2X,F10.7,/)
 194  CONTINUE
      WRITE(6,196)
 196  FORMAT(///,6X,'J',13X,'P',15X,'MH')
      READ (5,192) (P(J),J=1,N)
```

```
  192 FORMAT(7F10.7)
      READ (5,6001) (MH(J),J=1,N)
 6001 FORMAT(7I10)
      DO 197 J=1,N
      WRITE(6,199)J,P(J),MH(J)
 199  FORMAT(5X,I2,5X,F10.7,5X,I10)
 197  CONTINUE
      RETURN
      END
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *


C                 S U B R O U T I N E   L O W E R B


C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *


C     PURPOSE
C  LOWERB IS USED TO FIND THE LOWER BOUND AT NODE IS
      SUBROUTINE LOWERB(IS,N,IOPT)
      COMMON/LMN/IY(11,9)
      COMMON/KAL/ALB(11)
      COMMON/GHI/R(11),ID(11)
      COMMON/XYZ/ISS(11)
      COMMON/BRC/CA(11,255)
      COMMON/JHK/P(9)
      COMMON/ANZ/PR(9)
      DO 499 J=1,N
 499  PR(J)=P(J)
      IS1=ISS(IS)
      DO 43 J=1,N
      IF(IY(IS1,J).EQ.1) GO TO 43
      PR(J)=0.
 43   CONTINUE
C  SUMP1=SUM OF THE PROBILITY OF FAILURE OF THE UNTESTED L.R.U. AT THE
C  PREVIOUS NODE TO NODE IS
      SUMP1=0.
      DO 44 J=1,N
 44   SUMP1=SUMP1+PR(J)
      IF(ID(IS1)-1) 48,47,48
 47   CISS=0.
      GO TO 45
 48   CISS=SUMP1*CA(IS1,IOPT)
      IF(ID(IS).EQ.1) GO TO 46
 45   ALB(IS)=ALB(IS1)-R(IS1)+R(IS)+CISS
      GO TO 49
 46   ALB(IS)=ALB(IS1)-R(IS1)+R(IS)+CISS+R(IS+1)
 49   WRITE(6,859)IS,ALB(IS)
 859  FORMAT(//,10X,'IS=',I3,2X,'ALB(IS)=',F10.3)
      RETURN
      END
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *


C                 S U B R O U T I N E   B R A N C H


C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

```
C       PURPOSE
C   FIND THE BRANCH IOPT WITH MAXIMUM VALUE OF F(IS,I)
        SUBROUTINE BRANCH(IS,IOPT)
        COMMON/UVW/F(11,255)
        COMMON/ENT/K(11)
        M1=(2**(K(IS)-1))-1
        FMAX=F(IS,1)
        DO 96 I=2,M1
        IF(FMAX-F(IS,I)) 95,96,96
   95   FMAX=F(IS,I)
   96   CONTINUE
        DO 94 I=1,M1
        IF(F(IS,I).EQ.FMAX) GO TO 97
   94   CONTINUE
   97   IOPT=I
        F(IS,I)=0.
        RETURN
        END
C       * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

C                      S U B R O U T I N E   F I N D F

C       * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

C       PURPOSE
C   FINDF IS USED TO FIND THE VALUES OF  F(IS,I) OF ALL BRANCHES I=1,M1
C   AT NODE IS WHICH WILL BE USED IN THE BRANCHING RULES
        SUBROUTINE FINDF(IS,N)
        COMMON/ABC/IXX(11,255,9)
        COMMON/BBC/CA(11,255)
        COMMON/UVW/F(11,255)
        COMMON/LMN/IY(11,9)
        COMMON/JHK/P(9)
        COMMON/BAB/INDEX(11,255)
        COMMON/ENT/K(11)
        WRITE(6,863)
  863   FORMAT(///,11X,'I',23X,' F(IS,I)')
        M1=(2**(K(IS)-1))-1
C   PP1=SUM OF THE PROB.     OF FAILURE OF ALL UNTESTED L.R.U. AT NODE IS
C   PP2=SUM OF THE PROB.     OF FAILURE OF ALL UNTESTED L.R.U. IF THE
C   TEST IN BRANCH I PASSES
C   PP=PROBABILITY THAT THE TEST IN BRANCH I WILL PASS
        PP1=0.
        DO 90 J=1,N
        IF(IY(IS,J).EQ.0) GO TO 90
        PP1=PP1+P(J)
   90   CONTINUE
        DO 91 I=1,M1
        IF(INDEX(IS,I).EQ.0) GO TO 93
        PP2=0.
        DO 92 J=1,N
        IXYY=IY(IS,J)*IXX(IS,I,J)
        IF(IXYY.EQ.0) GO TO 92
        PP2=PP2+P(J)
   92   CONTINUE
        PP=PP2/PP1
        F(IS,I)=-(PP*ALOG(PP)/.693+(1.-PP)*ALOG(1.-PP)/.693)/CA(IS,I)
        GO TO 864
   93   F(IS,I)=0.
  864   WRITE(6,862)I,F(IS,I)
```

```
 862   FORMAT(10X,I4,20X,F10.4)
  91   CONTINUE
       RETURN
       END
C      * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

C                      S U B R O U T I N E   R E X P

C      * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

C      PURPOSE
C   REXP IS USED TO FIND THE MINIMUM EXPECTED COST TO FIND MALFUNCTIONED
C   L.R.U. FROM NODE IS
       SUBROUTINE REXP(IS,N,M)
       COMMON/LMN/IY(11,9)
       COMMON/GHI/R(11),ID(11)
       COMMON/ENT/K(11)
       COMMON/JHK/P(9)
       COMMON/MAG/C(255)
       COMMON/ANA/CR(255)
       COMMON/ANZ/PR(9)
       DO 41 J=1,N
       PR(J)=P(J)
       IF(IY(IS,J).EQ.1) GO TO 41
       PR(J)=1.
  41   CONTINUE
C   PMIN1=THE MINIMUM     PROBABILITY OF FAILURE AMONG THE UNTESTED
C   L.R.U. AT NODE IS
C   PMIN2=THE SECOND MIN. PROBABILITY OF FAILURE AMONG THE UNTESTED
C   L.R.U. AT NODE IS
       PMIN1=PR(1)
       DO 14 J=2,N
       IF(PR(J)-PMIN1) 13,13,14
  13   PMIN1=PR(J)
  14   CONTINUE
       DO 16 J=1,N
       IF(PR(J).EQ.PMIN1) GO TO 17
  16   CONTINUE
  17   PR(J)=1.
       PMIN2=PR(1)
       DO 20 J=2,N
       IF(PR(J)-PMIN2) 19,19,20
  19   PMIN2=PR(J)
  20   CONTINUE
       DO 599 I=1,M
 599   CR(I)=C(I)
       JJ=M-K(IS)+1
       DO 27 I=1,JJ
       CMAX=CR(1)
       DO 24 J=2,M
       IF(CMAX-CR(J)) 23,24,24
  23   CMAX=CR(J)
  24   CONTINUE
       DO 25 KZ=1,M
       IF(CR(KZ).EQ.CMAX) GO TO 26
  25   CONTINUE
  26   CR(KZ)=0.
  27   CONTINUE
C   CSUM=SUM OF THE (K(IS)-1) MINIMUM COSTS OF TESTS
       CSUM=0.
       DO 28 I=1,M
```

```
 28    CSUM=CSUM+CR(I)
       R(IS)=(PMIN1+PMIN2)*CSUM
       WRITE(6,839)IS,R(IS)
 839   FORMAT(//,10X,'IS=',I3,3X,'R(IS)=',F10.3)
 15    RETURN
       END
C      * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *


C                      S U B R O U T I N E   D O M I N A


C      * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *


C      PURPOSE
C   DOMINA IS USED TO FIND  INDEX(IS,I) AT NODE IS FOR ALL POSSIBLE
C   BRANCHES
       SUBROUTINE DOMINA(IS,N)
C   IF INDEX(IS,I)=0 ...BRANCH I IS DOMINATED BY ANOTHER BRANCH WHICH
C   ITS INDEX EQUALS 1
       DIMENSION PT1(255),PT2(255),PT(255),KSS1(255),KSS2(255),CAA(255)
       DIMENSION IYSS1(9),IYSS2(9),PAA(9)
       COMMON/ABC/IXX(11,255,9)
       COMMON/BBC/CA(11,255)
       COMMON/DEF/IXX1(11,255,9)
       COMMON/LMN/IY(11,9)
       COMMON/ENT/K(11)
       COMMON/JHK/P(9)
       COMMON/BAB/INDEX(11,255)
       M1=(2**(K(IS)-1))-1
       DO 50 I=1,M1
       PT1(I)=0.
       PT2(I)=0.
       KSS1(I)=0
       KSS2(I)=0
       DO 51 J=1,N
       IYSS1(J)=IY(IS,J)*IXX(IS,I,J)
       IYSS2(J)=IY(IS,J)*IXX1(IS,I,J)
       IF(IYSS1(J).EQ.0) GO TO 52
       KSS1(I)=KSS1(I)+1
       PT1(I)=PT1(I)+P(J)
 52    IF(IYSS2(J).EQ.0) GO TO 51
       KSS2(I)=KSS2(I)+1
       PT2(I)=PT2(I)+P(J)
 51    CONTINUE
       IF(KSS1(I)-KSS2(I)) 54,53,55
 53    IF(PT1(I)-PT2(I)) 54,54,55
 54    PT(I)=PT1(I)
       GO TO 50
 55    PT(I)=PT2(I)
 50    CONTINUE
       DO 60 I=1,M1
 60    CAA(I)=CA(IS,I)
C   FIND MINIMUM COST
       CAAMIN=CAA(1)
       DO 62 I=2,M1
       IF(CAA(I)-CAAMIN)61,61,62
 61    CAAMIN=CAA(I)
 62    CONTINUE
       DO 63 I=1,M1
       IF(CAA(I).EQ.CAAMIN) GO TO 64
 63    CONTINUE
 64    CA11=CAA(I)
```

```
      CAA(I)=100000.
      IAA=I
      DO 65 I=2,M1
      IF(CAA(I)-C22)66,66,65
 66   C22=CAA(I)
 65   CONTINUE
      DO 70 J=1,N
      PAA(J)=P(J)
      IF(IY(IS,J).EQ.1) GO TO 70
      PAA(J)=1.
 70   CONTINUE
      PAAMIN=PAA(1)
      DO 73 J=2,N
      IF(PAA(J)-PAAMIN) 72,72,73
 72   PAAMIN=PAA(J)
 73   CONTINUE
C  CHECK IF THE BRANCH WITH MINIMUM COST OF TEST DOMINATES ALL OTHER
C  BRANCHES OR NOT
      IF(CA11-PAAMIN*C22)74,74,75
 74   DO 76 I=1,M1
 76   INDEX(IS,I)=0
      GO TO 80
 75   IF(KSS1(IAA)-KSS2(IAA)) 77,78,77
 77   DO 79 I=1,M1
      IF(CA11-PT(IAA*CA(IS,I)))81,81,82
 81   INDEX(IS,I)=0
      GO TO 79
 82   INDEX(IS,I)=1
 79   CONTINUE
      GO TO 80
 78   DO 83 I=1,M1
      IF(KSS1(I)-KSS2(I))84,85,84
 84   IF(CA11-PT(IAA*CA(IS,I)))86,86,87
 86   INDEX(IS,I)=0
      GO TO 83
 87   INDEX(IS,I)=1
      GO TO 83
 85   IF(K(IS)-4)88,89,88
 88   INDEX(IS,I)=1
      GO TO 83
 89   INDEX(IS,I)=0
 83   CONTINUE
 80   INDEX(IS,IAA)=1
      RETURN
      END
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *


C                    S U B R O U T I N E   G E N B R A


C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *


C     PURPOSE
C  GENBRA IS USED TO GENERATE ALL POSSIBLE BRANCHES FROM NODE IS .
C  THEN FIND THE TEST WITH MIN COST IN EACH BRANCH
      SUBROUTINE GENBRA(IS,N,M)
      DIMENSION IXYX(255,9),KS1(255),MMM(255)
      COMMON/ABC/IXX(11,255,9)
      COMMON/BBC/CA(11,255)
      COMMON/DEF/IXX1(11,255,9)
      COMMON/LMN/IY(11,9)
      COMMON/MAJ/C(255)
```

```
      COMMON/ENT/K(11)
      DO 100  I=1,M
      KS1(I)=0
      DO 101  J=1,N
      IXYX(I,J)=IY(IS,J)*IXX(1,I,J)
      IF(IXYX(I,J).EQ.0) GO TO 101
      KS1(I)=KS1(I)+1
101   CONTINUE
100   CONTINUE
      IJK=(K(IS)/2)*2
      IF(IJK.EQ.K(IS)) GO TO 110
      DO 102 I=1,M
      IF(KS1(I).LT.(K(IS)+1)/2)GO TO 104
      DO 103 J=1,N
      IXYX(I,J)=IY(IS,J)*IXX1(1,I,J)
103   CONTINUE
104   DO 105 J=1,N
      IF(IXYX(I,J).EQ.1) GO TO 108
105   CONTINUE
121   MMM(I)=0
      GO TO 102
108   DO 106 J=1,N
      IF(IXYX(I,J)-IY(IS,J)) 109,106,109
106   CONTINUE
      GO TO 121
109   MMM(I)=1
102   CONTINUE
      GO TO 120
110   DO 130 I=1,M
      IF(KS1(I)-K(IS)/2)124,129,122
122   DO 123 J=1,N
123   IXYX(I,J)=IY(IS,J)*IXX1(1,I,J)
124   DO 125 J=1,N
      IF(IXYX(I,J).EQ.1)GO TO 128
125   CONTINUE
131   MMM(I)=0
      GO TO 130
128   DO 126 J=1,N
      IF(IXYX(I,J)-IY(IS,J))129,126,129
126   CONTINUE
      GO TO 131
129   MMM(I)=1
130   CONTINUE
      GO TO 1200
120   WRITE(6,650)
650   FORMAT(///,11X,'I',20X,'IXX(IS,I,J)')
      I=1
      DO 111 II=1,M
      IF(MMM(II).EQ.0) GO TO 111
      KM=II
      LLL=II+1
      DO 112 L=LLL,M
      IF(MMM(L).EQ.0)GO TO 112
      DO 115 J=1,N
      IF(IXYX(II,J)-IXYX(L,J))112,115,112
115   CONTINUE
      MMM(L)=0
      IF(C(KM)-C(L))118,118,116
116   KM=L
118   CA(IS,I)=C(KM)
      DO 119 J=1,N
      IXX(IS,I,J)=IXX(1,KM,J)
      IF(IXX(IS,I,J).EQ.1)GO TO 5
      IXX1(IS,I,J)=1
      GO TO 119
5     IXX1(IS,I,J)=0
119   CONTINUE
112   CONTINUE
```

```
      I=I+1
111   CONTINUE
      GO TO 1600
1200  WRITE(6,850)
      I=1
      DO 1111 II=1,M
      IF(MMM(II).EQ.0) GO TO 1111
      KM=II
      LLL=II+1
      DO 1112 L=LLL,M
      IF(MMM(L).EQ.0)GO TO 1112
      DO 1115 J=1,N
      IF(IXYX(II,J)-IXYX(L,J))1113,1115,1113
1115  CONTINUE
1117  MMM(L)=0
      IF(C(KM)-C(L))1118,1118,1116
1116  KM=L
1118  CA(IS,I)=C(KM)
      GO TO 1700
1113  DO 1114 JU=1,N
      IF(IXYX(II,JU)+IXYX(L,JU)-IY(IS,JU))1112,1114,1112
1114  CONTINUE
      GO TO 1117
1700  DO 1119 J=1,N
      IXX(IS,I,J)=IXX(1,KM,J)
      IF(IXX(IS,I,J).EQ.1)GO TO 1005
      IXX1(IS,I,J)=1
      GO TO 1119
1005  IXX1(IS,I,J)=0
1119  CONTINUE
1112  CONTINUE
      I=I+1
1111  CONTINUE
1600  M1=(2**(K(IS)-1))-1
      DO 852 I=1,M1
      WRITE(6,851)I,(IXX(IS,I,J),J=1,N)
851   FORMAT(10X,I4,20X,9I1)
852   CONTINUE
      RETURN
      END
```

DATE

FILMED