CLASSIFICATION VIA SEQUENTIAL TESTING

by

ÖMER ERHUN KUNDAKCIOĞLU

Submitted to the Graduate School of Engineering and Natural Sciences
in partial fulfillment of
the requirements for the degree of
Master of Science

Sabancı University
Spring 2004

CLASSIFICATION VIA SEQUENTIAL TESTING

APPROVED BY

Assist. Prof. Dr. Tonguç Ünlüyurt .....
(Thesis Supervisor)

Prof. Dr. Gündüz Ulusoy .....

Assist. Prof. Dr. Berrin Yanıkoğlu .....

DATE OF APPROVAL: .....

to my parents...

Acknowledgments

I am indebted in the preperation of this thesis to my supervisor, Assist. Prof. Dr. Tonguç Ünlüyurt, whose patience and kindness, as well as academic experience, have been invaluable to me.

I am extremely grateful to Prof. Dr. Gündüz Ulusoy and Assist. Prof. Dr. Berrin Yanıkoğlu for their comments, their time spent on my thesis and serving on my thesis committee.

The informal support and encouragement of many colleagues has been indispensable, and I would like particulary to acknowledge the contribution of Adil Soydal, Pınar Yılmaz, Utku Köktürk, Mehmet Kayhan, Bilge Aksan, Onur Çotur, and Emre Tavşancıl. I am also grateful to Ferit Akova, Burakhan Yalçın, Kamer Kaya, Oğuz Atan, Sinan Özgür, and Kemal Sümer for helping me get through the difficult times, and for all the emotional support, entertainment, and caring they provided.

I am forever indebted to my high school math teacher Mehmet Uz for his enthusiasm, his inspiration, his great efforts to explain things clearly and simply, and making mathematics fun for me.

I wish to thank my parents and my sister, the constant source of support, for their understanding, endless patience, and encouragement when it was mostly required.

Finally, I wish to thank Aysan Şirin, who has always been my pillar and my guiding light, for her understanding, patience, support, encouragement, favors, and all the other things that make it so worthwhile to know her.

CLASSIFICATION VIA SEQUENTIAL TESTING

Abstract

The problem of generating the sequence of tests required to reach a diagnostic conclusion with minimum average cost, which is also known as *test sequencing problem*, is considered. The test sequencing problem is formulated as an optimal binary AND/OR decision tree construction problem, whose solution is known to be NP-complete. The problem can be solved optimally using dynamic programming or AND/OR graph search methods (AO^*, CF, and HS). However, for large systems, the associated computational effort with dynamic programming or AND/OR graph search methods is substantial, due to the rapidly increasing number of nodes in AND/OR search graph. In order to prevent the computational explosion, one-step or multistep lookahead heuristic algorithms have been developed to solve the test sequencing problem. Our approach is based on integrating concepts from the one-step lookahead heuristic algorithms and the strategies used in Huffman coding. The effectiveness of the algorithms is demonstrated on several test cases. The traditional test sequencing problem is generalized here to include asymmetrical tests. Our approach to test sequencing can be adapted to solve a wide variety of binary identification problems arising in decision table programming, medical diagnosis, database query processing, quality assurance, and pattern recognition.

SIRALI TESTLER İLE SINIFLANDIRMA

Özet

Test düzenleme problemi adı da verilen, minimum maliyetle teşhis koymak için gerekli test sırası oluşturma problemi ele alınmıştır. Test düzenleme problemi, çözümünün NP-tam olduğu bilinen ikili VE/VE YA karar ağacı şeklinde formüle edilebilir. Problemin en iyi çözümü dinamik programlama ve ya VE/VE YA grafiği arama yöntemleriyle (AO^*, CF, ve HS) elde edilebilir. Ancak büyük sistemlerde, dinamik programlama ve ya VE/VE YA arama yöntemleri, VE/VE YA arama grafiğinde hızla artan noktalar yüzünden, ağır hesaplamaları beraberinde getirmektedir. Bu hesaplama patlamasının üstesinden gelmek için, test düzenleme problemini çözecek bir-adım ya da çok-adım ileri bakma yöntemi algoritmaları geliştirildi. Bizim yaklaşımımız, bir-adım ileri bakma yöntemi algoritmalarıyla, Huffman kodlamasında kullanılan stratejileri birleştirmektir. Algoritmaların etkinliği bir çok test durumu için gösterilmiştir. Geleneksel test düzenleme problemi asimetrik testler de katılarak genelleştirilmiştir. Test düzenleme problemine yaklaşımımız, karar tablosu problemi, tıbbî tanı, veritabanı sorgu işleme, kalite güvencesi, ve örüntü tanıma problemlerinde karşılaşılan ikili teşhis problemlerine uyarlanabilir.

Table of Contents

Acknowledgments **v**

Abstract **vi**

Özet **vii**

1 Introduction **1**
1.1 Motivation 1
1.2 Problem Definition 2
1.3 An Example 6

2 Literature Review **8**
2.1 Systems with Symmetrical Tests 8
2.2 Systems with Asymmetrical Tests 11
2.3 Two Polynomial Time Cases 12
2.4 Noiseless Coding Problem 12
2.5 Systems with Multivalued Tests 18

3 Tests with Uniform Costs **19**
3.1 A Note on Binding Strategy 19
3.2 Inclusion of Asymmetrical Tests 24
3.3 A Heuristic Based on Huffman Coding 25
3.4 An Example 26
3.5 Computational Results of Modified Huffman Coding 28

4 Tests with Non-Uniform Costs **30**
4.1 A Heuristic Based on Binding and Rollout Strategies 31
4.2 Heuristic Test Sequencing Algorithms Employed 34
4.3 Computational Complexity of Bind and Rollout Algorithm 34
4.4 Proficiency of Bind and Rollout Algorithm 35
4.5 An Example 36

5 Computational Results **39**

6 Conclusion and Extensions **45**

Appendix **49**

A Failure Probability Computation **49**
A.1 Prior Probabilities of Failures 49
A.2 Conditional Failure Probabilities 49

List of Figures

1.1 Decision tree for a test sequencing problem 4

1.2 AND/OR binary decision tree for a test sequencing problem 5

1.3 An optimal test sequence for the example presented in Table 1.2 . . . 7

2.1 Optimum binary coding procedure 14

2.2 Optimum binary coding procedure: A different perspective 16

2.3 Test sequencing for example 2.4.1 16

2.4 Alternative test sequencing for example 2.4.1 17

3.1 Traditional approach vs. binding strategy 20

3.2 One path leading the optimal solution using binding strategy 21

3.3 An unallowable binding for the case in Table 3.1 23

3.4 Application of separation heuristic for the case in Table 3.3 26

3.5 Application of Huffman coding based algorithm for the case in Table 3.3 28

4.1 Binary decision tree constructed using modified bind and rollout algorithm 38

List of Tables

1.1 Symmetrical test vs. asymmetrical test 3

1.2 Example: Diagnostic dictionary matrix, fault probabilities and test costs 6

2.1 Results of optimum binary coding procedure 14

2.2 Tests required to use results of noiseless coding for Example 2.4.1 . . 15

2.3 Alternative tests required to use results of noiseless coding for the example 2.4.1 17

3.1 Disadvantage of using binding strategy 22

3.2 Appropriate form to apply binding strategy 24

3.3 An example for Huffman coding based algorithm 26

3.4 First allowance check for the case in Table 3.3 27

3.5 Second allowance check for the case in Table 3.3 27

3.6 Third allowance check for the case in Table 3.3 27

3.7 Computational results of modified Huffman coding 29

4.1 An example to show that Huffman coding based algorithm lacks a tool to deal with asymmetrical tests 31

4.2 An example with asymmetrical tests and non-uniform costs 36

4.3 First iteration of modified bind and rollout algorithm 37

4.4 The case when s_2 and $s_{2'}$ are temporarily bind 38

5.1 Size of test problems 40

5.2 Information heuristic, modified separation heuristic, random solution, and modified bind and rollout heuristic versus optimal solution 43

5.3 Average worst results 44

5.4 Average time spent for test problems 44

CLASSIFICATION VIA SEQUENTIAL TESTING

by

ÖMER ERHUN KUNDAKCIOĞLU

Submitted to the Graduate School of Engineering and Natural Sciences
in partial fulfillment of
the requirements for the degree of
Master of Science

Sabancı University
Spring 2004

CLASSIFICATION VIA SEQUENTIAL TESTING

APPROVED BY

Assist. Prof. Dr. Tonguç Ünlüyurt .....
(Thesis Supervisor)

Prof. Dr. Gündüz Ulusoy .....

Assist. Prof. Dr. Berrin Yanıkoğlu .....

DATE OF APPROVAL: .....

to my parents...

Acknowledgments

I am indebted in the preperation of this thesis to my supervisor, Assist. Prof. Dr. Tonguç Ünlüyurt, whose patience and kindness, as well as academic experience, have been invaluable to me.

I am extremely grateful to Prof. Dr. Gündüz Ulusoy and Assist. Prof. Dr. Berrin Yanıkoğlu for their comments, their time spent on my thesis and serving on my thesis committee.

The informal support and encouragement of many colleagues has been indispensable, and I would like particulary to acknowledge the contribution of Adil Soydal, Pınar Yılmaz, Utku Köktürk, Mehmet Kayhan, Bilge Aksan, Onur Çotur, and Emre Tavşancıl. I am also grateful to Ferit Akova, Burakhan Yalçın, Kamer Kaya, Oğuz Atan, Sinan Özgür, and Kemal Sümer for helping me get through the difficult times, and for all the emotional support, entertainment, and caring they provided.

I am forever indebted to my high school math teacher Mehmet Uz for his enthusiasm, his inspiration, his great efforts to explain things clearly and simply, and making mathematics fun for me.

I wish to thank my parents and my sister, the constant source of support, for their understanding, endless patience, and encouragement when it was mostly required.

Finally, I wish to thank Aysan Şirin, who has always been my pillar and my guiding light, for her understanding, patience, support, encouragement, favors, and all the other things that make it so worthwhile to know her.

CLASSIFICATION VIA SEQUENTIAL TESTING

Abstract

The problem of generating the sequence of tests required to reach a diagnostic conclusion with minimum average cost, which is also known as *test sequencing problem*, is considered. The test sequencing problem is formulated as an optimal binary AND/OR decision tree construction problem, whose solution is known to be NP-complete. The problem can be solved optimally using dynamic programming or AND/OR graph search methods (AO^*, CF, and HS). However, for large systems, the associated computational effort with dynamic programming or AND/OR graph search methods is substantial, due to the rapidly increasing number of nodes in AND/OR search graph. In order to prevent the computational explosion, one-step or multistep lookahead heuristic algorithms have been developed to solve the test sequencing problem. Our approach is based on integrating concepts from the one-step lookahead heuristic algorithms and the strategies used in Huffman coding. The effectiveness of the algorithms is demonstrated on several test cases. The traditional test sequencing problem is generalized here to include asymmetrical tests. Our approach to test sequencing can be adapted to solve a wide variety of binary identification problems arising in decision table programming, medical diagnosis, database query processing, quality assurance, and pattern recognition.

SIRALI TESTLER İLE SINIFLANDIRMA

Özet

Test düzenleme problemi adı da verilen, minimum maliyetle teşhis koymak için gerekli test sırası oluşturma problemi ele alınmıştır. Test düzenleme problemi, çözümünün NP-tam olduğu bilinen ikili VE/VE YA karar ağacı şeklinde formüle edilebilir. Problemin en iyi çözümü dinamik programlama ve ya VE/VE YA grafiği arama yöntemleriyle (AO^*, CF, ve HS) elde edilebilir. Ancak büyük sistemlerde, dinamik programlama ve ya VE/VE YA arama yöntemleri, VE/VE YA arama grafiğinde hızla artan noktalar yüzünden, ağır hesaplamaları beraberinde getirmektedir. Bu hesaplama patlamasının üstesinden gelmek için, test düzenleme problemini çözecek bir-adım ya da çok-adım ileri bakma yöntemi algoritmaları geliştirildi. Bizim yaklaşımımız, bir-adım ileri bakma yöntemi algoritmalarıyla, Huffman kodlamasında kullanılan stratejileri birleştirmektir. Algoritmaların etkinliği bir çok test durumu için gösterilmiştir. Geleneksel test düzenleme problemi asimetrik testler de katılarak genelleştirilmiştir. Test düzenleme problemine yaklaşımımız, karar tablosu problemi, tıbbî tanı, veritabanı sorgu işleme, kalite güvencesi, ve örüntü tanıma problemlerinde karşılaşılan ikili teşhis problemlerine uyarlanabilir.

Table of Contents

Acknowledgments **v**

Abstract **vi**

Özet **vii**

1 Introduction **1**
1.1 Motivation 1
1.2 Problem Definition 2
1.3 An Example 6

2 Literature Review **8**
2.1 Systems with Symmetrical Tests 8
2.2 Systems with Asymmetrical Tests 11
2.3 Two Polynomial Time Cases 12
2.4 Noiseless Coding Problem 12
2.5 Systems with Multivalued Tests 18

3 Tests with Uniform Costs **19**
3.1 A Note on Binding Strategy 19
3.2 Inclusion of Asymmetrical Tests 24
3.3 A Heuristic Based on Huffman Coding 25
3.4 An Example 26
3.5 Computational Results of Modified Huffman Coding 28

4 Tests with Non-Uniform Costs **30**
4.1 A Heuristic Based on Binding and Rollout Strategies 31
4.2 Heuristic Test Sequencing Algorithms Employed 34
4.3 Computational Complexity of Bind and Rollout Algorithm 34
4.4 Proficiency of Bind and Rollout Algorithm 35
4.5 An Example 36

5 Computational Results **39**

6 Conclusion and Extensions **45**

Appendix **49**

A Failure Probability Computation **49**
A.1 Prior Probabilities of Failures 49
A.2 Conditional Failure Probabilities 49

List of Figures

1.1 Decision tree for a test sequencing problem 4

1.2 AND/OR binary decision tree for a test sequencing problem 5

1.3 An optimal test sequence for the example presented in Table 1.2 . . . 7

2.1 Optimum binary coding procedure 14

2.2 Optimum binary coding procedure: A different perspective 16

2.3 Test sequencing for example 2.4.1 16

2.4 Alternative test sequencing for example 2.4.1 17

3.1 Traditional approach vs. binding strategy 20

3.2 One path leading the optimal solution using binding strategy 21

3.3 An unallowable binding for the case in Table 3.1 23

3.4 Application of separation heuristic for the case in Table 3.3 26

3.5 Application of Huffman coding based algorithm for the case in Table 3.3 28

4.1 Binary decision tree constructed using modified bind and rollout algorithm 38

List of Tables

1.1 Symmetrical test vs. asymmetrical test 3

1.2 Example: Diagnostic dictionary matrix, fault probabilities and test costs 6

2.1 Results of optimum binary coding procedure 14

2.2 Tests required to use results of noiseless coding for Example 2.4.1 . . 15

2.3 Alternative tests required to use results of noiseless coding for the example 2.4.1 17

3.1 Disadvantage of using binding strategy 22

3.2 Appropriate form to apply binding strategy 24

3.3 An example for Huffman coding based algorithm 26

3.4 First allowance check for the case in Table 3.3 27

3.5 Second allowance check for the case in Table 3.3 27

3.6 Third allowance check for the case in Table 3.3 27

3.7 Computational results of modified Huffman coding 29

4.1 An example to show that Huffman coding based algorithm lacks a tool to deal with asymmetrical tests 31

4.2 An example with asymmetrical tests and non-uniform costs 36

4.3 First iteration of modified bind and rollout algorithm 37

4.4 The case when s_2 and $s_{2'}$ are temporarily bind 38

5.1 Size of test problems 40

5.2 Information heuristic, modified separation heuristic, random solution, and modified bind and rollout heuristic versus optimal solution 43

5.3 Average worst results 44

5.4 Average time spent for test problems 44

Chapter 1

Introduction

1.1 Motivation

In today's competitive world, complexity of systems are increasing rapidly as a result of growing demand on system performance and recent advances in very large-scale integration technology. However, in such a complex environment, the problem of maintaining and repairing these systems becomes even more difficult. The purpose of system maintenance is to keep the system running, and if the system fails, to diagnose and repair detected failures as quickly as possible.

Specifically, the goal of a diagnostic procedure is to identify the actual system state. The system is in a certain unknown state, before the diagnostic procedure. The diagnostic procedure identifies the actual state of the system by gathering information using available tests. Any measurement, observation, signal can be considered as an available test.

Our goal in this study is to develop an algorithm that exploits the (a priori) failure probabilities and test costs to construct efficient diagnostic procedures, to minimize the expected cost of diagnosis. Optimization of a diagnostic procedure, which is formally defined as a test sequencing problem in literature, is known to be an NP-complete problem [9].

The test sequencing problem belongs to the general class of binary identification problems that arise in a wide area of applications. Other than maintenance operations [1,8,11,17], such problems arise in botanical and zoological field of work, plant pathology, medical diagnosis, decision table programming, computerized banking, pattern recognition, nuclear power plant control [3, 9, 10], discriminant analysis of test data, reliability analysis of coherent systems, research and development plan-

ning (e.g., in allocation of limited funds among high-risk projects), communication networks, speech/voice recognition (e.g., in classification of pattern vectors), distributed computing, and in the design of interactive expert systems [2].

Next section of this chapter presents a formal definition of the test sequencing problem. Chapter 2 briefly explains the proposed solution approaches for the test sequencing problem and its variants in the literature. Chapter 3 gives the details of the test sequencing problem and our approach to the problem when test costs are uniform. In chapter 4 test costs are not uniform, and our new approach is discussed. Chapter 5 gives the details and results of the computational study. Chapter 6 includes the conclusion and possible extensions of the study.

1.2 Problem Definition

In this study, the test sequencing problem is considered in the following context. We are given [9, 11]:

1. a set of $m+1$ system states $S = \{s_0, s_1, s_2, \ldots, s_m\}$ associated with the system, where s_0 denotes the *fault-free* state of the system and s_i $(1 \leq i \leq m)$ denotes one of the m potential faulty states of the system;

2. the prior conditional probability vector of the system states $P = [p(s_0), \ldots, p(s_m)]^T$, where $p(s_0)$ is the conditional probability that no fault exists in the system and $p(s_i)$ $(1 \leq i \leq m)$ denotes the probability that system is in state i[1];

3. a set of n available reliable tests $T = \{t_1, t_2, \ldots, t_n\}$ with a cost vector $C = [c_1, c_2, \ldots, c_n]^T$, where c_j denotes the cost of applying test t_j, measured in terms of time, pain, manpower requirements, other economic factors, etc.;

4. a diagnostic dictionary matrix $D = [d_{ij}]$, where d_{ij} is 1 if test t_j detects a failure state s_i, and 0 otherwise.

The tests described above have binary outcomes, i.e., a test fails (outcome= 1) if it has detected a failure and passes otherwise (outcome= 0). In case of binary tests, two sets of system states are defined: one corresponding to the *fail* test outcome

[1]The techniques to compute these probabilities are explained in detail in Appendix A

(set A) and the other to the *pass* test outcome (set B). It is obvious that every system state has to be an element of at least one set (i.e., $A \cup B = S$, since the test always has a result). We distinguish between two types of tests [1], as follows:

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

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

In the case of asymmetrical test, there is at least one system state that remains on the candidate list regardless of the test outcome ($A \cap B \neq \emptyset \implies \exists s; s \in A \wedge s \in B$).

Diagnostic dictionary matrix for three different tests are shown in Table 1.1, where t_2 is an asymmetrical test. Note that when asymmetrical tests are included, values in diagnostic dictionary matrix may have any value $\in [0, 1]$. If t_1 is applied as the first test in the test sequence, the set of ambiguity will be divided into two distinct subsets, $A = \{s_1, s_2\}$ and $B = \{s_3, s_4\}$. On the other hand, if t_2 is applied as the first test, the set of ambiguity will be divided into two subsets, $A = \{s_1, s_3\}$ and $B = \{s_2, s_3, s_4\}$. In other words, when t_2 is applied, s_3 remains on the candidate list regardless of the outcome, therefore t_2 is an asymmetrical test. Equivalently, given that the system is in state s_3, the outcome of test t_2 is 1 with probability 0.4 and 0 with probability 0.6.

	Tests		
s	t_1	t_2	t_3
s_1	0	0	0
s_2	0	1	0
s_3	1	0.4	0
s_4	1	1	1

Table 1.1: Symmetrical test vs. asymmetrical test

A feasible solution for the test sequencing problem can be described as a binary decision tree where nodes correspond to the tests, arcs correspond to the outcome

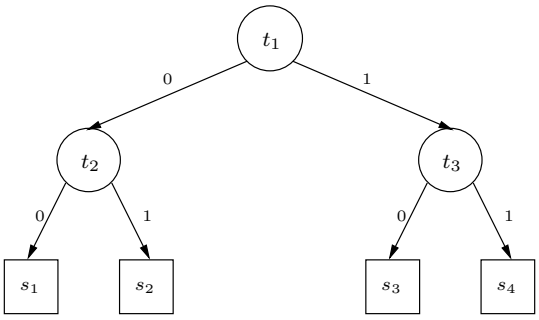

Figure 1.1: Decision tree for a test sequencing problem

of the tests, and leaves correspond to the actual system states. A feasible binary decision tree for the example in Table 1.1, is shown in Figure 1.1. For instance, tests t_1 and t_2 are used in the path leading to the identification of state s_1. In other words if the system is in state s_1, total cost to identify the system state is $(c_1 + c_2)$. Summing up over all states and tests, average cost of the decision tree is

$$J = p_1[c_1 + c_2] + p_2[c_1 + c_2] + p_3[c_1 + c_3] + p_4[c_1 + c_3]$$

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

$$J = p^T Ac = \sum_{i=0}^{m} \sum_{j=1}^{n} \alpha_{ij} p(s_i) c_j \tag{1.1}$$

(i.e., the average cost of the decision tree) is minimized [9]. In 1.1, $A = (\alpha_{ij})$ is an $(m + 1)$ by n matrix such that $\alpha_{ij} = 1$ if test t_j is used in the path leading to the identification of system state s_i and is 0 otherwise. An *optimal* diagnostic procedure is one which has the minimum cost over all diagnostic procedures which use tests from T to determine system state.

The problem above can be considered as a Markov decision problem (MDP), wherein the Markov state x denotes the suspect set of system states (also termed the ambiguity set), and the decision corresponds to the test to perform in state x [9]. The solution to the MDP is a deterministic AND/OR binary decision tree, with OR nodes labeled by ambiguity status x, AND nodes denoting tests(decisions) at OR nodes, and the weighted average length of the tree representing the expected

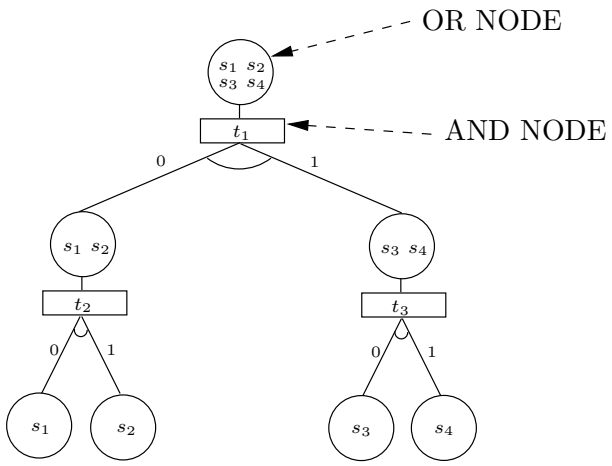

Figure 1.2: AND/OR binary decision tree for a test sequencing problem

test cost, J. However, the construction of the optimal decision tree is an NP-complete problem [3,9]. This study aims to develop a test sequencing algorithm with comparable results by integrating concepts from information theory and heuristic search to overcome the difficulties that appear as a result of problem complexity.

The corresponding AND/OR binary decision tree for the case represented in Figure 1.1 is shown in Figure 1.2.

Before proceeding an important distinction should be made. The heuristics developed and mentioned in this study are *not* classifiers. Instead, the heuristics develop a classifier. Thus, the heuristic is used only once for a given instance and the strategy produced by the heuristic will be used over and over by the user to classify as needed. This is why expected cost is minimized, instead of minimizing the number of tests to be used, maximum cost of identifying a state etc.

There are two major kinds of diagnostic procedures [17]: combinatorial and sequential. In a combinatorial procedure the sequence of tests to be executed is static (i.e., it does not depend on the result of previously executed tests). In a sequential procedure, the choice of the ith test to be executed is based on the results of the $(i-1)$ previously executed tests. Since in the combinatorial procedure all tests are always executed, the average time to diagnose a fault is higher than for a sequential procedure. In this study, we focused on **sequential** procedures.

1.3 An Example

The example presented in Table 1.2 is taken from [1] where there are five states and five tests. In this system, there are four faulty states s_1, s_2, s_3, s_4 and the fault-free state s_0. A set of five tests may be used to isolate the failure state. Tests t_2, t_4 and t_5 are symmetrical (binary values). Tests t_1 and t_3 are asymmetrical. In this example, test t_1 exhibits asymmetrical behavior only in the system state s_2 and test t_3 in the system state s_3. The value $d_{21} = 0.8$ means that the probability test t_1 fails is 0.8, if the system is in the state s_2.

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

Table 1.2: Example: Diagnostic dictionary matrix, fault probabilities and test costs

An optimal test sequence for the example presented in Table 1.2 is shown in Figure 1.3. This test sequence includes the asymmetrical test t_3 which gives an additional leaf (the system state s_3 resides in two leaves).

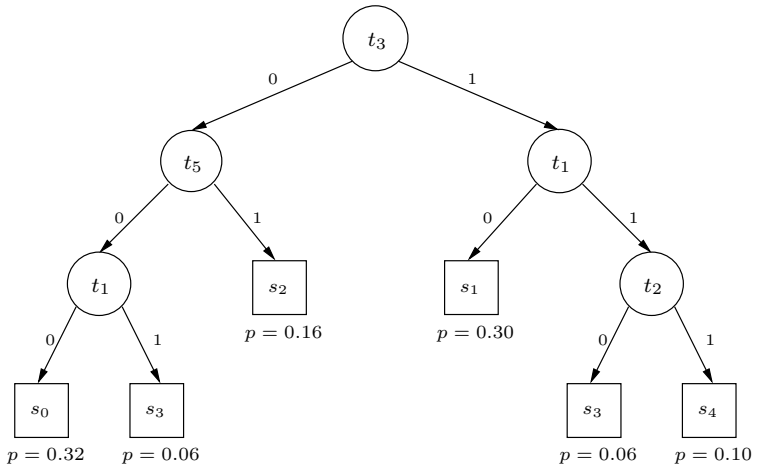

Figure 1.3: An optimal test sequence for the example presented in Table 1.2

Average cost for the optimal test sequence presented in Figure 1.3 is calculated as

$$J = 0.32[c_3+c_5+c_1]+0.06[c_3+c_5+c_1]+0.16[c_3+c_5]+0.30[c_3+c_1]+0.06[c_3+c_1+c_2]+0.10[c_3+c_1+c_2]$$

Chapter 2

Literature Review

The problem and its variations described in chapter 1 arise in various contexts in the literature, both for applied and theoretical considerations. Often, the researchers in one area have been unaware of the results that were obtained by researchers in other areas. In this and the following chapters, we shall bring together all these applications and results along with the results we have obtained.

2.1 Systems with Symmetrical Tests

Systems with symmetrical tests are simplest and most widely studied cases. However, even if we impose the additional constraint that tests have uniform costs, the problem turns out to be minimizing the expected number of tests, which is still NP-complete [12].

The existing solution approaches to the test sequencing problem can be categorized in two different groups: dynamic programming(DP), and "greedy" heuristics [9]. The bottom-up DP algorithm builds the optimal decision tree from the leaves up by identifying successively larger subtrees until the entire tree from the initial node of complete ambiguity is generated. The DP technique [3] has storage and computational requirements $O(3^n)$, and, hence, is impractical for large n, where n is the number of tests. Therefore, approximation techniques for constructing near-optimal decision trees are essential. Most of the traditional approximation techniques employ "greedy" heuristics; that is they perform a local, step-by-step optimization.

One of the earliest greedy heuristics is the information heuristic developed by Johnson [6]. In this algorithm a test t_k is selected in Markov state x, if it maximizes

the information gain per unit cost of the test:

$$k = \arg\max_j \{\frac{IG(x, t_j)}{c_j}\} \tag{2.1}$$

where $IG(x, t_j)$ is the information gain given by [6]:

$$IG(x, t_j) = -\{p(x_{jp}) \log_2 p(x_{jp}) + p(x_{jf}) \log_2 p(x_{jf})\} \tag{2.2}$$

where $\{x_{jp}, x_{jf}\}$ are the subsets of Markov state x corresponding to pass and fail outcomes of test t_j such that $x_{jp} \cup x_{jf} = x$, and $p(x_{jp})$, $p(x_{jf})$ are the conditional probabilities of the pass and fail outcomes of test t_j, respectively. Thus, the information heuristic is a one-step look-ahead procedure with computational complexity $O(mn)$.

Another similar heuristic[1] is the "separation heuristic" [4], where at each node of ambiguity a test t_k is selected that maximizes the distinguishability criterion $d_c(x, t_j)$ defined as

$$d_c(x, t_j) = p(x_{jp}) \cdot p(x_{jf}) \tag{2.3}$$

Varshney *et al.* [16] develops an algorithm for the construction of efficient sequential experiments. The approach in that construction is to minimize the upper bound at each step during the construction. A multistep look-ahead procedure similar to the information heuristic is used to derive these upper bounds.

The approach of Pattipati and Alexandridis [9] is based on integrating concepts from information theory and heuristic AND/OR graph search methods. AO* [7] is employed as the heuristic graph search method with three information theoretic HEFs[2] namely HEF_1: Huffman code length-based, HEF_2: Entropy-based, and HEF_3: Entropy+1 based functions. Using heuristics HEF_1 or HEF_2, AO* is guaranteed to find an optimal solution. HEF_3 does not always give an optimal solution but it is useful for complex examples which are intractable with HEF_1 and HEF_2.

A different approach is proposed by Fraughnaugh *et al.* in [2]. A number of different algorithms using various heuristic techniques including hillclimbing, random

[1]Information heuristic provides the same decision tree as the distinguishability criterion when test costs are equal [9].

[2]HEF: Heuristic Evaluation Function.

search, and tabu search are developed. A *generic decision rule* is used to determine which test to perform next in any state x:

$$(cost)^{\alpha}(|0.5 - probability|)^{\beta}(|0.5 - proportion|)^{\gamma} \tag{2.4}$$

Cost of a test, *probability* that a test fails, and *proportion* of unclassified states are values to use and the test that minimizes 2.4 is selected in any state x. Tabu search is used to find best values for decision variables α, β, and γ which affect the decision tree, thus cost of the policy.

Another interesting paper where the same problem comes up is by Raghavan *et al.* [11]. In this paper, various AO^* and information heuristic-based algorithms to solve test sequencing problem are developed. The major contribution in this work is that, a generalized test sequencing problem that incorporate various practical features such as precedence constraints, rectification etc. is considered. Rectification is the replacement of a potentially faulty component without prior diagnosis, i.e., the state is not known to be faulty for certain, but is replaced with a known good part for some rectification cost.

In [12], Raghavan *et al.* consider not only the test sequencing problem (i.e., how to determine a test sequence that minimizes expected testing cost), but three more problems as well; how to determine a test sequence that does not depend on the failure probability distribution, how to determine a test sequence that minimizes average ambiguity group size without using more than a number of tests, and how to determine a test sequence that minimizes the storage cost of tests[3] in the diagnostic strategy.

In [13], Shakeri *et al.* consider algorithms for multiple fault diagnosis. In multiple fault diagnosis problem, the system can be in *fault-free* state s_0, or in one of m potential faulty states s_i $(1 \leq i \leq m)$, or in any one of 2^m possible combination of failure states. Single fault strategy of their previous work [9, 11] is extended to diagnose multiple faults by successively isolating the potential single-fault candidates, then double-fault candidates, and so on.

Tu and Pattipati combine rollout algorithm with test sequencing heuristics in [14]. Rollout algorithm based on a heuristic test sequencing algorithm H, denoted

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

by RH, proceeds as follows:

The cost of constructing a test tree at a nondestination node[4] i is denoted by $H(i)$. Let $N(i)$ denote the set of immediate successor nodes of an OR node i, that is $N(i) = \{j|(i,j) \text{ is an arc}\}$ which contains all the tests available at node i. The rollout algorithm starts with the root node S and at any intermediate OR node i, RH adds to the test sequence a test j_{k+1} such that

$$j_{k+1} = \arg \min_{j \in N(i)} H_j(i) \tag{2.5}$$

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

2.2 Systems with Asymmetrical Tests

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

In [1], Biasizzo *et al.* prove that the same heuristics that have been employed in the traditional solution of the problem such as AO* algorithm with heuristics based on Huffman coding, can also be employed for the generalized case with asymmetrical tests. After numerous examples it is observed that AND/OR graph search algorithm pushes asymmetrical tests towards the leaves of the decision tree where they actually exhibit the symmetrical property.

In [17], Žužek *et al.* present the sequential diagnosis tool (SDT) that enables the user to generate solutions of the *generalized* test sequencing problem. The purpose of this study is to report this first sequential diagnosis software, that provide solutions to the generalized case including asymmetrical tests. The SDT reported in [17] includes both classes of algorithms: information heuristic algorithm and separation algorithm for fast generation of suboptimal solutions, and algorithm AO* for the generation of optimal solutions.

[4] Also termed the set of ambiguity, OR node.

2.3 Two Polynomial Time Cases

Optimal test algorithms with computational complexity $O(m \log m)$ can be designed for two extreme cases of the test sequencing problem [9].

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

$$J = \sum_{j=1}^{m} c_{[j]}[p(s_0) + \sum_{i=j}^{m} p(s_{[i]})] \tag{2.6}$$

The optimal test sequence is the priority rule

$$\frac{p(s_{[1]})}{c_{[1]}} \geq \frac{p(s_{[2]})}{c_{[2]}} \geq \ldots \geq \frac{p(s_{[m]}}{c_{[m]}} \tag{2.7}$$

On the other extreme, if all 2^m tests are available and the test costs are equal, the test sequencing problem is identical to Huffman coding problem; that is, the problem of generating the minimum redundancy, prefix free binary code of a set of binary messages for transmission over a noiseless channel [5]. Detailed information on noiseless coding problem and the analogy between the test sequencing and the noiseless coding problem is described below.

2.4 Noiseless Coding Problem

In noiseless coding problem, which is also referred to as *Huffman coding problem* or *the problem of constructing minimum-redundancy codes*, a coding scheme is constructed in such a way that the average number of coding digits per message is minimized.

Let us assume that there are k messages in our problem set $U = \{u_1, \ldots, u_k\}$ with the associated probability measure $P_U(u_k)$. The messages are to be encoded into binary[5] sequences for storage. In order to do that we associate a binary codeword a_k to each u_k. The set of all codewords $A = \{a_1, \ldots, a_k\}$ is known as a binary code. We constrain A to be uniquely decodable, i.e., for each finite message in U, the binary sequence corresponding to the encoding of this message is different from the binary

[5]In noiseless coding problem, there are D different types of symbols that can be used in coding. Since we are not dealing with multivalued tests, D is fixed to 2 to explain the analogy easier.

sequence corresponding to the encoding of any other message in U. Another basic restriction is that the message codes should be constructed in such a way that no additional indication is necessary to specify where a message code begins and ends once the starting point of a sequence of messages is known. That necessitates that codes should be prefix-free. A prefix-free code is a code in which no codeword is the prefix of any other codeword. The objective is to minimize the average storage which is equivalent to minimizing the average codeword length $\overline{W}$. This is defined as

$$\overline{W} = \sum_{k=1}^{K} W_k P_U(u_k) \tag{2.8}$$

where W_k is the length of the codeword a_k. Huffman [5] develops an optimum method of coding an emsemble of messages consisting of a finite number of members. For the binary case, this procedure is as follows.

Step 0: Designate K terminal nodes as $u_1, \ldots, u_K$ and assign probability $P_U(u_k)$ to node u_k for $k = 1, \ldots, K$. Consider these K nodes as *active*.

Step 1: Tie together the two least likely active nodes with a binary branch. Deactivate these two active nodes, activate the new node, and assign it a probability equal to the sum of the probabilities of the two nodes just deactivated.

Step 2: If now there is only one active node, then ground this node. Otherwise, go to Step 1.

Example 2.4.1 Suppose there are 6 messages to be coded $U = \{u_1, \ldots, u_6\}$, where $P_U(u_1) = 0.35$, $P_U(u_2) = 0.10$, $P_U(u_3) = 0.18$, $P_U(u_4) = 0.10$, $P_U(u_5) = 0.15$, and $P_U(u_6) = 0.12$. When we apply Huffman's procedure, the nodes are tied as in Figure 2.1 and the corresponding coding results are summarized in Table 2.1.

The left hand column in Figure 2.1 contains the ordered message probabilities of the ensemble to be coded where $K = 6$. Since each combination of two messages (indicated by a bracket) is accompanied by the assigning of a new digit to each, then the total number of digits which should be assigned to each original message is the same as the number of combinations indicated for that message. For example,

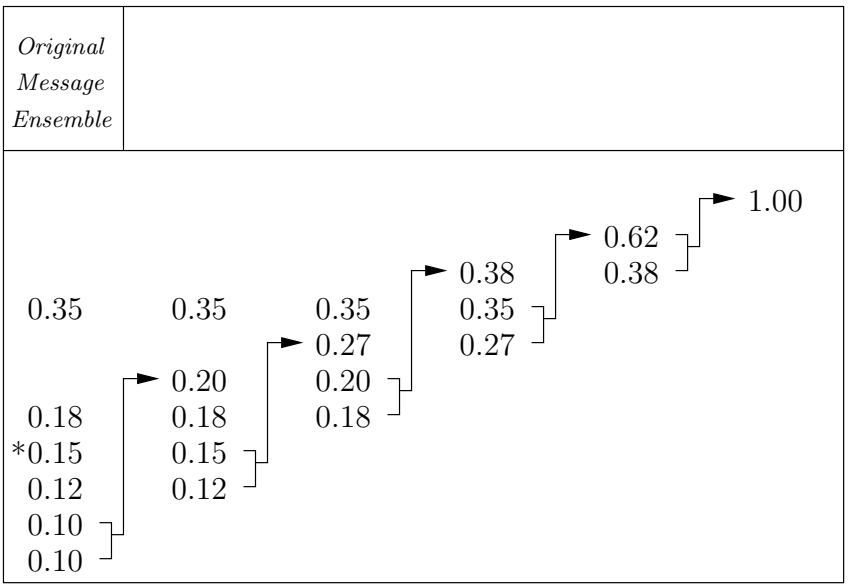

Figure 2.1: Optimum binary coding procedure

the message marked *, or a composite of which it is a part, is combined with others three times, and therefore should be assigned a code length of three digits.

When there is no alternative in choosing the two least probable messages, then it is clear that the requirements, established as necessary, are also sufficient for deriving an optimum code. There may arise situations in which a choice may be made between two or more groupings of least likely messages. No such case arises in example 2.4.1, however it is possible to rearrange codes in any manner among equally likely messages without affecting the average code length, and so a choice of either of the alternatives could have been made.

u	$P_U(u)$	$W(k)$	$W_k P_U(u_k)$	*Code*
u_1	0.35	2	0.70	11
u_2	0.10	3	0.30	011
u_3	0.18	2	0.36	00
u_4	0.10	3	0.30	010
u_5	0.15	3	0.45	101
u_6	0.12	3	0.36	100
			$\overline{W} = 2.47$	

Table 2.1: Results of optimum binary coding procedure

The code in Table 2.1 was obtained by using the digit 0 for the *lower* message and the digit 1 for the *upper* message. In Figure 2.1, when two least probable messages, u_2 and u_4, are tied together at first step. The last digit for u_4 is set to 0 and u_2 is set to 1. At second step, last digit of *lower* message, u_6, is set to 0 and last digit of *upper* message, u_5, is set to 1. At third step, last digit of *lower* message, u_3, is set to 0 but *upper* messages (i.e., u_2 and u_4) have already a last digit, so the digit before last are used and are set to 1. The algorithm proceeds that way and the coding results are shown in Table 2.1.

The analogy between the test sequencing and the noiseless coding problem is given in [11] as follows: the system states correspond to the binary messages, the sequence of test results are similar to the message code word, the average number of tests is characterized by the average length of code word, and the test sequencing algorithm is the coding scheme. The only differences are that the generation of a test algorithm is constrained by the availability of the tests, whereas no such constraint exists for the coding problem, and the tests may have unequal costs in the test sequencing problem.

	Tests		
	t_1	t_2	t_3
	Test Costs		
s	1	1	1
s_1	1	1	*
s_2	0	1	1
s_3	0	0	*
s_4	0	1	0
s_5	1	0	1
s_6	1	0	0

Table 2.2: Tests required to use results of noiseless coding for Example 2.4.1

In order to use results of noiseless coding problem we need availability of tests[6] shown in Table 2.2 for Example 2.4.1 and every test possible in general. Figure 2.1 can be modified as in Figure 2.2 to demonstrate the tests required easily. To sum up

[6] $*$ marks indicate that tests may have any value $\in [0, 1]$, i.e., may even be asymmetrical

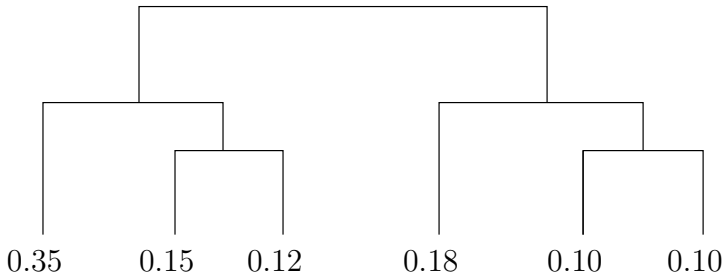

Figure 2.2: Optimum binary coding procedure: A different perspective

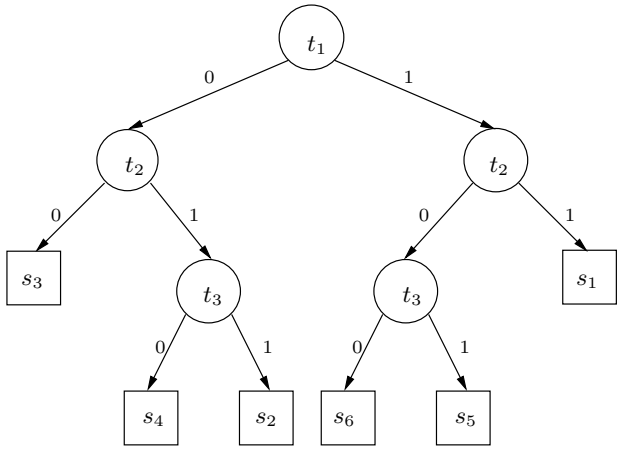

Figure 2.3: Test sequencing for example 2.4.1

we can say that, if there exist the desired columns (i.e., test results) in diagnostic dictionary matrix D with uniform test costs, we can use the results of Huffman coding procedure as in Figure 2.3. Note that Figure 2.2 and Figure 2.3 are actually performing the same operations at each node of ambiguity.

It is easy to notice that, tests required to use the results of noiseless coding are not unique, i.e., there may exist many combination of tests that have the same use. There is another alternative set of tests in Table 2.3 for which the diagnostic procedure will be as in Figure 2.4.

Several admissible HEF's are derived in [9] for the basic test sequencing problem by appealing to the analogy between the test sequencing and the Huffman coding problem.

	Tests				
	t_1	t_2	t_3	t_4	t_5
	Test Costs				
s	1	1	1	1	1
s_1	1	1	$*$	$*$	$*$
s_2	0	$*$	$*$	1	1
s_3	0	$*$	$*$	0	$*$
s_4	0	$*$	$*$	1	0
s_5	1	0	1	$*$	$*$
s_6	1	0	0	$*$	$*$

Table 2.3: Alternative tests required to use results of noiseless coding for the example 2.4.1

Property 1: The average conditional Huffman codeword length $w^*(x)$ for any node of ambiguity subset x provides a lower bound on the conditional average length, $1(x)$ of *any* test algorithm rooted at x (including the optimal test algorithm with length $1^*(x)$). Formally

$$w^*(x) \leq [\hat{p}(x)]^{(-1)} \sum_{s_i \in x} p(s_i)(\sum_{j=1}^{n} \alpha_{ij}(x)) \tag{2.9}$$

where $\alpha_{ij} = 1$ if test t_j is used by a test algorithm rooted at x to identify the system state s_i and is zero otherwise. Let G be an AND/OR graph. An HEF $h(x)$ defined

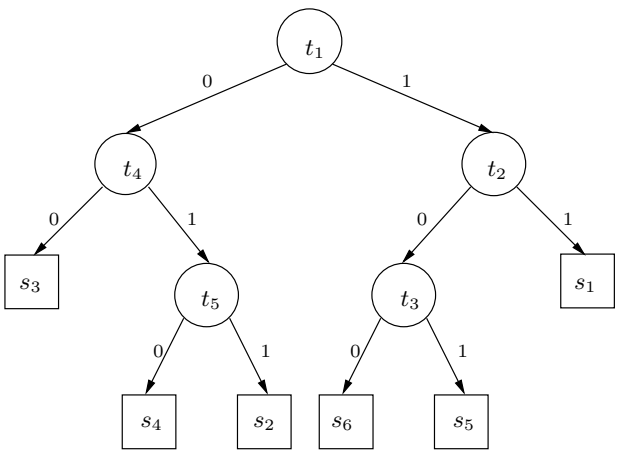

Figure 2.4: Alternative test sequencing for example 2.4.1

on the nodes of G is admissible if for each node x in G, $h(x) \leq h^*(x)$, the optimal cost-to-go. In particular, $h(x)$ can be infinite only if $h^*(x)$ is infinite. The above property of Huffman code can be used to derive an admissible HEF as mentioned before.

2.5 Systems with Multivalued Tests

Traditionally, we only consider binary outcome tests in the fault diagnosis problem. However, in practice, available diagnostic tests may exhibit significantly different behaviors. Generally, a test may have more than two possible outcomes. We call such test sets multivalued tests [15, 17, 18]. Basically, the algorithms used for a binary test system can also be applied to multivalued test systems. Although there are proposed algorithms in the literature [11, 17] that can handle multivalued tests, performance of algorithms in systems with multivalued tests may not be as high as in systems with binary outcome tests, since these are more generalized systems. Throughout this study we shall only deal with binary outcome tests because they arise more frequently.

Chapter 3

Tests with Uniform Costs

After the discussion of several proposed algorithms for sequential testing in chapter 2, it can be concluded that the need for *fast and near optimal* heuristics is of crucial importance. Greedy heuristics are used not only to obtain fast and near optimal results, but also to obtain upper bounds as in [16] or as HEFs in AND/OR graph search methods as in [9, 11].

There is a certain degree of analogy between Noiseless Coding Problem and Test Sequencing Problem that is discussed in section 2.4. However, it is well known that traditional test sequencing heuristics suggest *classification-separation* whereas Huffman's algorithm [5] is based on *tieing-binding together* sets of states which are two different perspectives.

3.1 A Note on Binding Strategy

In this section we shall provide a strategy for test sequencing problem that constructs the decision tree bottom-up. This strategy is naturally based on binding two sets of states at each iteration unlike the traditional methods that are based on dividing a set of ambiguity at each iteration. Constructing the decision tree upwards (i.e., beginning from actual states, ending with the set of all states, S) is similar to the idea of Huffman coding. We are not aware of any bottom-up based approach previously proposed for the test sequencing problem. Advantages and disadvantages of using a binding strategy can be described as follows:

One advantage of using binding strategy is that, it works fast. In the worst case there are m iterations which is the same as the traditional greedy heuristics. Second, and the most important, advantage of this strategy is that, unlike the

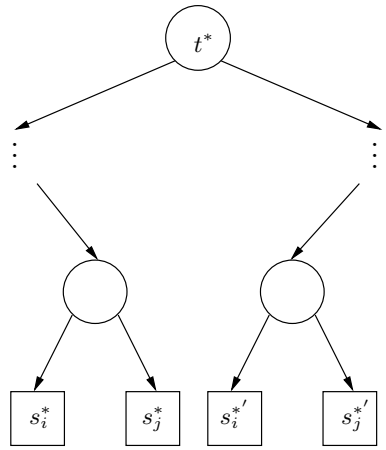

Figure 3.1: Traditional approach vs. binding strategy

traditional approach, there are usually more than one path leading to the optimal binary decision tree.

Suppose that the binary decision tree shown in Figure 3.1 is the optimal tree for a specific test sequencing problem. If a greedy heuristic based on traditional classification approach is employed, the algorithm should result with *perform test* t^* at the end of first iteration. On the other hand, if a heuristic based on binding strategy that constructs the tree upwards is employed, the algorithm may result with either *bind states* s_{i^*} *and* s_{j^*} or *bind states* $s_i^{*'}$ *and* $s_j^{*'}$ both of which have the possibility of constructing the optimal tree. The only situation that we can not make use of this useful property is the case when the only optimal binary decision tree is as shown in Figure 3.2.

The major disadvantage of using binding strategy is the additional computational work at each iteration. In traditional approach, at any node of ambiguity, any test is guaranteed to lead to a feasible solution, however, when an algorithm based on binding strategy is employed, binding any two sets of states may not lead to a feasible solution. In other words, *allowance* of binding decisions should be checked at each iteration, since they are not guaranteed to give feasible test sequences.

It is crucial to note that when an algorithm based on binding strategy is employed, as long as the problem is feasible at any time, there always exists an al-

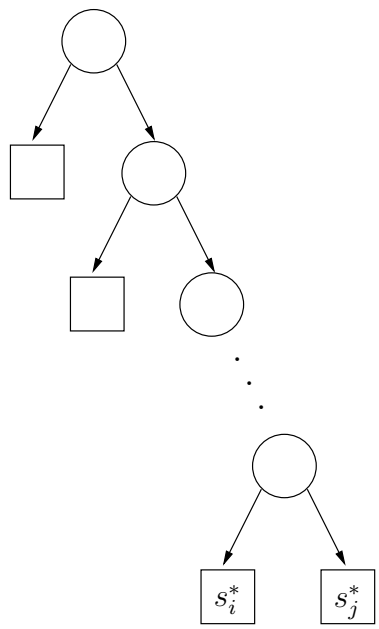

Figure 3.2: One path leading the optimal solution using binding strategy

lowable pair of sets of states to be bound together using tests available. Note also that, when defining the test sequencing problem, the standard assumption that the problem should be feasible, is imposed. These statements and the following lemma form a base for all of the proposed algorithms in this study.

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

Proof: Firstly, the term "feasibility" should be defined. At any stage, if it is possible to classify every state or sets of states using tests available, the problem is feasible.

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

If there exists such a feasible binary decision tree, intuitively every OR node in this tree is feasible. Also if there exists a feasible binary decision tree, then there exists at least one binding and a test that performs the binding at the bottom of the tree. Since every OR node in this tree is feasible, this binding is guaranteed to give a new feasible problem. □

Suppose the diagnostic dictionary matrix in Table 3.1 is the case. It is easy

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

Table 3.1: Disadvantage of using binding strategy

to see that the traditional approach may come up with any test at any iteration, and that will lead to a solution. Therefore, no test[1] is forbidden in any iteration. On the other hand, if an algorithm based on binding strategy is employed, that algorithm should not allow binding some sets of states in general. For instance, at first iteration binding s_1 and s_2 should not be allowed. If binding s_1 and s_2 is allowed and a new set is defined as $s_4 = s_1 \cup s_2$, the algorithm necessitates that all other states (in this case s_3 only) should be seperated from this new set s_4 (i.e., the problem should be feasible) before this binding, using any available test as in Figure 3.3. s_1 and s_2 may be classified using t_1, t_2, or t_3 all of which have different values for s_1 and s_2. This implies that none of these tests can be used in the following iterations that considers binding s_4 and any other set (i.e., test nodes above node T). At the same time it is necessary to bind s_4 and remaining states, which means that binding s_1 and s_2 leads to an infeasible result.

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

- Insert a new state, say s_k, where $p(s_k) = p(s_{i^*}) + p(s_{j^*})$.
- If $d_{i^*l} \neq d_{j^*l}$, set $d_{kl} = 2\ \forall l$, otherwise set $d_{kl} = d_{i^*l}$. Note that, a value of 2 in the diagnostic dictionary matrix implies that the test is not available in the

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

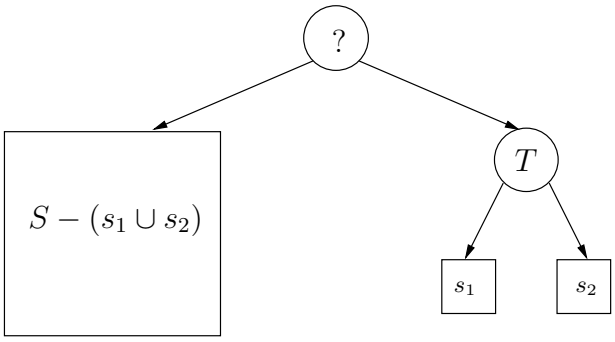

Figure 3.3: An unallowable binding for the case in Table 3.1

following steps.

- Delete rows corresponding to s_{i^*} and s_{j^*}.

Allowance check for a pair, s_{i^*} and s_{j^*}, is performed as follows:

Algorithm Allowance Check for i^* and j^*

INPUT: Diagnostic dictionary matrix $D = [d_{ij}]$, i^*, j^*.	
OUTPUT: Allowable or unallowable binding (i.e., 0 or 1)	
Step 0:	Construct a set of current states $S'' = S' = S \cup s_k - (s_{i^*} \cup s_{j^*})$. Temporarily update the diagnostic dictionary matrix for s_{i^*} and s_{j^*}.
Step 1:	Construct a set of available tests, $T' = \{t_l \mid d_{il} \neq 2,\ i \in S'\}$. If $T' = \emptyset \vee d_{il} = d_{jl}\ \forall i, j \in S'\ \forall l \in T'$ go to Step 4.
Step 2:	Using all tests in T' classify S' into subsets S_a.
Step 3:	For each subset S_a, set $S' = S_a$ and go to Step 1.
Step 4:	If there exists one set of state in S' or there exist sets of states all of which originally belong to the same state, then $S'' = S'' - S'$. Otherwise the binding is unallowable (i.e., output 0).
Step 5:	If $S'' = \emptyset$, the binding is allowable (i.e., output 1). Otherwise continue with other subsets.

After allowance check is completed, diagnostic dictionary martix is reverted back to its original condition. As discussed earlier, it is guaranteed that after an allowable binding, the problem is feasible, i.e., there exists at least one allowable pair of sets of states to be bound. To sum up, using the technique above at each iteration, one can construct feasible test sequences.

3.2 Inclusion of Asymmetrical Tests

When there exists asymmetrical tests, allowance check algorithm is applied similarly but the diagnostic matrix should be converted to the appropriate form. For the problem in Table 1.2, the appropriate form is shown in Table 3.2.

	Tests					
	t_1	t_2	t_3	t_4	t_5	
	Test costs c_j					
s_i	1	1	1	1	1	$p(s_i)$
s_0	0	0	0	1	0	0.32
s_1	0	0	1	1	0	0.30
s_2	0	0	0	1	1	0.032
$s_{2'}$	1	0	0	1	1	0.128
s_3	1	0	0	0	0	0.06
$s_{3'}$	1	0	1	0	0	0.06
s_4	1	1	1	1	0	0.10

Table 3.2: Appropriate form to apply binding strategy

The expression "sets of states all of which originally belong to the same state" in allowance check algorithm implies that, when $S' = \{s_2, s_{2'}\}$ in Step 4, although there are two states in the set, since they originally belong to the same state s_2, it does not ruin the allowance of the binding. In other words, the decision tree does not have to classify a set, consisting of s_i variants (i.e., s_i, $s_{i'}$, $s_{i''}$ and so on), further.

3.3 A Heuristic Based on Huffman Coding

To benefit from the advantages of binding strategy, *good* and *allowable* decisions should be made. When the tests have uniform costs, the logic behind Huffman coding [5] can be used in binding procedure. We employed the following technique[2] based on Huffman coding, which is known to give optimal solutions when 2^m tests are available[3], where m is the number of states, as discussed in section 2.3.

Algorithm Modified Huffman Coding

INPUT: Diagnostic dictionary matrix $D = [d_{ij}]$, prior conditional probability vector P.	
OUTPUT: Binary decision tree.	
Step 0:	Construct a set of current states $S_c = S$.
Step 1:	For every pair of sets of states i and j in S_c, compute $p_{ij}^{sum} = p_i + p_j$, and mark every pair.
Step 2:	Choose a marked pair with smallest p_{ij}^{sum} value.
Step 3:	Perform allowance check for i and j. If binding i and j is allowable, go to Step 4. Otherwise unmark the pair and go to Step 2.
Step 4:	Set $i^* = i$ and $j^* = j$.
Step 5:	Permanently insert a new state, say s_k, where $p(s_k) = p(s_{i^*}) + p(s_{j^*})$.
Step 6:	If $d_{i^*l} \neq d_{j^*l}$, set $d_{kl} = 2$, otherwise set $d_{kl} = d_{i^*l}$.
Step 7:	Permanently delete rows corresponding to s_{i^*} and s_{j^*}.
Step 8:	Set $S_c = S_c \cup s_k - (s_{i^*} \cup s_{j^*})$. If S_c consists of one state TERMINATE ALGORITHM, otherwise go to Step 1.

[2]It is crucial to note again that there always exists at least one allowable binding (i.e. marked pair in the algorithm) at any iteration if the initial problem is feasible. See Lemma 1 for details.

[3]Note that, in this case every pair of sets of states to be bound at any iteration is allowable, i.e., at any iteration $\exists l$, s.t. $d_{il} + d_{jl} = 1 \; \forall i, j$.

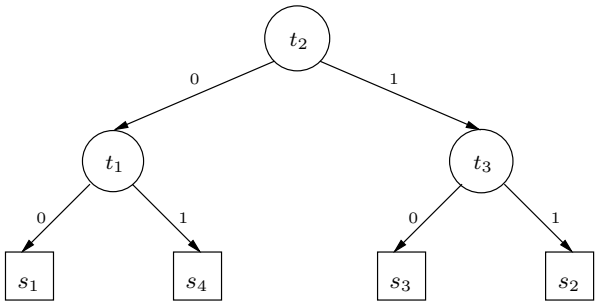

Figure 3.4: Application of separation heuristic for the case in Table 3.3

3.4 An Example

		Tests				
		t_1	t_2	t_3	t_4	t_5
		Test Costs				
s	$p(s)$	1	1	1	1	1
s_1	0.4	0	0	0	0	0
s_2	0.248889	0	1	1	0	1
s_3	0.328889	1	1	0	1	1
s_4	0.022222	1	0	1	0	0

Table 3.3: An example for Huffman coding based algorithm

When separation heuristic is employed for the problem presented in Table 3.3 the binary decision tree is formed as in Figure 3.4 which gives a cost of 2.

However when our Huffman code based algorithm is employed, it gives a cost of 1.942222 and the algorithm proceeds as follows:

$S_c = \{s_1, s_2, s_3, s_4\}$. Choose s_2 and s_4 with $p_{24}^{sum} = 0.271111$. To perform allowance check, algorithm allowance check is applied, and the diagnostic dictionary matrix is updated temporarily as in Table 3.4

t_3 and t_4 are applied immediately and each subset consists of one state. Therefore the problem in Table 3.4 is feasible, and the binding is allowable.

S_c is updated as $S_c = \{s_1, s_3, s_5\}$ and Table 3.4 becomes permanent. Choose s_3 and s_5 with $p_{35}^{sum} = 0.6$. Allowance check should be performed again, and the diagnostic dictionary matrix is updated temporarily as in Table 3.5

		Tests				
		t_1	t_2	t_3	t_4	t_5
		Test Costs				
s	$p(s)$	1	1	1	1	1
s_1	0.4	0	0	0	0	0
s_3	0.328889	1	1	0	1	1
s_5	0.271111	2	2	1	0	2

Table 3.4: First allowance check for the case in Table 3.3

		Tests				
		t_1	t_2	t_3	t_4	t_5
		Test Costs				
s	$p(s)$	1	1	1	1	1
s_1	0.4	0	0	0	0	0
s_6	0.6	2	2	2	2	2

Table 3.5: Second allowance check for the case in Table 3.3

No test can be applied and the subset $S' = \{s_1, s_6\}$ does not consist of one state. Hence the problem in Table 3.5 is infeasible, and binding s_3 and s_5 is not allowed. Unmark that pair and choose the next best pair. s_1 and s_5 are chosen with $p_{15}^{sum} = 0.671111$, and the diagnostic dictionary matrix is updated temporarily as in Table 3.6

		Tests				
		t_1	t_2	t_3	t_4	t_5
		Test Costs				
s	$p(s)$	1	1	1	1	1
s_3	0.328889	1	1	0	1	1
s_6	0.671111	2	2	2	0	2

Table 3.6: Third allowance check for the case in Table 3.3

t_4 is applied and each subset consists of one state. Therefore the problem in Table 3.6 is feasible, and the binding is allowable. Finally s_3 and s_6 are bound

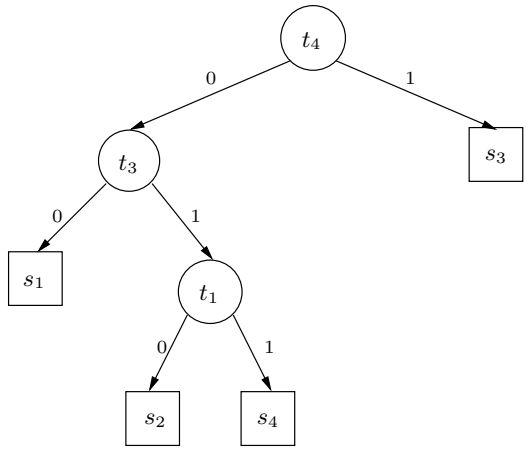

Figure 3.5: Application of Huffman coding based algorithm for the case in Table 3.3

together and the algorithm terminates.

The binary decision tree formed by the algorithm is shown in Figure 3.5

The average cost of the decision tree in Figure 3.5 is

$$\begin{aligned} J &= p(s_1)[c_3 + c_4] + p(s_2)[c_1 + c_3 + c_4] + p(s_3)[c_4] + p(s_4)[c_1 + c_3 + c_4] \\ &= 0.4[2] + 0.248889[3] + 0.328889[1] + 0.022222[3] \\ &= 1.942222 \end{aligned}$$

3.5 Computational Results of Modified Huffman Coding

Computational results of modified Huffman coding is presented in Table 3.7. Those results encouraged us to introduce a new algorithm based on binding strategy for the case where test costs are non-uniform.

The details of problem sets such as probability distribution of system states, number of runs, etc. are described in chapter 5. Note that, the results presented in Table 3.7 are average results for each case and test costs are uniform.

Number of states	Number of tests	Number of optimal results achieved	Average Percentage Error
6	4	29/40	1.006
10	8	11/40	3.291
15	10	3/40	3.894
20	12	3/40	4.407
30	15	0/40	4.287

Table 3.7: Computational results of modified Huffman coding

Chapter 4

Tests with Non-Uniform Costs

A powerful tool to construct a decision tree for test sequencing problem is introduced in section 3.3 for the case when costs are uniform. However, in general, when it is considered to bind two sets of states, cost of binding those two states as well as probabilities of those sets should be considered. Since the algorithm described in chapter 3 performed well in the case with uniform costs, it is adapted to the case where test costs are non-uniform, and this time the algorithm will also take "cost of tests" into account.

In this chapter, a basic algorithm is introduced that is applicable in the general case where test costs are not uniform. Similar to the Huffman coding based algorithm, the binary decision tree is constructed *bottom-up* with *binding strategy* using allowance check algorithm described in section 3.1. At each iteration inputs are probabilities of sets of states and minimum[1] cost of binding sets of states.

Note that, in general Huffman coding based algorithm neither gives optimal results nor has any tool to deal with asymmetrical tests. When it is applied to the problem in Table 4.1, the algorithm will first bind $s_{1'}$ and s_2 with $p_{1'2}^{sum} = 0.4$, which is an allowable binding. The algorithm will obviously bind s_1 at the next iteration so the algorithm will result an average cost > 1, whereas the optimal solution gives a cost of 1. Performing only t_1 leads to the optimal solution because of the special structure of the problem.

The algorithm based on Huffman coding deals only with probabilities, since every binding (i.e., performing a test) costs 1 similar to the noiseless coding problem[2].

[1]When two sets are considered to be bind together, and there exist multiple tests that can perform binding operation, intuitively the test with minimum cost should be specified.

[2]Binding i and j increments the objective function by $(p_i + p_j) \cdot 1$ in both problems

		Tests	
		t_1	t_2
		Test Costs	
s	$p(s)$	1	1
s_1	0.8	0	0.75
s_2	0.2	1	0

		Tests	
		t_1	t_2
		Test Costs	
s	$p(s)$	1	1
s_1	0.6	0	1
s_2	0.2	1	0
$s_{1'}$	0.2	0	0

Table 4.1: An example to show that Huffman coding based algorithm lacks a tool to deal with asymmetrical tests

However in the example presented in Table 4.1, binding s_1 and $s_{1'}$ is "free".

Turning back to the discussion of tests with non-uniform costs, the algorithm should consider cost of binding as well as probabilities of sets of states. A generic algorithm that deals with cost of binding, can also deal with asymmetrical tests, which is nothing but a zero cost test in certain iterations.

4.1 A Heuristic Based on Binding and Rollout Strategies

The heuristic described in this section is based on binding and rollout strategies. At each iteration, allowance check algorithm described in section 3.1 is applied to every pair of sets of states, and the allowable pairs are determined to be bound together. Among this candidate set, an average cost Ω_{ij}^{H} is estimated for each pair via a heuristic test sequencing algorithm, H, and the most promising selection is made which is similar to the *rollout* strategy of [14]. Heuristic test sequencing algorithms employed in the algorithm are discussed in section 4.2. At each iteration Ω_{ij} is calculated as

$$\Omega_{ij}^{H} = (p_i + p_j)u_{ij} + H_{i,j}^{*} \tag{4.1}$$

u_{ij} in equation 4.1 is the test with minimum cost that can bind sets i and j. If sets i and j belong to the same state, $u_{ij} = 0$, otherwise

$$u_{ij} = \min_{k}\{c_k | d_{ik} + d_{jk} = 1\} \tag{4.2}$$

The term $(p_i + p_j)u_{ij}$ in equation 4.1 gives the exact cost of binding i and j and $H_{i,j}^{*}$ gives an estimate for the cost the decision tree excluding the test used for classifying i and j. This operation is simply performed by binding i and j, updating diagnostic dictionary matrix, which is guaranteed to be feasible since binding i and j is allowable, and applying a heuristic test sequencing algorithm, H for an average cost estimate. Note that, the heuristic test sequencing algorithm is applied in traditional way (i.e., tree is constructed normally, from top to bottom after i and j are strictly bound together).

Algorithm Modified Bind and Rollout

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

4.2 Heuristic Test Sequencing Algorithms Employed

Four greedy heuristics are employed to obtain an estimate in our algorithm. The algorithm is applied from top to down and the set of ambiguity is represented by S^O at each step. Obviously a test t_k where $d_{ik} = 2$ for any $i \in S^O$ is not evaluated, since it is not applicable to that set of ambiguity. The heuristics used are:

1. Information heuristic developed by Johnson [6].
2. Modified separation heuristic [4]: A test t_k is selected at each step, where $k = \arg\min_j \{\frac{c_j}{p(x_{jp}) \cdot p(x_{jf})}\}$.
3. Cheapest test: A useful[3] test t_k is selected at each step, where $k = \arg\min_j \{c_j\}$.
4. New greedy algorithm: A test t_k is selected at each step, where $k = \arg\min_\alpha \{\frac{c_\alpha}{\lambda_\alpha}\}$. $\lambda_\alpha = \sum_{d_{i\alpha}+d_{j\alpha}=1 \wedge i,j \in S^O} (p_i + p_j)$.

4.3 Computational Complexity of Bind and Rollout Algorithm

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

- at most $\begin{pmatrix} m \\ 2 \end{pmatrix}$ allowance checks
- at most $\begin{pmatrix} m \\ 2 \end{pmatrix}$ heuristic test sequencing algorithm, H implementations
- selection of the minimum of these $\begin{pmatrix} m \\ 2 \end{pmatrix}$ estimates

Suppose that the complexity of heuristic test sequencing algorithm employed is $O(H)$ and allowance check algorithm is $O(AC)$. The the computational complexity of bind and rollout algorithm, $O(B\&R)$ is calculated as

$$O(B\&R) = O(m(m^2 O(AC) + m^2 O(H) + m^2)) = O(m^3(O(AC) + O(H))) \quad (4.3)$$

Allowance check algorithm consists of m iterations in the worst case and at each iteration availability of tests are checked. Test availability is essentially checked by

[3] A test is useful if it performs seperation for the OR node in consideration (i.e., the values in the diagnostic dictionary matrix are not the same for all states in that OR node).

looking up the entries of an $m \times n$ matrix. Hence the complexity of this availability check is $O(mn)$.

$$O(AC) = O(m^2 n) \tag{4.4}$$

and plugging into Equation 4.3

$$O(B\&R) = O(m^5 n + m^3 O(H)) \tag{4.5}$$

4.4 Proficiency of Bind and Rollout Algorithm

***Lemma* 2** When modified bind and rollout algorithm is used via any heuristic test sequencing algorithm H, one would not obtain a worse solution than when H is applied on its own.

Proof: Suppose that T_H is the binary decision tree constructed via heuristic test sequencing algorithm H and the average cost is J^H. Bind and rollout algorithm starts to construct the decision tree, $T_{B\&R(H)}$, bottom-up via heuristic test sequencing algorithm H. Note that at any iteration

$$J^H = (\text{Total cost of bindings performed up to that iteration}) + \Omega_{ij}^H \tag{4.6}$$

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

Bind and rollout algorithm calculates Ω values for each pair and therefore $T_{B\&R(H)}$ deviates from T_H at an iteration only if

$$\Omega_{i^*j^*}^H \leq \Omega_{ij}^H \tag{4.7}$$

where bind and rollout algorithm proposes to bind i^* and j^* and H proposes to bind i and j. Suppose that J' is the cost of bindings performed up to that iteration. Using Equation 4.7

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

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

and from Equation 4.6

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

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

Since Ω gives an upper bound for the upcoming cost in bind and rollout algorithm

$$J^{B\&R(H)} \leq \Omega^H_{i^*j^*} + J' \leq J^H$$

and finally

$$J^{B\&R(H)} \leq J^H \tag{4.8}$$

□

4.5 An Example

	Tests					
	t_1	t_2	t_3	t_4	t_5	
	Test Costs					
s	30	28	28	30	21	$p(s)$
s_1	0	1	1	1	1	0.113111
s_2	0	0	0	0	1	0.123393
s_3	1	0	1	1	1	0.179949
s_4	0	1	0	1	0	0.241645
s_5	0	1	1	0	1	0.0462725
$s_{2'}$	0	0	0	0	0	0.107969
$s_{1'}$	1	0	1	0	1	0.187661

Table 4.2: An example with asymmetrical tests and non-uniform costs

When modified bind and rollout algorithm is applied to the case in Table 4.2, binding allowance of every combination of set pairs and the resulting Ω estimates, are calculated as in Table 4.3. New greedy algorithm is employed as heuristic test sequencing algorithm. For example $\Omega^H_{2,2'}$ is calculated as

$$\Omega^H_{2,2'} = (p_2 + p_{2'})0 + H^*_{2,2'}$$

$H^*_{2,2'}$ is the average cost of the decision tree when new greedy test sequencing algorithm, H is applied to the problem in Table 4.4. Diagnostic dictionary matrix and probabilities in Table 4.4 are constructed by temporarily binding s_2 and $s_{2'}$.

In the case represented in Table 4.4, test t_5 is not available as the first test, since there exists a 2 in the corresponding column. For all tests except t_5, $\frac{c_\alpha}{\lambda_\alpha}$ is calculated

i,j	Allowance Check	Ω_{ij}^H
$1,2$	-	∞
$1,3$	$\surd$	73.9769
$1,4$	$\surd$	76.8586
$1,5$	$\surd$	74.0283
$1,2'$	-	∞
$1,1'$	-	∞
$2,3$	-	∞
$2,4$	-	∞
$2,5$	$\surd$	74.5630
$2,2'$	$\surd$	69.0026

i,j	Allowance Check	Ω_{ij}^H
$2,1'$	$\surd$	77.2211
$3,4$	-	∞
$3,5$	-	∞
$3,2'$	-	∞
$3,1'$	$\surd$	78.5578
$4,5$	-	∞
$4,2'$	$\surd$	74.0283
$4,1'$	-	∞
$5,2'$	-	∞
$5,1'$	$\surd$	75.0617
$2',1'$	-	∞

Table 4.3: First iteration of modified bind and rollout algorithm

and the test with minimum value is selected. As an example, at the beginning $S^O = \{s_1, s_3, s_4, s_5, s_{1'}, s_6\}$ and λ_1 is calculated as $[p(1) + p(3)] + [p(1) + p(1')] + [p(3)+p(4)]+[p(3)+p(5)]+[p(3)+p(6)]+[p(4)+p(1')]+[p(5)+p(1')]+[p(1')+p(6)]$. After best test is selected, same greedy heuristic is applied for all nodes of ambiguity.

At the end of the first main iteration states s_2 and $s_{2'}$ are permanently bind since it has the minimum Ω_{ij}^H value. The algorithm proceeds with the same logic and results with the binary decision tree shown in Figure 4.1, which is the optimal solution. Note that, it is just a coincidence that the optimal test sequence has the same average cost with the estimated cost at the first iteration.

	Tests					
	t_1	t_2	t_3	t_4	t_5	
	Test Costs					
s	30	28	28	30	21	$p(s)$
s_1	0	1	1	1	1	0.113111
s_3	1	0	1	1	1	0.179949
s_4	0	1	0	1	0	0.241645
s_5	0	1	1	0	1	0.0462725
$s_{1'}$	1	0	1	0	1	0.187661
s_6	0	0	0	0	2	0.225362

Table 4.4: The case when s_2 and $s_{2'}$ are temporarily bind

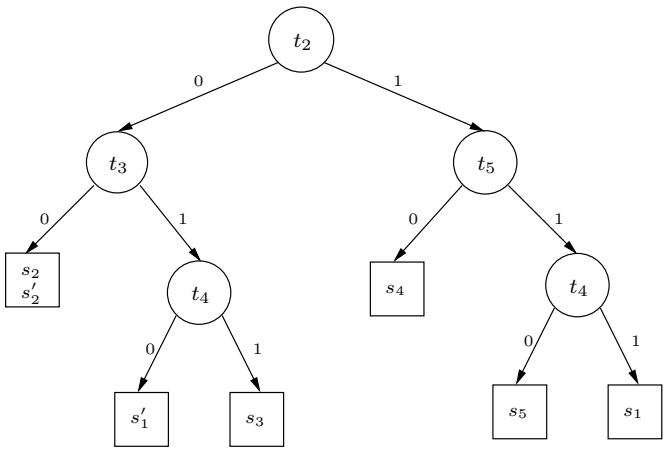

Figure 4.1: Binary decision tree constructed using modified bind and rollout algorithm

Chapter 5

Computational Results

In this chapter, the results of experiments are reported. The experiments are conducted with 3 heuristics; information heuristic developed by Johnson [6], modified separation heuristic [4], and modified bind and rollout described in section 4.1. The optimal solution is calculated using enumeration. Additionally a random solution is found for benchmarking.

To obtain a standard set of problems for comparison of our heuristic, we developed a problem generator to generate random problem instances of various sizes. The test bed of problems consists of a set of small problems and a set of larger problems, whose sizes are shown in Table 5.1. Problems, whose number of classes are marked with *, includes asymmetrical tests[1]. The test bed we have used consists of problems from three categories:

1. The first category is a set of problems with costs equal to 1. [PROB1-PROB9]

2. The second category is a set of problems with costs ranging from 25 to 30. [PROB11-PROB19]

3. The third category is a set of problems with costs ranging from 20 to 30. [PROB21-PROB29]

Number of states in each problem set are especially larger than the number of tests on purpose. The main reasons, why these types of problems are considered, are:

[1] In all experiments with asymmetrical tests, there exist $1.2(m+1)$ rows and $(m+1)$ states.

Problems	Number of Classes	Number of Tests
PROB1, PROB11, PROB21	6	4
PROB2, PROB12, PROB22	8	4
PROB3, PROB13, PROB23	10	5
PROB4, PROB14, PROB24	15	8
PROB5, PROB15, PROB25	20	10
PROB6, PROB16, PROB26	25	15
PROB7, PROB17, PROB27	10*	6
PROB8, PROB18, PROB28	25*	15
PROB9, PROB19, PROB29	30*	20

Table 5.1: Size of test problems

• When there exist fewer tests, random solutions might give comparable results and reach even the optimal solution for certain runs, because there are not so many combinations of test sequences.

• When there exist fewer tests, it is harder for the heuristics to achieve good solutions, because the flexibility of the heuristic reduces. In other words, once the heuristic fails to find the correct move at any iteration, it is hard to compensate it.

• When there exist fewer tests, enumeration can be completed in reasonable time.

We can conclude that, if the same study is performed with more tests, solution quality of the random solution will decrease, time spent for enumeration will increase, and heuristics will be more applicable. Hence, the conditions presented here can be easily modified to obtain better results.

For each problem set, 10 random runs are performed where state probabilities are uniform (i.e., $p(s_i) = \frac{1}{m+1}\ \forall i$). Secondly, 10 random runs are performed where state probabilities are uniformly distributed in the interval $(0.3, 0.7)$. Thirdly, 10 random runs are performed where state probabilities are uniformly distributed in the interval $(0, 1)$. Finally, 10 random runs are performed where probabilities of half of the states are uniformly distributed in the interval $(0.1, 0.2)$ and other half of the states are uniformly distributed in the interval $(0.8, 0.9)$. Therefore each problem set consists of 40 independent runs and their averages are reported as results in tables

5.2, 5.3, and 5.4. Only PROB9, PROB19, and PROB29 consist of 8 runs (i.e., 2 runs for each probability distribution) since enumeration of these set of problems is time consuming.

In Table 5.2, OPT is the number of times the heuristic reached the optimal solution among 40 runs, MAX is the maximum percentage error of heuristic, and AVG is the average percentage error of heuristic. In Table 5.3, average worst results for each problem set are presented, which are obtained by inverting the objective function and using the same enumeration technique used in finding the optimal solution. Average worst results are calculated to demonstrate the experimental conditions. In other words, if average worst results are $(1 + b)$ optimal, a heuristic that gives $(1+a)$ optimal results on the average can not be considered as sufficiently approximate, as long as b is much larger than a. For example, an algorithm that gives 10% error on the average for a certain problem set may seem sufficiently approximate. However if we know that, average worst result for the problem set is $1.12 \cdot optimal$, we may not consider the same results as sufficiently approximate and look for improvements.

Note that, in modified bind and rollout heuristic, four greedy heuristic test sequencing algorithms are employed for estimation and the one that gives the best result is reported in Table 5.2. Since modified bind and rollout is a polynomial time heuristic, although the operation is performed four times using different greedy heuristics, the algorithm was not time consuming as seen in Table 5.4[2]. When costs are uniform, modified Huffman coding algorithm described in section 3.3 is applied and afterwards modified bind and rollout algorithm is run via two different greedy heuristics: information heuristic[3], and new greedy heuristic. On the other hand, when cost are non-uniform, modified bind and rollout algorithm is run via four different greedy heuristics: information heuristic, modified separation heuristic, new greedy heuristic, and selecting the test with minimum cost and the best results obtained are reported as the results of main algorithm in Table 5.2.

As a conclusion, the bind and rollout algorithm is mostly suitable for middle-large

[2]Computer used in this study has Celeron 1.33 Ghz CPU and 256 MB RAM and the algorithm is coded in C++.

[3]When costs are uniform, there is no need to use modified separation heuristic because information heuristic gives the same result [9].

sized problems, where number of tests is not small enough to perform enumeration. For instance, when 4 runs are performed for 20 states with 25 asymmetrical tests[4], it takes 30332.8 seconds to perform complete enumeration on the average. The average percentage error for the bind and rollout algorithm is 0.997 whereas the average percentage error for other one step heuristics is 10.62.

[4]In this set of experiments there exist $1.5(m+1)$ rows and $(m+1)$ states. 1 run is performed for each probability distribution and test costs are uniform.

	Random Solution			Modified Bind and Rollout Heuristic			Modified Separation Heuristic			Information Heuristic		
Problems	OPT	AVG	MAX	OPT	AVG	MAX	OPT	AVG	MAX	OPT	AVG	MAX
PROB1	7	8.295	69.411	40	0	0	33	0.392	5.914	33	0.392	5.914
PROB2	7	7.332	31.646	40	0	0	29	0.523	6.362	29	0.523	6.362
PROB3	1	6.831	21.607	38	0.024	0.58	21	0.877	7.544	21	0.877	7.544
PROB4	1	9.25	21.282	30	0.113	1.692	10	1.95	7.816	10	1.95	7.816
PROB5	1	9.117	19.599	27	0.108	1.309	15	1.032	4.824	15	1.032	4.824
PROB6	0	9.708	22.614	29	0.101	0.914	11	1.127	4.289	11	1.127	4.289
PROB7	1	10.72	21.338	37	0.024	0.513	13	2.796	11.04	13	2.796	11.04
PROB8	0	12.756	25.525	20	0.424	2.797	0	5.176	11.014	0	5.176	11.014
PROB9	0	15.198	30.605	4	0.336	1.093	0	5.787	8.971	0	5.787	8.971
PROB11	4	8.895	38.957	40	0	0	26	1.041	11.573	22	1.305	11.573
PROB12	1	6.665	22.104	38	0.082	2.914	26	0.71	6.196	20	1.249	6.196
PROB13	0	9.043	24.205	38	0.014	0.363	20	0.649	6.966	14	1.525	8.792
PROB14	0	11.495	33.158	31	0.128	1.866	10	1.362	7.14	3	2.475	7.48
PROB15	0	13.133	28.566	27	0.044	0.385	6	1.033	3.537	3	1.967	4.849
PROB16	0	16.368	31.48	27	0.076	0.758	2	1.149	3.877	3	1.936	5.148
PROB17	0	13.308	29.68	38	0.021	0.761	9	3.087	9.896	3	3.519	13.496
PROB18	0	17.433	32.236	13	0.255	1.779	0	2.964	7.207	0	4.526	10.812
PROB19	0	18.179	21.737	4	0.26	1.197	0	3.905	6.321	0	4.348	6.28
PROB21	2	9.706	26.602	40	0	0	28	0.578	6.646	25	1.029	8.861
PROB22	1	7.983	22.325	38	0.026	0.591	26	0.781	4.22	21	1.422	7.03
PROB23	0	10.03	20.475	38	0.034	1.172	14	1.18	4.584	6	2.361	6.966
PROB24	0	16.053	40.992	33	0.053	0.735	9	0.999	4.08	3	2.908	9.116
PROB25	0	17.56	34.329	25	0.141	1.645	5	1.099	4.681	1	2.915	8.718
PROB26	0	22.741	37.278	20	0.088	0.529	4	0.867	3.015	0	2.657	6.274
PROB27	0	15.451	36.083	36	0.052	1.437	4	3.8	14.513	2	4.824	14.596
PROB28	0	24.618	37.743	16	0.162	0.814	2	2.027	5.803	0	3.947	7.611
PROB29	0	26.399	38.359	1	0.46	1.366	0	2.626	4.554	0	3.94	6.836

Table 5.2: Information heuristic, modified separation heuristic, random solution, and modified bind and rollout heuristic versus optimal solution

Problems	WORST
PROB1	1.18448 · *optimal*
PROB2	1.14351 · *optimal*
PROB3	1.17998 · *optimal*
PROB4	1.29119 · *optimal*
PROB5	1.32621 · *optimal*
PROB6	1.41222 · *optimal*
PROB7	1.26649 · *optimal*
PROB8	1.44883 · *optimal*
PROB9	1.54114 · *optimal*

Problems	WORST
PROB11	1.19146 · *optimal*
PROB12	1.15919 · *optimal*
PROB13	1.20283 · *optimal*
PROB14	1.3471 · *optimal*
PROB15	1.39283 · *optimal*
PROB16	1.50093 · *optimal*
PROB17	1.29931 · *optimal*
PROB18	1.53053 · *optimal*
PROB19	1.61134 · *optimal*

Problems	WORST
PROB21	1.20243 · *optimal*
PROB22	1.14979 · *optimal*
PROB23	1.24834 · *optimal*
PROB24	1.4074 · *optimal*
PROB25	1.48952 · *optimal*
PROB26	1.6514 · *optimal*
PROB27	1.3405 · *optimal*
PROB28	1.65755 · *optimal*
PROB29	1.76806 · *optimal*

Table 5.3: Average worst results

Problems	Random Solution	One Step Greedy Heuristics	Modified Bind and Rollout Heuristic	Optimal Solution
PROB1, PROB11, PROB21	0.0020 sec/run	0.0002 sec/run	0.0147 sec/run	0.0002 sec/run
PROB2, PROB12, PROB22	0.0022 sec/run	0.0010 sec/run	0.0170 sec/run	0.0002 sec/run
PROB3, PROB13, PROB23	0.0115 sec/run	0.0007 sec/run	0.0691 sec/run	0.001 sec/run
PROB4, PROB14, PROB24	0.0062 sec/run	0.0017 sec/run	0.1137 sec/run	0.0646 sec/run
PROB5, PROB15, PROB25	0.0182 sec/run	0.0030 sec/run	0.5204 sec/run	0.55 sec/run
PROB6, PROB16, PROB26	0.0140 sec/run	0.0060 sec/run	2 sec/run	74.9 sec/run
PROB7, PROB17, PROB27	0.0170 sec/run	0.0015 sec/run	0.1112 sec/run	0.0042 sec/run
PROB8, PROB18, PROB28	0.0090 sec/run	0.0045 sec/run	8.625 sec/run	212.82 sec/run
PROB9, PROB19, PROB29	0.14 sec/run	0.0047 sec/run	55.75 sec/run	8049.38 sec/run

Table 5.4: Average time spent for test problems

Chapter 6

Conclusion and Extensions

Bind and rollout approach in this study is an initial attempt to construct the decision tree bottom-up. However it is accurate and delivers near-optimal test sequences in reasonable time, even for large problems. When evaluating the running time of the algorithm, it is important to note that this algorithm is run only once for an application and the rule returned by the algorithm is then used repeatedly by the user for classification.

The power of this approach is that it tries to see the whole picture. In traditional methods, the algorithms are myopic, because the test which seems *currently* the best is selected at each iteration. However, there may not exist the desired tests with reasonable costs to use later. These methods usually don't care about the next move, hence may come up with solutions far away from optimum. In bind and rollout approach, the algorithm performs a move considering the next moves. Exact cost of current move is added to average estimated cost of upcoming moves, which is the typical property of rollout strategy.

Algorithms described in the literature construct the tree starting from first test to perform which is the most critical one. It is obvious that, in a classification problem, a test that is performed earlier in sequential progress (i.e., in upper parts of decision tree) is more critical than a test performed later. Since there exist more states in a node of ambiguity in upper parts of tree, cost contribution in that part is more than the lower parts of the tree. Also the corresponding column of diagnostic dictionary matrix affects the solution much more if a test is performed earlier. It is not a good idea to guess the answer without binding the pieces together. In bind and rollout algorithm, relatively unimportant decisions are performed first, which

makes it much more flexible compared to the traditional methods. This strategy also provides the possibility of existence of multiple correct decisions that lead to good results.

The drawback of this algorithm is the additional computational work to perform at each iteration. But considering that this algorithm is run only once for an application, the increase in quality of solutions may be worth the increase in time required for certain applications.

Bind and rollout approach is easily applicable to any test sequencing problem. The approach may be used with any one-step or multistep look-ahead algorithm and it will obviously improve the performance of the look-ahead algorithm. In this study we employed four simple one-step look-ahead algorithms in bind and rollout approach but inclusion of a more approximate algorithm as heuristic test sequencing algorithm will improve the results. To sum up, bind and rollout approach guarantees to give better results than any look-ahead algorithm by employing that algorithm as heuristic test sequencing algorithm, H.

The algorithm described here can be used not only to obtain fast-near optimal results, but also to obtain upper bounds in heuristic AND/OR graph search methods. Combining that algorithm with a heuristic that constructs tight lower bounds, may lead to quickly found optimal solutions via a search method.

Bibliography

[1] A. Biassizo, A. Žužek, and F.Novak. Sequential diagnosis with asymmetrical tests. *Comput. J.*, 41(3):163–170, April 1998.

[2] K. Fraughnaugh, J. Ryan, H. Zullo, and L. A. Cox Jr. Heuristics for efficient classification. *Annals of Operations Research*, 78:189–200, 1998.

[3] M. R. Garey. Optimal binary identification procedures. *SIAM J. Appl. Math.*, 23(2):173–186, July 1972.

[4] M. R. Garey and R. L. Graham. Performance bounds on the splitting algorithm for binary testing. *Acta Inform.*, (3):347–355, 1974.

[5] D. A. Huffman. A method for the construction of minimum-redundancy codes. In *Proceedings of the I.R.E.*, pages 1098–1101, September 1952.

[6] R. A. Johnson. An information theory approach to diagnosis. In *Proc. 6th Symp. Rel. Quality Contr.*, volume 24, pages 102–109, 1960.

[7] A. Martelli and U. Montanari. Additive and/or graphs. In *Proc. 3rd Int. Conf. Artificial Intell.*, pages 1–11, Stanford, CA, August 1973.

[8] M. Mastrianni. Virtual integrated test and diagnostics. *Sikorsky Pres. ARPA*, 9 November 1994.

[9] K. R. Pattipati and M. G. Alexandridis. Application of heuristic search and information theory to sequential fault diagnosis. *IEEE Trans. Syst., Man, Cybern.*, 20:872–887, July 1990.

[10] K. R. Pattipati and M. Dontamsetty. On a generalized test sequencing problem. *IEEE Trans. Syst., Man, Cybern.*, 22(2):392–396, March-April 1992.

[11] V. Raghavan, M. Shakeri, and K. R. Pattipati. Optimal and near-optimal test sequencing algorithms with realistic test models. *IEEE Trans. Syst., Man, Cybern.*, 29:11–27, January 1999.

[12] V. Raghavan, M. Shakeri, and K. R. Pattipati. Test-sequencing problems arising in design planning and design for testability. *IEEE Trans. Syst., Man, Cybern. A*, 30:1–14, January 2000.

[13] M. Shakeri, V. Raghavan, K. R. Pattipati, and A. Patterson-Hine. Sequential testing algorithms for multiple fault diagnosis. *IEEE Trans. Syst., Man, Cybern. A*, 29(2):153–163, March 1999.

[14] F. Tu and K. R. Pattipati. Rollout strategies for sequential fault diagnosis. *IEEE Trans. Syst., Man, Cybern. A*, 33(1):86–99, January 2003.

[15] F. Tu, K. R. Pattipati, S. Deb, and V. N. Malepati. Computationally efficient algorithms for multiple fault diagnosis in large graph-based systems. *IEEE Trans. Syst., Man, Cybern. A*, 33(1):73–85, January 2003.

[16] P. K. Varshney, C. R. P. Hartmann, and J. M. DeFaria Jr. Application of information theory to sequential fault diagnosis. *IEEE Trans. on Computers*, C-31:164–170, February 1982.

[17] A. Žužek, A. Biassizo, and F.Novak. Sequential diagnosis tool. *Microprocess. Microsyst.*, 24:191–197, August 2000.

[18] R. W. Yeung. On noiseless diagnosis. *IEEE Trans. Syst., Man, Cybern.*, 24:1074–1082, July 1994.

Appendix A

Failure Probability Computation

Note that the techniques described here are taken from Raghavan *et al.* [11].

A.1 Prior Probabilities of Failures

Let $F(t)$ represent the Cumulative Density Function (CDF) of the time to failure of a given component s_i. It is known that the component s_i was put into operation at time $t = 0$. Furthermore, s_i was known to be good at time $t = t_1$. It is desired to compute the probability p'_i of it being faulty at time $t = t_2$, where $t_2 > t_1$. This implies that the failure of s_i could have occured at any time in the interval $(t_1, t_2]$, in case it did occur. Hence we need to compute

$$\begin{aligned} p'_i &= \text{Prob } (s_i \text{ failed at } t = t_2 | s_i \text{ is good at } t = t_1) \\ &= \text{Prob } (t_1 < x \leq t_2 | t_1 < x) \end{aligned} \tag{A.1.1}$$

where x is the time at which s_1 failed. In terms of the CDF $F_i(t)$ of the failure time, we can write p'_i as

$$p'_i = \frac{F_i(t_2) - F_i(t_1)}{1 - F_i(t_1)} \tag{A.1.2}$$

Thus, given the cumulative distribution function of the failure time of a component, we can compute the *a priori* probability of failure for that component.

A.2 Conditional Failure Probabilities

For well-maintained systems, it is assumed that at most a single failure exists in the system at the time of testing. This is a valid assumption for many mission critical systems, which are frequently tested for faults and refurbished regularly. Let $S =$

$\{s_1, s_2, \ldots, s_m\}$ be the set of failure states in the system. Let $P' = \{p'_1, p'_2, \ldots, p'_m\}$ be the set of *a priori* probabilities of the failure sources. We assume that the failure states s_i for $1 \leq i \leq m$ are independent.

Let us assume that $S_I \subseteq S$ denotes a set of failure states such that, all failure states $s_i \in S_I$ have occured and all states $s_j \notin S_I$ have not occured. By our assumption of system state independence, the probability of above event is given by $P(S_I) = \prod_{s_i \in S_I} p'_i \prod_{s_j \notin S_I} (1 - p'_j)$. Therefore, the probability that the system is fault free is $p'_0 = \prod_{s_j \in S} (1 - p'_j)$.

Now, if we assume that only a single fault or no fault could be present in the system at any given time, then the set of admissible events include only those involving $S_I = \{s_i\}$ $\forall s_i \in S$, and $S_I = \emptyset$. Therefore the single fault probabilities of the various system states (s_0 being the fault-free state) are given via

$$\begin{aligned} p(s_i) &= \frac{\frac{p'_i}{1-p'_i}}{1 + \sum_{j=1}^{m} \frac{p'_j}{1-p'_j}} \quad \forall 1 \leq i \leq m \\ p(s_0) &= \frac{1}{1 + \sum_{j=1}^{m} \frac{p'_j}{1-p'_j}} \end{aligned} \tag{A.2.3}$$

Thus the test sequencing problem with single fault assumption can be formulated with the failure state probability vector $P = \{p(s_0), p(s_1), \ldots, p(s_m)\}$, starting from the *a priori* probability vector $P' = \{p'_1, p'_2, \ldots, p'_m\}$.

Bibliography

[1] A. Biassizo, A. Žužek, and F.Novak. Sequential diagnosis with asymmetrical tests. *Comput. J.*, 41(3):163–170, April 1998.

[2] K. Fraughnaugh, J. Ryan, H. Zullo, and L. A. Cox Jr. Heuristics for efficient classification. *Annals of Operations Research*, 78:189–200, 1998.

[3] M. R. Garey. Optimal binary identification procedures. *SIAM J. Appl. Math.*, 23(2):173–186, July 1972.

[4] M. R. Garey and R. L. Graham. Performance bounds on the splitting algorithm for binary testing. *Acta Inform.*, (3):347–355, 1974.

[5] D. A. Huffman. A method for the construction of minimum-redundancy codes. In *Proceedings of the I.R.E.*, pages 1098–1101, September 1952.

[6] R. A. Johnson. An information theory approach to diagnosis. In *Proc. 6th Symp. Rel. Quality Contr.*, volume 24, pages 102–109, 1960.

[7] A. Martelli and U. Montanari. Additive and/or graphs. In *Proc. 3rd Int. Conf. Artificial Intell.*, pages 1–11, Stanford, CA, August 1973.

[8] M. Mastrianni. Virtual integrated test and diagnostics. *Sikorsky Pres. ARPA*, 9 November 1994.

[9] K. R. Pattipati and M. G. Alexandridis. Application of heuristic search and information theory to sequential fault diagnosis. *IEEE Trans. Syst., Man, Cybern.*, 20:872–887, July 1990.

[10] K. R. Pattipati and M. Dontamsetty. On a generalized test sequencing problem. *IEEE Trans. Syst., Man, Cybern.*, 22(2):392–396, March-April 1992.

[11] V. Raghavan, M. Shakeri, and K. R. Pattipati. Optimal and near-optimal test sequencing algorithms with realistic test models. *IEEE Trans. Syst., Man, Cybern.*, 29:11–27, January 1999.

[12] V. Raghavan, M. Shakeri, and K. R. Pattipati. Test-sequencing problems arising in design planning and design for testability. *IEEE Trans. Syst., Man, Cybern. A*, 30:1–14, January 2000.

[13] M. Shakeri, V. Raghavan, K. R. Pattipati, and A. Patterson-Hine. Sequential testing algorithms for multiple fault diagnosis. *IEEE Trans. Syst., Man, Cybern. A*, 29(2):153–163, March 1999.

[14] F. Tu and K. R. Pattipati. Rollout strategies for sequential fault diagnosis. *IEEE Trans. Syst., Man, Cybern. A*, 33(1):86–99, January 2003.

[15] F. Tu, K. R. Pattipati, S. Deb, and V. N. Malepati. Computationally efficient algorithms for multiple fault diagnosis in large graph-based systems. *IEEE Trans. Syst., Man, Cybern. A*, 33(1):73–85, January 2003.

[16] P. K. Varshney, C. R. P. Hartmann, and J. M. DeFaria Jr. Application of information theory to sequential fault diagnosis. *IEEE Trans. on Computers*, C-31:164–170, February 1982.

[17] A. Žužek, A. Biassizo, and F.Novak. Sequential diagnosis tool. *Microprocess. Microsyst.*, 24:191–197, August 2000.

[18] R. W. Yeung. On noiseless diagnosis. *IEEE Trans. Syst., Man, Cybern.*, 24:1074–1082, July 1994.