MULTIPLE FAULT ISOLATION IN REDUNDANT SYSTEMS

NASA-Ames/University Aerospace Institutes Agreement

NCC2-5123

Final Report

Dr. Krishna R. Pattipati
Dept. of Electrical and Systems Engineering
University of Connecticut
Storrs, CT 06269-3157
Tel./Fax: (860) 486-2890/486-5585

Dr. Ann Patterson-Hine
Dr. David Iverson
NASA-Ames Research Center
Moffett Field, CA 94035-1000

Submitted to:

Ms. Amy Chu
Director
NASA-Ames University Consortium
NASA-Ames Research Center
Moffett Field, CA 94035

NASA Final Report

May 1997

TABLE OF CONTENTS

ABSTRACT

Fault diagnosis in large-scale systems that are products of modern technology present formidable challenges to manufacturers and users. This is due to large number of failure sources in such systems and the need to quickly isolate and rectify failures with minimal down time. In addition, for fault-tolerant systems and systems with infrequent opportunity for maintenance (e.g., Hubble telescope, space station), the assumption of at most a single fault in the system is unrealistic. In this project, we have developed novel block and sequential diagnostic strategies to isolate multiple faults in the shortest possible time without making the unrealistic single fault assumption.

1.EXECUTIVE SUMMARY

1.1Problem Definition and Significance

Diagnosis is the process of identifying the *cause* of a malfunction by observing its *effects* at various monitoring/test points in a system. As technology advances, there is a significant increase in the complexity and sophistication of systems. Moreover, integration and miniaturization have sharply limited access to test points. Thus, the number of failure sources have increased while reduction in monitoring points have resulted in reduced fault observability, making it increasingly difficult to troubleshoot these systems. Consequently, system maintenance presents formidable challenges to manufacturers and users. In this vein, computer-aided design techniques for system modeling and computational algorithms for test sequencing are of paramount significance. This research has developed novel multiple fault diagnosis algorithms to directly address this vital need.

Maintenance and design have traditionally been two separate engineering disciplines with often conflicting objectives: maximizing ease of maintenance versus optimizing performance, size and cost. Testability analysis has been an ad hoc, manual effort, in which maintenance engineers attempt to identify an efficient method of troubleshooting for the given product, with little or no control over product design. Testability deficiencies in the design can not therefore be rectified. This adversely impacts the life-cycle cost. It is now widely recognized that testability must be engineered into the product at the design stage itself, so that an optimal compromise is achieved between system maintainability and performance. This process of refining a system design to improve testability is termed Design for Testability (DFT), and is now a requirement in most complex system development projects.

Our previous research has developed *multi-signal directed graph* modeling techniques that enable the representation of a system either top-down (as lower-level details become available), bottom-up (for system integration tasks) or a combination of both. In addition, we have devised test sequencing algorithms to analyze the testability of a system design, and to determine a near optimal sequence of tests for diagnosing *single faults* in hierarchical systems. A solution to the test sequencing problem is a decision tree, which specifies the test to perform next depending on the outcomes of previously applied tests. A novel feature of our approach is the integration of concepts from information theory and AND/OR graph search techniques to overcome the computational explosion of the optimal test sequencing problem [1]. Furthermore, the top-down nature of the search algorithms have enabled us to derive a variety of near-optimal and practical diagnostic strategies that provide a tradeoff between the degree of suboptimality and computational complexity [1-11]. The resulting algorithms have demonstrated their utility on large hierarchical systems: Boeing-Sikorsky has employed our algorithms on a flight-control system model with 8 levels of hierarchy and 10,000 faults and test points; we have generated the troubleshooting strategies of a space shuttle main propulsion system with 7000 failure sources and a similar number of test points (using a digraph model

provided by NASA-Ames) in only 1.5 hours on a Sparc station 10. In this effort, we have extended the single-fault diagnostic strategies to situations where multiple faults may be present.

1.2 Research Results

1.2.1 Sequential Algorithms for Multiple Fault Diagnosis

As part of our effort on multiple fault diagnosis, we investigated the problem of constructing near-optimal test sequencing algorithms for diagnosing multiple faults in complex systems. The computational complexity of solving the optimal multiple-fault isolation problem is super-exponential, that is, it is much more difficult than the single-fault isolation problem [1], which, by itself, is exponential. By employing concepts from information theory and Lagrangian relaxation, we developed several static and dynamic (on-line or interactive) test sequencing algorithms for the multiple fault isolation problem that provide a tradeoff between the degree of suboptimality and computational complexity. Furthermore, we derived novel diagnostic strategies that generate a static diagnostic directed graph (digraph), instead of a static diagnostic tree, for multiple fault diagnosis. Using this approach, the storage complexity of the overall diagnostic strategy reduces substantially. Computational results based on real-world systems from Sikorsky Aircraft indicate that the size of static multiple fault strategy is strictly related to the structure of the system, and that the use of an on-line multiple fault strategy can diagnose faults in systems with as many as 10,000 failure sources. The details on sequential multiple fault strategies may be found in the following references:

12. Shakeri, M., Pattipati, K., Raghavan, V., Patterson-Hine, A., and Kell, T.," Sequential Test Strategies for Multiple Fault Isolation", *1995 IEEE AUTOTESTCON*, Atlanta, GA, Aug. 1995.

13. Shakeri, M., Pattipati, K., Raghavan, V., Patterson-Hine, A., and Iverson, D.L.," Multiple Fault Isolation in Redundant Systems", *1995 IEEE International Conference on Systems, Man and Cybernetics*, Van Couver, BC, October 1995.

14. Shakeri, M., Raghavan, V., Pattipati, K., and Patterson-Hine, A., "Sequential Testing Algorithms for Multiple Fault Isolation," submitted to *IEEE Transactions on Systems, Man and Cybernetics*, August 1996.

1.2.2 Fault Diagnosis with Imperfect Tests

We investigated two fault diagnosis problems for the case when tests are imperfect : (1) sequential fault diagnosis under single fault assumption; and (2) fault diagnosis when all test results are available as a block.

When tests are imperfect, the test sequencing problem corresponds to a partially observed Markov decision problem (POMDP), a sequential multi-stage decision problem wherein the states are the set of possible failure sources and information regarding the states is obtained via

the results of imperfect tests. The optimal solution for this problem was obtained by applying a continuous state dynamic programming (DP) recursion. However, the DP recursion is computationally very expensive owing to the continuous nature of the state vector comprising the probabilities of faults. In order to alleviate the computational explosion, we developed an efficient implementation of the DP recursion. We also considered various problems with special structure (e.g., parallel systems) and derived closed-form solution/index-rules without having to resort to DP. Finally, we developed a variety of top-down graph search algorithms for problems with no special structure, including multi-step DP, multi-step information heuristics and certainty equivalence algorithms. We compared these near-optimal algorithms with DP for small problems to gauge their effectiveness. The details on test sequencing with unreliable tests may be found in the following reference:

15. Raghavan, V., Shakeri, M., and Pattipati, K., "Test Sequencing Algorithms with Unreliable Tests," submitted to *IEEE Transactions on Systems, Man and Cybernetics*, August 1996.

Next, we considered the problem of constructing optimal and near-optimal multiple fault diagnosis (MFD) in bipartite systems (i.e., systems with failure sources connected directly with tests) with unreliable tests. It is known that exact computation of conditional probabilities for multiple fault diagnosis is NP-hard. The novel features of our diagnostic algorithms was the use of Lagrangian relaxation and subgradient optimization methods to provide: (1) near-optimal solutions for the MFD problem, and (2) upper bounds for an optimal branch-and-bound algorithm. The proposed method was illustrated using several medical diagnosis examples. Computational results indicated that: (1) our algorithm has superior computational performance to the existing algorithms (approximately three orders of magnitude improvement over the algorithms in the artificial intelligence literature; (2) the near-optimal algorithm generates the most likely candidates with very high accuracy; and (3) our algorithm can find the most likely candidates in systems with as many as 1000 faults. The details of the algorithm may be found in the following references:

16. Shakeri, M., Raghavan, V., Pattipati, K., and Patterson-Hine, A., "Optimal and Near-optimal Algorithms for Multiple Fault Diagnosis with Unreliable Tests," *1996 IEEE AUTOTEST Conference*, Dayton, OH, September 1996.

17. Shakeri, M., Raghavan, V., Pattipati, K., and Patterson-Hine, A., "Algorithms for Multiple Fault Diagnosis with Unreliable Tests," submitted to *IEEE Transactions on Systems, Man and Cybernetics*, August 1996.

2.LIST OF PUBLICATIONS AND PRESENTATIONS

2.1Publications

7. Shakeri, M., Pattipati, K., Raghavan, V., Patterson-Hine, A., and Kell, T.," Sequential Test Strategies for Multiple Fault Isolation", *1995 IEEE AUTOTESTCON*, Atlanta, GA, Aug. 1995.

8. Shakeri, M., Pattipati, K., Raghavan, V., Patterson-Hine, A., and Iverson, D.L.," Multiple Fault Isolation in Redundant Systems", *1995 IEEE International Conference on Systems, Man and Cybernetics*, Van Couver, BC, October 1995.

9. Shakeri, M., Raghavan, V., Pattipati, K., and Patterson-Hine, A., "Sequential Testing Algorithms for Multiple Fault Isolation," submitted to *IEEE Transactions on Systems, Man and Cybernetics*, August 1996.

10. Raghavan, V., Shakeri, M., and Pattipati, K., "Test Sequencing Algorithms with Unreliable Tests," submitted to *IEEE Transactions on Systems, Man and Cybernetics*, August 1996.

11. Shakeri, M., Raghavan, V., Pattipati, K., and Patterson-Hine, A., "Optimal and Near-optimal Algorithms for Multiple Fault Diagnosis with Unreliable Tests," *1996 IEEE AUTOTEST Conference*, Dayton, OH, September 1996.

12. Shakeri, M., Raghavan, V., Pattipati, K., and Patterson-Hine, A., "Algorithms for Multiple Fault Diagnosis with Unreliable Tests," submitted to *IEEE Transactions on Systems, Man and Cybernetics*, August 1996.

2.2Presentations

The research results were presented at the following conferences and corporations:

- 1995 IEEE Systems, Man and Cybernetics Conf., San Antonio, TX; Van Couver, BC.
- 1996 IEEE Automatic Testing Conference, Anaheim, CA; Atlanta, GA; Dayton, OH.
- Sikorsky Aircraft, Stratford and Bridgeport, CT
- Boeing Helicopters, Philadelphia, PA
- Boeing Defense and Space Group, Seattle, WA
- Lockheed-Martin Integrated Systems, Orlando, FL
- NASA-Ames Research Center, Moffett Field, CA.

2.3Awards

- Professor Krishna R. Pattipati was elected Fellow of the *Institute of Electrical and Electronics Engineers* (IEEE) for contributions to "discrete optimization algorithms for large-scale systems, and team decisionmaking" in 1995.
- The paper entitled, "Sequential Testing Algorithms for Multiple Fault Isolation," received the Best Student Paper Award at the 1995 IEEE Automatic Testing Conference, Atlanta, GA, August 1995. A $250. cash award was given to student authors.

3.BIBLIOGRAPHY

1. Pattipati, K.R., and M.G. Alexanridis, " Application of Heuristic Search and Information Theory to Sequential Fault Diagnosis, " *IEEE Trans. on SMC*, Vol. 20, No. 4, July 1990, pp. 872-887.

2. Pattipati, K.R., S. Deb, M. Dontamsetty, and A. Maitra, " START: System Testability Analysis and Research Tool," *IEEE AES Magazine*, January 1991, pp. 13-20.

3. Pattipati, K.R., and M. Dontamsetty, "On a Generalized Test Sequencing Problem," *IEEE Trans. on SMC*, Vol. 22, No. 2, March 1992, pp. 332-336.

4. Pattipati, K.R., M.G. Alexandridis, and J.C. Deckert, "Time Efficient Sequencer of Tests (TEST)," *Proceedings of the IEEE AUTOTESTCON*, Long Island, NY, Oct. 1985, pp. 49-62.

5. Raghavan, V., Shakeri, M., and Pattipati, K., "Optimal and Near-optimal Test Sequencing Algorithms with Realistic Test Models," submitted to *IEEE Transactions on Systems, Man and Cybernetics*, August 1996.

6. Raghavan, V., Shakeri, M., and Pattipati, K., "Test Sequencing Problems Arising in Design For Testability," submitted to *IEEE Transactions on Systems, Man and Cybernetics*, August 1996.

7. Raghavan, V., "*Algorithms for Sequential Fault Diagnosis*," Ph.D. Thesis, Department of Electrical and Systems Engineering, University of Connecticut, Storrs, CT 06269-3157, May 1996.

8. Pattipati, K.R., V. Raghavan, S. Deb, M. Shakeri and R. Shrestha, "TEAMS: Testability Engineering And Maintenance System", invited paper at *1994 ACC* , Baltimore, MD, June 1994.

9. Deb, S., K.R. Pattipati, V. Raghavan, M. Shakeri, and R. Shrestha, "Multi-Signal Flow Graphs: A Novel Approach for System Testability Analysis and Fault Diagnosis," *IEEE Aerospace and Electronic Systems Magazine*, May 1995, pp. 14-25.

10. Shakeri, M., "*Advances in Fault Diagnosis and Testability Analysis of Large-Scale Systems*," Ph.D. Thesis, University of Connecticut, Storrs, CT, 06269, August 1996.

11. Shakeri, M., Pattipati, K., Raghavan, V., and Deb. S.,"Near Optimal Sequential Testing Algorithms for Multiple Fault Isolation.", Proc. of the *IEEE SMC conference*, San Antonio, Texas, Oct. 1994, pp. 1908-1913.

1995 IEEE Autotest Conference, September 1995

Sequential Test Strategies for Multiple Fault Isolation*

M. Shakeri, K.R. Pattipati and V. Raghavan
U-157, Department of Electrical and Systems Engineering,
University of Connecticut, Storrs, CT 06269-3157
E-mail: krishna@sol.uconn.edu

A. Patterson-Hine
NASA-Ames Research Center, Mail Stop 269-4,
Moffett Field, CA 94035-1000
E-mail: Ann_Patterson-Hine@styx.arc.nasa.gov

T. Kell
Technical Support Services,
Sikorsky Aircraft, Bridgeport, CT 06604
E-mail: ekell@sikorsky.com

Abstract

In this paper, we consider the problem of constructing near-optimal test sequencing algorithms for diagnosing multiple faults in redundant (fault-tolerant) systems. The computational complexity of solving the optimal multiple-fault isolation problem is super-exponential, that is, it is much more difficult than the single-fault isolation problem, which, by itself, is NP-hard[1] [1]. By employing concepts from information theory and Lagrangian relaxation, we present several static and dynamic (on-line or interactive) test sequencing algorithms for the multiple fault isolation problem that provide a trade-off between the degree of suboptimality and computational complexity. Furthermore, we present novel diagnostic strategies that generate a static diagnostic directed graph (digraph), instead of a static diagnostic tree, for multiple fault diagnosis. Using this approach, the storage complexity of the overall diagnostic strategy reduces substantially. Computational results based on real-world systems indicate that the size of a static multiple fault strategy is strictly related to the structure of the system, and that the use of an on-line multiple fault strategy can diagnose faults in systems with as many as 10,000 failure sources.

1 Introduction

The complexity associated with the maintenance of large integrated systems, such as the space shuttle or a modern aircraft consisting of mechanical, electro-mechanical and hydraulic subsystems, presents formidable challenges to manufacturers and end users. This is due to the large number of failure sources and the need to quickly isolate and rectify such failures with minimal down time. In addition, for redundant (fault-tolerant) systems and for systems with little or no opportunity for repair or maintenance during their operation (e.g., Hubble telescope, space station), the assumption of at most a single failure in the system between consecutive maintenance actions is unrealistic. Thus, the efficient maintenance of complex redundant systems requires advanced diagnostic algorithms for multiple fault isolation. This paper considers the problem of constructing efficient algorithms for diagnosing multiple faults in systems with and without redundancy.

For diagnostic purposes, we only need to model how a failure (or cause) propagates to the various monitoring points. Consequently, it is sufficient to model the system in its failure space. That is, the model does not describe how the system normally performs, but how the various failure sources manifest themselves as malfunctions. The failure propagation is modeled in the form of first-order cause-effect relationships using digraph techniques. The fundamental premise of digraph techniques is that the cause-effect linkages must connect the fault origin to the observed symptoms of the fault. The digraph models encompass a variety of modeling approaches, including dependency models [32], signed directed graphs [33], and fault trees [34].

Once a system is described in terms of a digraph model, the full order dependencies among failure sources and tests can be captured by a binary test matrix B, consisting of the failure sources as row indices and the tests as column indices [12]. This binary test matrix can be used to diagnose single faults, as well as multiple faults in sys-

*Research supported in part by the Department of Economic Development of the State of Connecticut, NASA-Ames Research Center, Sikorsky Aircraft and Qualtech Systems, Inc.

[1]This means that the computational requirements of an optimal algorithm cannot be bounded by a polynomial function of the number of failure sources and/or the number of tests.

tems having no redundancy. This assertion is based on the assumption that the failure sources are independent and, consequently, the failure signature of a multiple failure is the union of failure signatures of individual failure sources. However, this property is not valid for systems with redundancy, even under the assumption of failure independence. The single faults and minimal faults, i.e., minimum number of faults with a failure signature different from the union of failure signatures of individual faults, together with their failure signatures, constitute the necessary information for fault diagnosis in redundant systems. Thus, the problem of generating a binary test matrix in redundant systems reduces to the problem of finding minimal faults of a digraph model. After generating the binary test matrix, the problem is to design a sequential testing strategy for diagnosing multiple faults. Thus, multiple fault diagnosis involves two sequential steps: (1) generation of a binary test matrix, which contains all the necessary information for single-fault and multiple-fault diagnosis, and (2) design of a multiple-fault testing strategy that unambiguously isolates the failure sources with minimum expected testing cost (or time).

The problem of finding minimal faults in digraph models is much more difficult than that in the fault tree models, which, by itself, is NP-hard [2]. This is because a fault tree model contains no cycles (feedback loops), and because there exists only one target event, for which the minimal faults (cuts) should be computed. Rauzy [2] considered the problem of computing minimal faults (cuts) of fault tree models, and presented an efficient method to compute them using binary decision diagrams. Vatn [3] presented a method for the identification of minimal cut sets in a fault tree. The cut sets are stored in a virtual tree structure. In this method, by traversing the virtual tree, minimal cuts of size one are identified first. Then, in the second iteration, all minimal cuts of size two are identified and compared with the cut sets of size one to exclude non-minimal cuts. This procedure is continued until all minimal cuts are identified.

Since the number of minimal cuts can increase exponentially with the size of the tree, it is practical to truncate the computation by neglecting higher order and/or low-probability faults. Brown [4, 5] presented an algorithm that uses probability-based truncation, and determines a rigorous upper bound on each event-probability by propagating the effect of all the truncated cut sets in the form of numeric residuals. Iverson and Patterson-Hine [6] considered the problem of generating singletons (single fault) and doubletons (double faults) in digraph models. A major contribution of this paper is the development of a top-down recursive algorithm that finds all the minimal faults in digraph models, and an efficient bottom-up algorithm that finds minimal faults up to a limited size. The failure signatures of minimal faults are generated thereafter, and the single-fault binary test matrix is augmented to include this information.

Davis [7, 8] described a fault diagnosis system that reasons from the knowledge of structure and behavior. Failure candidate generation in this approach occurs in three basic steps: circuit simulation and discrepancy collection, potential candidate determination, and global consistency determination using constraint suspension techniques. However, for multiple fault diagnosis, this approach suffers from severe computational explosion. de Kleer and Williams [9] presented a model-based approach to fault diagnosis. By keeping track of multiple sets of consistent and inconsistent components, their algorithm generates minimal sets of faulty candidates rather than generating all possible candidates. This approach requires the complete specification of system components, the state and observed variables associated with each component, and the functional relationships among the state variables. However, the precise information required by these models is typically not available for complex systems and is too costly to obtain. In addition, because of extensive use of functional simulation, this approach is extremely slow, and, thus, is not appropriate for fault diagnosis in large scale systems with the complexities of many orders of magnitude more than the examples presented in [9]. Sheppard and Simpson [35] provided a formal analysis of the multiple failure problem in the context of information flow model. They discussed the computational complexity of several algorithms for diagnosing multiple failures, and developed algorithms to generate multiple fault diagnoses for a given ambiguity group. However, this method does not take into account the failure probabilities of components, test costs, or system redundancies.

In this paper, we first extend the single-fault strategy of our previous work [1, 10, 12, 28] to diagnose multiple faults by successive replacement of single fault candidates. Using this strategy, we seek to isolate the potential single-fault candidates, then double-fault candidates, and so on. Since a component may be repaired/replaced before confirming that it is indeed faulty, the probability of false alarm error or RTOK (retest OK) is higher than that with multiple fault strategies that use all informative tests before repairing a component in the system.

Next, we focus on developing a class of Sure strategies [11] for diagnosing multiple faults in digraph models that employ all informative tests before diagnosis. The basic idea of these strategies is to find one or more definitely failed components, while not making an error when other co-existing faults are present. Furthermore, in order to eliminate the problems associated with the stor-

age of the complete diagnostic strategy, an interactive testing strategy has been implemented. Instead of generating the entire diagnostic tree, the interactive testing strategy suggests the next test to be applied, given the outcomes of previously applied tests, and generates the path leading to the isolation of multiple failures in a system. We employ concepts from information theory and Lagrangian relaxation to generate several on-line diagnostic strategies. Using these strategies, we can diagnose multiple faults in large systems with as many as 10,000 failure sources.

2 Problem Formulation

We assume that the system is modeled by the digraph $\mathbf{DG} = \{\mathbf{S}, \mathbf{T}, \mathbf{A}, \mathbf{E}\}$, where $\mathbf{E}$ denotes the set of directed edges specifying the functional information flow in the system and

- $\mathbf{S} = \{s_1, ..., s_m\}$ is a finite set of *independent* failure sources (failure aspects) associated with the system;
- $\mathbf{T} = \{t_1, t_2, ..., t_n\}$ is a finite set of n available binary outcome tests, where the integrity of system failure sources/components/modules can be ascertained;
- $\mathbf{A} = \{a_1, ..., a_K\}$ is a finite set of AND nodes representing system redundancies.

The input requirements of the various nodes of the directed graph are as follows:

1. Failure node: Unconditional a priori probability vector of failure nodes $\mathbf{P} = [p(s_1), ..., p(s_m)]$, where $p(s_i)$ is the a priori probability of failure source s_i.
2. Test node: A set of test costs $\mathbf{C} = \{c_1, c_2, ..., c_n\}$, where c_j is the cost of applying test t_j, measured in terms of time, manpower requirements, or other economic factors.
3. AND node: Two sets $\mathbf{U} = \{u_1, ..., u_K\}$ and $\mathbf{V} = \{v_1, ..., v_K\}$, where u_k and v_k denote u_k-out-of-v_k logic for AND node a_k, i.e., AND node a_k has v_k inputs and a failure must occur in at least u_k inputs of this AND node for the faults to propagate to the output.

The problem is to design a testing strategy that unambiguously isolates the failure sources with minimum expected testing cost. The AND/OR sequential test strategy is represented in the form of a tree or a graph, where the OR nodes represent the suspect sets of failure sources, AND nodes are tests applied at various OR nodes, and the leaves are the isolated failure sources.

3 Single Fault Testing Strategies

Once a system is described in terms of a digraph model, all the necessary information for fault diagnosis can be captured by a binary test matrix (fault dictionary), $B = [b_{ij}]$ of dimension $m \times n$. In a single fault strategy, it is assumed that the system is tested frequently enough that at most one component has failed. Thus, the test matrix denotes the full-order dependency among single failures and the tests in the system, i.e., the rows and columns of the test matrix correspond to failure sources and tests, respectively. The test matrix can be computed by the reachability analysis algorithms [12].

The single fault diagnosis problem, in its simplest form, is the five-tuple $(S, P, \mathbf{T}, \mathbf{C}, D)$, where

- $S = \mathbf{S} \vee \{s_0\} = \{s_0, s_1, ..., s_m\}$ is a set of failure sources, where s_0 is a dummy failure source denoting fault-free condition and $\vee$ denotes the union of two sets;
- $P = [p_0, p_1, ..., p_m]$ is the *conditional* probability vector associated with the set of failure sources S based on a single fault assumption [11], where p_0 is the probability of fault-free condition, s_0. These are related to unconditional prior probabilities $\{p(s_i)\}$ via:

$$p_0 = \frac{1}{1 + \sum_{k=1}^{m} \frac{p(s_k)}{1-p(s_k)}} \qquad (1)$$

$$p_i = \frac{\frac{p(s_i)}{1-p(s_i)}}{1 + \sum_{k=1}^{m} \frac{p(s_k)}{1-p(s_k)}} \quad \text{for} \quad i = 1, ..., m$$

- $\mathbf{T}$ and $\mathbf{C}$ are as defined in Section 2;
- $D = [d_{ij}]$ is a binary test matrix of dimension $(m+1) \times n$, where $d_{0j} = 0$ for $1 \leq j \leq n$, and $d_{ij} = b_{ij}$ for $1 \leq i \leq m$ and $1 \leq j \leq n$.

The algorithms for designing optimal single-fault diagnostic strategies are based on dynamic programming (DP) [13], and AND/OR graph search procedures. The DP technique is based on a bottom-up procedure, and has storage and computational requirements of $O(3^n)$ for even the simplest test sequencing problem. The AND/OR[2] graph search algorithms are top-down heuristic graph search procedures that employ a cost-to-go estimate to speed up the solution search process [1].

[2]These AND/OR nodes of the search graph should not be confused with the AND nodes of a digraph model. AND/OR graph search formalizes the strategy generation process, where as AND node of the digraph model denotes redundancy.

A novel feature of this approach is that the cost-to-go estimate (termed the Heuristic Evaluation Function (HEF)) is derived from Huffman coding and entropy. These information theoretic lower bounds ensure that an optimal solution is found using the AO*, HS, and CF search algorithms [12]. In addition, because of the top-down nature of the AND/OR graph search algorithms, several near-optimal search algorithms have been derived: (1) AO^*_ϵ algorithm, (2) limited search AO*, and (3) Multi-step information heuristics. Furthermore, because of their top-down nature, these algorithms extend naturally to: (1) modular diagnosis, (2) precedence constraints, setup operations, and resources and (3) rectification. The algorithms have been implemented in a software package, termed TEAMS (Testability Engineering and Maintenance System[12]). For convenience, these algorithms are referred to as the **TEAMS-S** algorithms.

Example 1.a: Consider the digraph model in Figure 1. In this system, there are five failure sources s_1,..., s_5. The set of five tests, labeled t_1,..., t_5, may be used to identify the unknown failure sources. The test matrix, along with the a priori probabilities of failure sources and test costs, is shown in Table 1. Based on a single or no fault assumption, the set of failure aspects $S = \{s_0, s_1, ..., s_5\}$, with the concomitant conditional probability vector P =[0.700, 0.01, 0.020, 0.100, 0.050, 0.120]. An optimal test strategy for this example is shown in Figure 2. For this test strategy, the average test cost is $J = \sum_{i=0}^{m} \sum_{t_j \in TA_i} c_j \cdot p_i$=2.18, where TA_i is the set of applied tests in the path leading to the isolation of failure source $s_i \in S$.

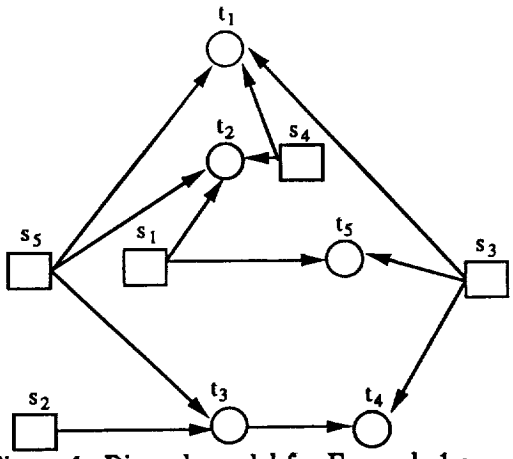

Figure 1: Digraph model for Example 1.a

The single fault assumption may not be valid in situations where the opportunity for frequent maintenance does not exist. In such cases, the single fault strategies can give wrong diagnosis when multiple failures occur. In [11], we showed that the set of hidden faults and masking false failures are potential multiple fault candidates at each leaf node of the single fault diagnostic tree. The set of hidden faults for failure source s_i consists of those failure sources whose failure signatures corresponding to TA_i are subsets of the failure signature of s_i, while the set of masking false failures for failure source s_i consists of those sets of failure sources whose failure signatures corresponding to tests TA_i add up to mask the failure signature of s_i. Hidden faults can be diagnosed by applying a single fault strategy repeatedly [11]. However, if the set of masking false failures at the leaf nodes is not empty, the single fault strategy will give wrong diagnosis, and repairing the implicated fault is obviously of no use in this case. In the next section, we present an extended single fault strategy to diagnose masking false failures, as well as hidden faults in a system.

FAILURE SOURCES	TESTS TEST COSTS c_j 1 1 1 1 1 t_1 t_2 t_3 t_4 t_5	FAULT PROBABILITIES $p(s_i)$
s_1	0 1 0 0 1	0.014
s_2	0 0 1 1 0	0.027
s_3	1 0 0 1 1	0.125
s_4	1 1 0 0 0	0.068
s_5	1 1 1 1 0	0.146

Table 1: Test Matrix, Apriori Fault Probabilities and Test Costs for Example 1.a

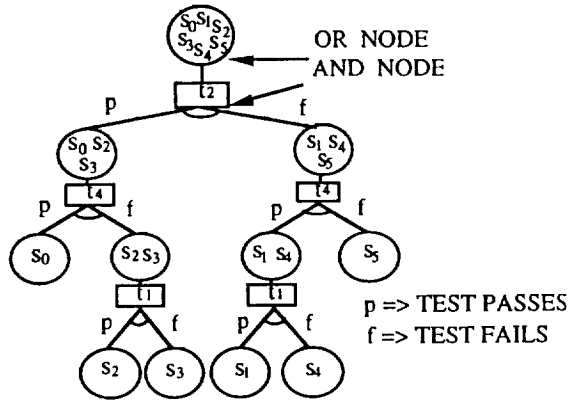

Figure 2: Single-fault Test Strategy for the System of Example 1.a

4 Multiple Fault Diagnosis Using an Extended Single Fault Testing Strategy

In order to formalize this approach, let

- TS_j = test signature associated with test t_j. It indicates all the failure sources detectable by test t_j, i.e., $TS_j = \{s_i | b_{ij} = 1 \text{ for } 1 \leq i \leq m\}$,
- G = union of test signatures of previously passed tests.

In this approach, we invoke a single fault strategy, and repair/replace the identified component at each leaf node, if any. Then, we check whether the repaired/replaced component at each leaf node is definitely faulty or not. If for any test t_j that failed previously, the cardinality of $TS_j - G$ is one, i.e., $TS_j - G$ contains only one failure source, then the corresponding failure source is definitely faulty. If the repaired/replaced component is definitely faulty, we apply additional tests, if necessary, to isolate the remaining faults. Additional tests can be applied from either the root OR node, or from the first failed test in the path leading to the identification of previous faults. This process ensures that we do not come back to the same leaf node twice.

Alternatively, if the replaced module is not definitely faulty, there exist other sets of components which have the same failure signature as the failure signature of replaced module, i.e., masking false failures [11]. In this case, if we start from the root OR node or the first failed test in the path, we may reach the same leaf node. In order to solve this problem, we remove the replaced modules from the ambiguity group at the current stage of diagnosis, and invoke the single fault strategy **TEAMS-S** to isolate the remaining suspected components. Then, we repair/replace the identified modules at each leaf node. If the repaired/replaced module at a leaf node of this tree is definitely faulty, we apply additional tests from the root OR node or from the first failed test *after last repair*. On the other hand, if the identified module at a leaf node is not definitely faulty, we update the ambiguity group and invoke single fault strategy as before. This procedure is **continued until no test gives further information or the system is fault-free.**

One drawback of the extended single-fault strategy is that the probability of repairing/replacing a good component, i.e., false alarm error or RTOK (retest OK), is higher than that with multiple fault strategies that employ all informative tests before repairing a component in the system (see section 5.2). Furthermore, in the case of very large systems, it is practical to solve multiple fault isolation problems up to a certain cardinality $\hat{L} \geq 1$, e.g., single or double failures. This is based on the premise that multiple faults of large cardinality are much less likely to occur. However, in an extended single fault strategy, if we stop expanding the diagnostic tree after limited repair actions, say $\hat{L}$, it does not mean that we can diagnose multiple faults up to size $\hat{L}$ using the same tree. This is because a component may be repaired/replaced before confirming that it is indeed faulty.

Example 1.b: In this example, we consider the same system as in Example 1.a. The extended single fault diagnostic strategy for this example is shown in Figure 3, where the ACTION nodes represent the actions to be performed at each stage of diagnosis. Note that the shaded parts of the tree are the same as those in a single fault diagnostic tree of Figure 2. The average testing cost for this case is J =2.780. The joint probability that s_5 is good, and is repaired/replaced is 0.0103.

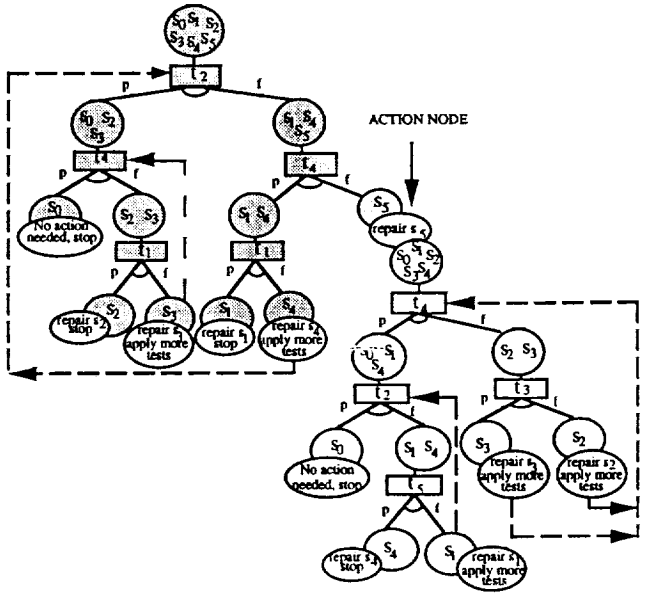

Figure 3: Extended Single Fault Strategy to diagnose multiple faults in Example 1.a

5 Multiple Fault Testing Strategy in Systems without Redundancy (AND nodes)

In digraph models without AND nodes, i.e., without redundancy, a test-matrix containing the full-order dependency among single failures and the tests can be used to diagnose multiple faults. This is because in these models the failure signature of a multiple-failure is assumed to be the union of failure signatures of individual failures (failure independence assumption).

One approach that employs all informative tests before repairing/replacing a component is to consider all possible combinations of failure sources, i.e., 2^S, and generate an optimal multiple fault diagnostic strategy using the single-fault test sequencing algorithm **TEAMS-S**. How-

ever, the storage and computational complexity of optimal multiple-fault isolation problem is super-exponential in m. In order to reduce storage complexity, we use a compact set notation [14], and in order to reduce the computational complexity, we present a class of Sure diagnosis strategies for multiple fault isolation.

5.1 Compact Set Notation

Following Grunberg et al. [14], we use the compact notation $A = \Theta(L; F_1, ..., F_L; G)$ to denote the multiple fault ambiguity group at each OR node. The F_i for $i = 1, ..., L$ and G are subsets of $S = \{s_0, s_1, ..., s_m\}$; G is the set of known good failure sources (failure free sources), and F_i for $i = 1, ..., L$ are sets that are known to contain at least one definitely failed failure source each, i.e.,

$$\Theta(L; F_1, F_2, ..., F_L; G) = \{X \subseteq S | \\ X \wedge F_i \neq \emptyset \text{ for } i = 1, ..., L, \text{ and } X \wedge G = \emptyset\}$$

where $\wedge$ denotes the intersection of two sets. In the following, we summarize some of the properties of compact set notation [11, 30]:

1. Multiple fault logic using the compact set notation is as follows: the initial hypothesis set is the set of all subsets of S, i.e., $A = \Theta(1; F_1 = S ; G = \emptyset)$. After performing a test, say t_j, the hypothesis set $A = \Theta(L; F_1, ..., F_L; G)$ is decomposed as follows:

$$A \leftarrow \begin{cases} \Theta(L; (F_1 \wedge TS_j^c), ..., (F_L \wedge TS_j^c); \\ (G \vee TS_j)) & \text{if } t_j \text{ passes} \\ \Theta(L+1; F_1, ..., F_L, TS_j \wedge G^c; G) & \text{if } t_j \text{ fails} \end{cases}$$

   where TS_j^c and G^c are complements of the sets TS_j and G, respectively.

2. If $E \supseteq F_i$ for some i (that is, E is a superset of F_i), then $\Theta(L+1; F_1, ..., F_L, E; G) = \Theta(L; F_1, ..., F_L; G)$ [14]. **Thus, we should not apply any test whose signature is a superset of one of the F_i's, since the test does not give any new information.**

3. $A = \Theta(L; F_1, ..., F_L; G) = \Theta(L; F_1 \wedge G^c, ..., F_L \wedge G^c; G)$, where superscript c denotes the set complement, i.e., $G^c = S - G$ [14].

4. **Given a set of previously applied passed tests $T_p \subseteq \mathbf{T}$ and failed tests $T_f \subseteq \mathbf{T}$, the multiple fault ambiguity group at the current stage of diagnosis can be generated directly as follows:** $\Theta(L; F_1, ..., F_L; G)$, where $G = \vee_{t_i \in T_p} TS_i$, $L = |T_f| + 1$, $F_1 = S$ (see the first property), and $F_{i+1} = TS_j \wedge G^c$ for $i = 1, ..., |T_f|$ and $t_j \in T_f$; and then, employ property 2 to remove super sets from the set $F = \{F_1, ..., F_L\}$.

5. If $|T_f| = 0$, then $L = 1$ and $s_0 \in F_1$. If $|T_f| > 0$, none of the F_i's contains s_0.

6. The worst case storage complexity of compact set notation for an OR node is $O(mn)$ [11].

7. The failure sources belonging to F_i with cardinality $|F_i| = 1$ are definitely faulty (*one-for-sure* condition).

5.2 Sure Strategies for Multiple Fault Diagnosis

In this section, we present three diagnostic strategies, Sure 1-3, that seek to find definitely failed components, even though there may be others still undiagnosed. Thus, these strategies isolate failures one (or more) at a time, while not making an error when multiple faults are present. The framework for Sure strategies is sketched in Figure 4.

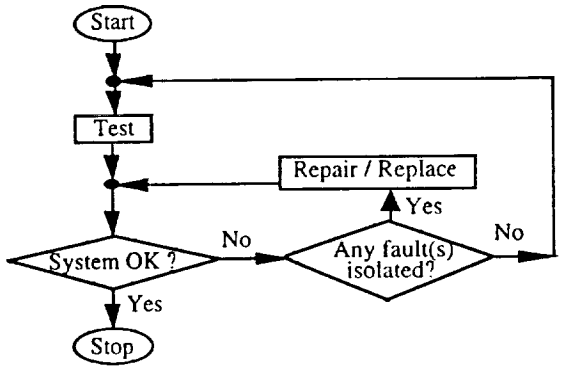

Figure 4: Framework of Sure Strategies in a Test-and-repair Cycle

The three basic ingredients of Sure 1-3 are: (i) *minimal candidate generation*, (ii) *minimal candidate isolation*, and(iii) ***multiple fault propagation.*** **The minimality property implies that a particular candidate includes the minimum number of failure sources that explains all test results observed so far (if any). Consequently, the inherent combinatorial explosion that occurs in generating an optimal multiple fault strategy is reduced substantially.** Before describing the algorithms, we define minimal (irreducible) set and hitting set of a set of subsets:

Definition 1: A minimal or irreducible set for a collection of subsets $Q = \{Q_1, ..., Q_k\}$ is a set $I(Q) \subseteq Q$ such that $I(Q) = Q - \{Q_i | \exists Q_j \in Q \text{ and } Q_j \subseteq Q_i\}$, i.e., $I(Q)$ is equal to set Q without any super set.

Definition 2: A hitting set for a collection of sets $Q = \{Q_1, ..., Q_k\}$ is a set $H(Q) = \{H_1, ..., H_q\}$ such that $H_j \subseteq \vee_{1 \leq i \leq k} Q_i$ for $j = 1, ..., q$, and $H_j \wedge Q_i \neq \emptyset$ for $i = 1, ..., k$.

Based on these definitions, it can be shown that [30]:

Lemma 1: The minimal set of a multiple fault ambiguity group $A = \Theta(L;\ F_1,\ ...,\ F_L; G)$ is the minimal hitting set for the collection of sets $F = \{F_1, ..., F_L\}$, i.e., $I(A) = I(H(F))$.

In Sure 1-3 strategies, at each stage of diagnosis, we consider the minimal candidate set of the multiple fault suspect set corresponding to the OR node at that stage. Reiter [15] has derived an algorithm to determine the minimal hitting set of a collection of sets, and Greiner et al. [16] have presented a correction to the Reiter's algorithm. We use this technique to determine the minimal hitting set of $F = \{F_1, ..., F_L\}$ at an OR node. After determining the minimal candidates of a multiple fault suspect set at the current stage, we evaluate the conditional probabilities of minimal candidates using Bayes' rule. Then, we invoke the single fault strategy TEAMS-S to isolate these candidates, and propagate multiple fault suspect set through the resulting diagnostic tree. Note that, using the fourth property of compact set notation, it is sufficient to generate and store multiple fault ambiguity group at the leaf nodes of this tree only. We repeat these procedures for each leaf node of the tree until: (1) the intersection of minimal candidates is not empty, i.e., the corresponding failure sources are definitely faulty, or (2) no test provides further information. The former corresponds to the case when the cardinality of one or more F_i in the ambiguity group is one.

After repairing/replacing the components isolated by Sure strategies, we apply additional tests, if necessary, to isolate the remaining failure sources. We explore three different approaches for the application of additional tests: (1) start from the root OR node of the diagnostic tree; (2) start from the first failed test in the path leading to the isolation of previous faults; (3) update the multiple fault suspect set at the leaf node by integrating previous test results using the fourth property of the compact set notation, removing repaired/replaced failure sources from the ambiguity group at the leaf node, and invoking Sure strategies for the updated ambiguity group. Sure 1-3 algorithms correspond to the first, second and third approaches for applying additional tests, respectively. These are presented in detail in [17].

The Sure1 diagnostic strategy is simple and the resulting diagnostic tree is very similar to the single fault diagnostic tree. However, the expected testing cost using this strategy is usually high. The expected testing cost using Sure2 diagnostic strategy is less than the first one, but the next test to be performed after repairing/replacing each failure source will be different. Furthermore, the diagnostic tree will change to a digraph (directed graph). The expected testing cost for the third approach is the smallest, but the size of the diagnostic tree will be considerably larger than the others. This is because the number of leaves of the diagnostic tree is the same as the number of distinguishable multiple-fault failure signatures. For example, in the worst case, i.e., when the test matrix B is diagonal, the number of leaves is 2^m. This is because there are 2^m possible multiple-fault failure signatures. But, the number of leaf nodes in Sure1 and Sure2 diagnostic strategies in this case are the same as in a single-fault strategy, i.e., $m + 1$.

One of the interesting features of Sure strategies is that the starting point for all three algorithms is the same tree as in a single fault strategy for the system under consideration. This is because the minimal candidate set for 2^S is $\{s_0, s_1, ..., s_m\}$. Therefore, these strategies isolate a single fault with the smallest average cost, while not making an error when multiple faults are present. Furthermore, in the case of very large systems, instead of generating all minimal candidates, we can generate minimal candidates of size less than a certain threshold, $\hat{L}$, and diagnose multiple faults up to that size.

Example 1.c: Figure 5, without (with) the dashed lines, shows the multiple fault strategy for the system in Example 1.a, based on Sure1(Sure2) algorithm, where A_i denotes the ambiguity group corresponding to the OR node i, and $A_1 = \Theta(1; \{s_0, s_1, s_2, s_3, s_4, s_5\}; \emptyset)$; $A_2 = \Theta(1; \{s_0, s_2, s_3\}; \{s_1, s_4, s_5\})$; $A_3 = \Theta(1; \{s_1, s_4, s_5\}; \emptyset)$; $A_4 = \Theta(1; \{s_0\}; \{s_1, s_2, s_3, s_4, s_5\})$; $A_5 = \Theta(1;\ \{s_2, s_3\};\ \{s_1, s_4, s_5\})$; $A_6 = \Theta(1;\ \{s_1, s_4\};\ \{s_2, s_3, s_5\})$; $A_7 = \Theta(2;\ \{s_1, s_4, s_5\},\ \{s_2, s_3, s_5\}; \emptyset)$; $A_8 = \Theta(1;\ \{s_2\};\ \{s_1, s_3, s_4, s_5\})$; $A_9 = \Theta(1;\ \{s_3\};\ \{s_1, s_4, s_5\})$; $A_{10} = \Theta(1;\ \{s_1\};\ \{s_2, s_3, s_4, s_5\})$; $A_{11} = \Theta(1;\ \{s_4\};\ \{s_2, s_3, s_5\})$; $A_{12} = \Theta(2; \{s_4, s_5\}, \{s_2, s_5\}; \{s_1, s_3\})$; $A_{13} = \Theta(3;\ \{s_1, s_4, s_5\},\ \{s_2, s_3, s_5\},\ \{s_1, s_3\};\ \emptyset)$; $A_{14} = \Theta(2;\ \{s_3\}, \{s_1, s_4\};\ \{s_2, s_5\})$; $A_{15} = \Theta(3;\ \{s_1, s_4, s_5\},\ \{s_1, s_3\}, \{s_2, s_5\}; \emptyset)$; $A_{16} = \Theta(2; \{s_1\}, \{s_2\}; \{s_3, s_4, s_5\})$; $A_{17} = \Theta(4; \{s_1, s_3\}, \{s_2, s_5\}, \{s_3, s_4, s_5\}, \{s_1, s_4, s_5\}; \emptyset)$

Note that the shaded parts of the tree are the same as those in the single fault diagnostic tree of Figure 2. The average testing cost for the optimal multiple fault strategy is $J = 2.411$, and the average testing cost for the first (Sure1) and second (Sure2) approaches using the diagnostic strategy of Figure 5 are $J = 2.715$ and $J = 2.616$, respectively.

Example 1.d: The Sure3 strategy for Example 1.a is shown in Figure 6, where $A_{18} = A_{20} = A_{24} = \Theta(1; \{s_0\}; \{s_1, s_2, s_3, s_4, s_5\})$; $A_{19} = \Theta(1; \{s_2\}; \{s_1, s_3, s_4, s_5\})$; $A_{23} = \Theta(1; \{s_4\}; \{s_2, s_3, s_5\})$; $A_{21} = A_{22} = A_{25} = \Theta(1; \{s_1\}; \{s_2, s_3, s_4, s_5\})$;

Note that the shaded and dashed parts of the tree in Figure 6 are the same as those in Figure 5. For this test strategy, the average test cost $J = 2.535$. In this example, we considered a block replacement strategy when no test gives further information, for example, see ambiguity groups A_{12} and A_{17}.

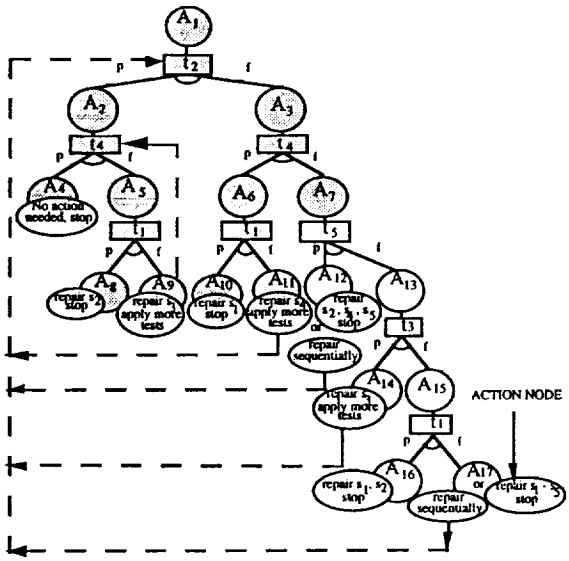

Figure 5: Sure1 and Sure2 Test Strategies for Example 1.a

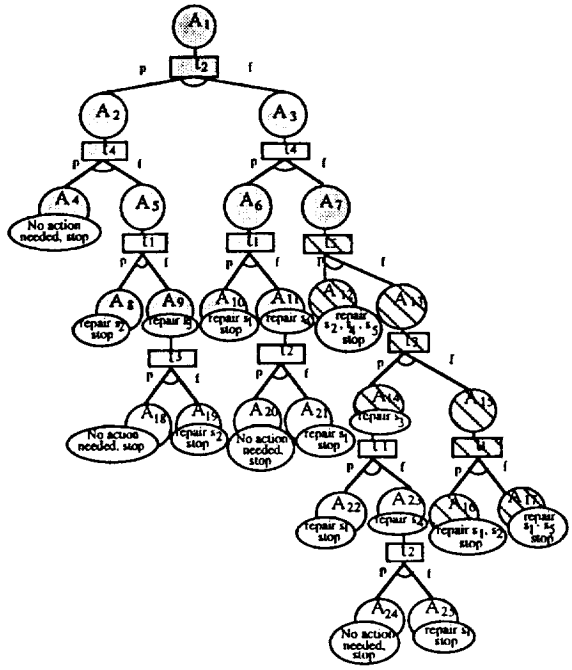

Figure 6: Sure3 Test Strategy for Example 1.a

6 Multiple Fault Testing Strategy in Systems with Redundancy (AND nodes)

In digraph models with AND nodes, the assumption that the failure signature of a multiple failure is the union of failure signatures of the corresponding individual failures is not valid. This is because the failures of multiple modules can propagate to the output of AND nodes, and therefore, generate a different failure signature. In these models, minimal faults and their failure signatures contain all the necessary information for multiple fault diagnosis [30].

In this section, we first present a top-down recursive algorithm to find all the minimal faults and their failure signatures; the minimal fault algorithm is presented in detail in [18]. Even though this algorithm can easily be extended to generate minimal faults with a limited size $\hat{L}$, **we present an efficient bottom-up procedure to generate minimal faults up to a limited size $\hat{L}$. This is because, in very large systems, it is efficient and practical to generate minimal faults up to a specified size $\hat{L}$ using the bottom-up procedure.**

Then, after generating minimal faults, we augment the binary test matrix B to include minimal faults, and extend Sure diagnostic strategies of the previous section to systems with redundancy.

6.1 Minimal Fault Algorithm

In order to generate the minimal faults and their failure signatures, we use the reachability analysis algorithm of [12] to build: (1) failure source-test dependency matrix B of dimension $m \times n$, which denotes the full-order dependency among single failure sources and tests, (2) failure source-AND node dependency matrix H of dimension $m \times z$, which denotes the full-order dependency among failure sources and AND node inputs and outputs, where $z = \sum_{j=1}^{K}(v_j + 1)$, (3) AND node-test dependency matrix E of dimension $K \times n$, which denotes the full-order dependency among AND nodes and tests, (4) AND node-AND node dependency matrix R of dimension $K \times z$, which denotes the full-order dependency among AND node outputs and AND node inputs and outputs, and (5) AND node-AND node reachability matrix Q of dimension $K \times K$, which denotes the full-order dependency between AND nodes and AND nodes by setting AND nodes' logic of u_k-out-of-v_k to 1-out-of-v_k in **the reachability analysis algorithm, i.e., AND nodes devolve into OR nodes.**

For notational convenience, given a binary matrix $X = [x_{ij}]$ of dimension $k_1 \times k_2$, we define Xr_i for $i = 1, ..., k_1$ as its ith row, and Xc_j for $j = 1, ..., k_2$ as its jth column.

6.1.1 Top-down approach

Using the binary matrices, the top-down minimal fault algorithm finds the minimal faults of the digraph model via the following steps (see [18] for details):

Step 1:
Because of the definition of minimal faults, the algorithm needs to process only those AND nodes $As \subseteq$ A for which there exists at least a path from the AND node to a test.

The algorithm sorts the AND nodes in As such that each AND node will be processed before any other AND node reachable from it. This step prevents the algorithm from performing the same operations twice.

The procedure for finding and sorting As is as follows: (1) the algorithm finds a subset of AND nodes $Ae \subseteq A$ such that each AND node $a_k \in Ae$ can be detected by at least a test, i.e., $Ae = \vee Ec_j$ for $j = 1, ..., n$; (2) using AND node-AND node reachability matrix Q, it finds the subset of AND nodes As that reach Ae, i.e., $As = \vee_{a_k \in Ae} Qc_k$; and (3) the algorithm sorts the AND nodes in As based on the number of AND nodes reaching them in ascending order. Note that the number of AND nodes reaching an AND node a_j is $\sum_{k=1}^{K} q_{kj}$.

Step 2:

For each AND node $a_k \in As$, the algorithm finds the set of failure sources and AND nodes that can reach the AND node inputs and output. Then, it removes single failures affecting the AND node from its input signatures, and generates the minimal combinations of AND nodes and failure nodes for each AND node $a_k \in As$ using one of the following three approaches: (1) minimal hitting set method using a breadth-first search [15, 16],(2) minimal hitting set method using a depth-first search, and (3) binary decision diagrams [2]. However, because of the small number of AND node inputs, i.e., v_k for $k = 1, ..., K$, usually 2 or 3, there is no significant difference in using any of these three approaches.

Note that we can consider a limit $\hat{L}$ for the number of failure sources in the minimal combinations of AND nodes. In the first and second approaches, those combinations with more than $\hat{L}$ failure sources are not expanded. In the third approach, at first the decision diagram is generated, and then, the combinations with more than $\hat{L}$ faults are eliminated. Furthermore, when v_k=2 for $k = 1, ..., K$, the problem of finding minimal combinations for each AND node reduces to the problem of finding the cross-products of failure signatures of AND node inputs [6].

Step 3:

After generating the minimal combinations for each AND node in As, the algorithm processes one AND node at a time. The subroutine for this part is a recursive function and, for simplicity, we call it MFG (Minimal Fault Generator). In order to find minimal faults for an AND node, say a_i, we call MFG for a_i. MFG replaces a_i with one of its minimal combinations. If this combination contains no AND nodes, MFG adds this combination to the set of minimal faults of AND node a_i only if it is not a superset of one of them. On the other hand, if the combination contains AND nodes, it selects one of the AND nodes from this combination, say a_j, and calls MFG again for a_j. This procedure continues until no AND node remains in that combination, or a previously processed AND node is selected, i.e., there exists a feedback loop containing the AND node. In the former case, MFG adds this combination to the set of minimal faults of the AND node, only if it is not a superset of one of them. In the latter case, if the failure of the combination can propagate to the output of the AND node, MFG ignores that AND node, and continues. Otherwise, it returns without doing any thing. This step prevents the algorithm from entering an infinite loop, when a cycle is encountered.

Note that we can consider a limit $\hat{L}$ for the number of faults in the minimal faults of AND nodes. In this case, at each iteration, MFG checks whether the number of faults in the AND nodes and failure nodes combination is greater than the limit $\hat{L}$ or not. If the number of failure nodes in this combination is greater than $\hat{L}$, it returns; otherwise, it expands the selected AND node as before.

In order to make the algorithm efficient, we employ the following Lemma:

Lemma 2: Let us assume that Xr is a vector of dimension z, and if $a_k \in As$, $X_r[k(l)]$=1 for $l = 0, ..., v_k$, otherwise $X_r[k(l)]$=0 for $l = 0, ..., v_k$. If $Hr_i \wedge Xr$ is equal to $Hr_j \wedge Xr$ and there exists a minimal combination $mc \in MC(a_k)$ and $s_i \in mc$, then $(mc - \{s_i\}) \vee \{s_j\} \in MC(a_k)$. Further, if there exists a minimal fault $mf \in MF(a_k)$ and $s_i \in mf$, then $(mf - \{s_i\}) \vee \{s_j\} \in MF(a_k)$.

Using this Lemma, before generating minimal combinations of each AND node, we find all the failure sources with the same failure signature in the H matrix. That is, we generate the set M=$\{M_1, M_2, ..., M_\alpha\}$ such that $M_l \subseteq$ S for $l = 1, ..., \alpha$ and $\forall s_i \in M_l$ have the same failure signatures in the binary matrix H. Using this approach, the failure sources that have the same effect on the AND nodes, i.e., the failure sources in series [6], or those in Gross feedback loops [12], are considered as a group of failures. Thus, instead of generating the minimal combinations and minimal faults for each AND node based on S, we generate them based on M only, i.e., minimal faults are subsets of M. After generating these sets, we expand the minimal faults of AND nodes based on M to generate the minimal faults based on S.

Step 4:

After generating the minimal faults of the AND nodes, the algorithm generates the minimal faults of the digraph model (MF_d). Firstly, using the AND node-test dependency matrix E, the algorithm removes the minimal faults of those AND nodes that cannot be detected by any test. Secondly, if a set of faults belongs to the set of minimal faults of two or more AND nodes, the algorithm considers only one of them. Then, using the binary

matrices, the algorithm generates the failure signatures of remaining faults. Note that the remaining faults may contain supersets, and because of the test points in the digraph model, a superset may/may not be a minimal fault of the digraph model. Thus, those supersets, which have the same failure signatures as the union of the failure signatures of their subsets, are removed.

6.1.2 Bottom-up Approach

The bottom-up approach can be used to generate minimal faults up to a limited size, say 2 or 3, in systems with as many as 10,000 failure sources and 1000 AND nodes. For clarity, let us assume that v_k=2 for $k = 1, ..., K$. In this algorithm, using the first step of the top-down procedure, we find a subset of AND nodes $As \subseteq A$ that should be processed. Then, using the failure source-AND node dependency matrix H, for each AND node $a_k \in As$, the algorithm finds the failure sources that can reach one of the AND node inputs, but cannot reach their outputs, i.e., $Sc_k(1), Sc_k(2)$ for $a_k \in As$. By finding the cross-products [6] between two sets $Sc_k(1)$ and $Sc_k(2)$, the algorithm generates minimal combinations of size 2. Then, using the failure source-AND node dependency matrix H and AND node-AND node binary matrix R, the failure signatures of these faults can be found and stored in a binary matrix B'. Using binary matrix B', the algorithm finds the failure sources that can reach one of the AND node inputs, but cannot reach their outputs, i.e., $Sc'_k(1), Sc'_k(2)$ for $a_k \in As$. By finding the cross-products between two sets $Sc'_k(1)$ and $Sc_k(2)$, $Sc_k(1)$ and $Sc'_k(2)$, and $Sc'_k(1)$ and $Sc'_k(2)$, the algorithm generates all minimal combinations of size 3, as well as some minimal combinations of size 4. This procedure is continued until either no failure can reach any of the AND node inputs, or all the minimal faults of the desired size are generated. After generating minimal combinations of size $\hat{L}$, using the fourth step of the top-down algorithm, minimal faults of size $\hat{L}$ of the digraph model are generated. **Note that, because of the presence of feedback loops and common elements in some paths in the digraph models, it is *not* efficient to use a bottom-up approach *to find all the minimal faults of a digraph.***

6.2 Extended Compact set Notation

After generating minimal faults and their failure signatures, we expand the binary matrix B with the minimal fault failure signatures. Thus, in systems with AND nodes, each row of the test matrix corresponds to a subset of S= $\{s_1, ..., s_m\}$. For notational simplicity, let us assume that the new test matrix contains $m_n = m + m_f$ rows, where m_f is the number of minimal faults. We define $\mathbf{W}=\{w_1, ..., w_{m_n}\}$, where $w_i = \{s_i\}$ for $i = 1, ..., m$, and $w_i \subseteq$ S for $m + 1 \leq i \leq m_n$.

After generating the binary test matrix, we extend the compact set notation of the previous section to systems with redundancy. In this case, the ambiguity group at each OR node of the AND/OR graph is based on W, i.e., the F_i for $i = 1, ..., L$ and G are subsets of $W=\mathbf{W} \vee \{w_0\}$ $= \{w_0, w_1, ..., w_{m_n}\}$, where $\{w_0\} = \{s_0\}$ and

$$\Theta(L; F_1, F_2, ..., F_L; G) = \{X \subseteq W | \\ X \wedge F_i \neq \emptyset \text{ for } i = 1, ..., L, \text{ and } \mathcal{F}(X) \wedge G = \emptyset\}$$

where $\mathcal{F}(X) = \vee_{w_i \subseteq S(X)} w_i$, and $S(X) \subseteq S$ is the set of all failure sources in $w_j \in X$, i.e., $S(X) = \{s_i | \forall w_j \in X \text{ and } s_i \in w_j\}$.

6.3 Extended Sure Strategies

In order to derive the Sure diagnostic strategy, we need to generate the minimal candidates at each iteration. Note that, Lemma 1 is not valid for minimal faults in a system with redundancy. This is because w_j for $j = 1, ..., m_n$ are not independent, and because of the AND nodes, the failure signature of a set of components that has some thing in common with the F_i's is consistent with the failed tests, but it may be inconsistent with the passed tests. In this case, the set of minimal candidates of a multiple fault ambiguity group is generated using the following Lemma [30].

Lemma 3: The minimal set of a multiple fault ambiguity group $A = \Theta(L; F_1, ..., F_L; G)$ for a system with redundancy is $I(A)=\{X | X \in I(H(F))$ and $\mathcal{F}(X) \wedge G = \emptyset\}$, where $F = \{F_1, ..., F_L\}$. That is, the minimal set of a multiple fault ambiguity group contains only those elements of the minimal hitting set F that are consistent with the set of good components, G.

In addition, the *one-for-sure* condition of previous section should be generalized as follows:

Lemma 4: If the cardinality of any F_i is one, all the failure sources in $w_j \in F_i$ are faulty, and if the cardinality of F_i is greater than one, all the failure sources in $\wedge_{w_j \in F_i} w_j$, are definitely faulty. Evidently, these two conditions can be combined as follows: all the failure sources in $\wedge_{w_j \in F_i} w_j$, for $i = 1, ..., L$, are definitely faulty.

Further, we can use the following two Lemmas to update the ambiguity groups at each OR node.

Lemma 5: Let us assume that we repaired definitely failed components $X \subseteq S$, and that there exists a $w_j \in G$ such that $|w_j - X| = 1$ and $s_k = |w_j - X|$. Then, s_k is good and should be added to the good component subset G.

Lemma 6: If we repair definitely failed components $w_i = \{s_i\}$, and there exist a w_j such that $s_i \in w_j$, then w_j should be added to the good component subset G.

In summary, the Sure diagnostic strategies for systems with redundancy is as follows:

- the ambiguity group at each OR node of the AND/OR graph is represented based on W (rather than S),
- minimal candidates are generated based on Lemma 3,
- definitely failed components at the leaf nodes are found using Lemma 4, and
- the ambiguity groups at the leaf nodes are updated based on Lemmas 5 and 6.

Example 1.e: Consider the digraph model in Figure 7. This digraph model differs from the one in Figure 1 in that we have added an AND node a_1. The minimal fault for this digraph model is w_6=$\{s_1, s_3\}$ (see Table 2). Figure 8 without (with) the dashed lines, shows the multiple fault strategy for this system, based on Sure1 (Sure2) diagnostic strategy, and A_i denotes the ambiguity group corresponding to the OR node i, and $A_1 = \Theta(1; \{w_0, w_1, w_2, w_3, w_4, w_5, w_6\}; \emptyset)$; $A_2 = \Theta(1; \{w_0, w_2, w_3\}; \{w_1, w_4, w_5, w_6\})$; $A_3 = \Theta(1; \{w_1, w_4, w_5, w_6\}; \emptyset)$; $A_4 = \Theta(1; \{w_0\}; \{w_1, w_2, w_3, w_4, w_5, w_6\})$; $A_5 = \Theta(1; \{w_2, w_3\}; \{w_1, w_4, w_5, w_6\})$; $A_6 = \Theta(1; \{w_1, w_4\}; \{w_2, w_3, w_5, w_6\})$; $A_7 = \Theta(1; \{w_1, w_4, w_5, w_6\}, \{w_2, w_3, w_5, w_6\}; \emptyset)$; $A_8 = \Theta(1; \{w_2\}; \{w_1, w_3, w_4, w_5, w_6\})$; $A_9 = \Theta(1; \{w_3\}; \{w_1, w_4, w_5, w_6\})$; $A_{10} = \Theta(1; \{w_4\}; \{w_1, w_2, w_3, w_5, w_6\})$; $A_{11} = \Theta(1; \{w_1\}; \{w_2, w_3, w_5, w_6\})$; $A_{12} = \Theta(1; \{w_2, w_5\}, \{w_4, w_5\}; \{w_1, w_3, w_6\})$; $A_{13} = \Theta(1; \{w_1, w_3, w_6\}, \{w_2, w_3, w_5, w_6\}, \{w_1, w_4, w_5, w_6\}; \emptyset)$; $A_{14} = \Theta(1; \{w_3\}, \{w_1, w_4\}; \{w_2, w_5, w_6\})$; $A_{15} = \Theta(1; \{w_1, w_3, w_6\}, \{w_2, w_5, w_6\}, \{w_1, w_4, w_5, w_6\})$; $A_{16} = \Theta(1; \{w_1\}, \{w_2\}; \{w_3, w_4, w_5, w_6\})$; $A_{17} = \Theta(1; \{w_1, w_3, w_6\}, \{w_2, w_5, w_6\}, \{w_1, w_4, w_5, w_6\}, \{w_3, w_4, w_5, w_6\})$.

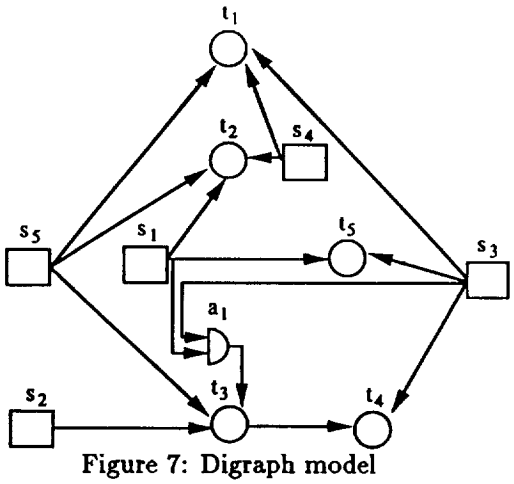

Figure 7: Digraph model

Note that $A_{14} = \Theta(1; \{w_3\}, \{w_1, w_4\}; \{w_2, w_5, w_6\})$; i.e., $F_1 = \{w_3\}$, $F_2 = \{w_1, w_4\}$, and $G = \{w_2, w_5, w_6\})$. Therefore, $w_3 = \{s_3\}$ is definitely faulty. After repairing s_3, using Lemma 5; i.e., $w_6 \in G$, $w_1 = \{s_1\}$ is good and should be added to the list of good components G. Thus, by eliminating w_1 from F_2, we conclude that w_4 is definitely faulty; the cardinality of F_2 is one. After repairing w_4, G = S, and there is no need to apply additional tests. The average testing cost for Sure1 and Sure2 diagnostic strategy using the diagnostic strategy of Figure 7, are $J = 2.603$ and $J = 2.505$, respectively.

FAILURE SOURCES	TESTS $t_1\ t_2\ t_3\ t_4\ t_5$ TEST COSTS c_j 1 1 1 1 1
$w_1 = \{s_1\}$	0 1 0 0 1
$w_2 = \{s_2\}$	0 0 1 1 0
$w_3 = \{s_3\}$	1 0 0 1 1
$w_4 = \{s_4\}$	1 1 0 0 0
$w_5 = \{s_5\}$	1 1 1 1 0
$w_6 = \{s_1, s_3\}$	1 1 1 1 1

Table 2: Test Matrix and Test Costs for Example 1.e

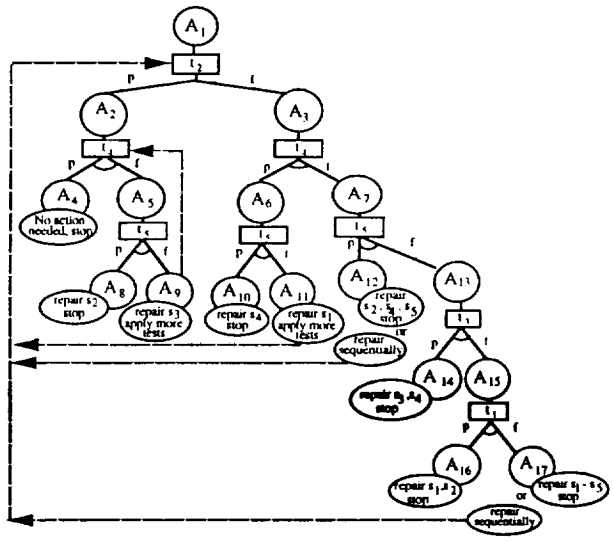

Figure 8: Multiple Fault Diagnostic Strategy for Example 1.e

Example 1.f: The Sure3 strategy for Example 1.e is shown in Figure 9, where $A_{18} = A_{20} = \Theta(1; \{w_0\}; \{w_1, w_2, w_3, w_4, w_5, w_6\})$; $A_{19} = \Theta(1; \{w_2\}; \{w_1, w_3, w_4, w_5, w_6\})$; $A_{21} = \Theta(1; \{w_4\}; \{w_1, w_2, w_3, w_5, w_6\})$.

Note that the dashed parts of the tree in Figure 9 are the same as those in Figure 8. For this test strategy, the average test cost $J = 2.492$. In this example, we considered a block replacement strategy when no test gives further information, for example, see ambiguity groups

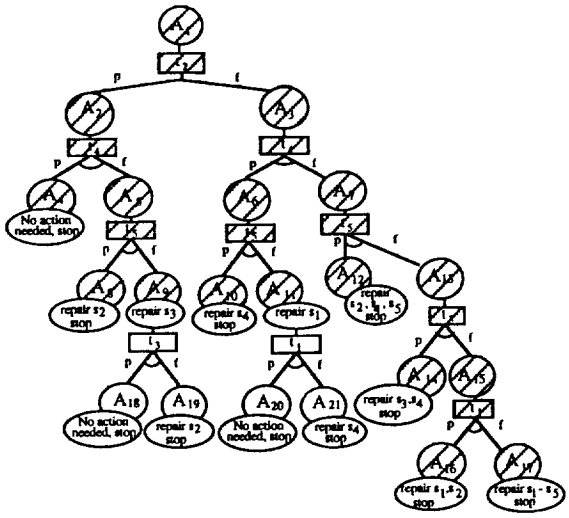

Figure 9: Sure3 Test Strategy for Example 1.e

A_{12} and A_{17}.

7 On-line Multiple Fault Diagnosis Strategies

In this section, we consider the problem of designing an *on-line* (interactive or dynamic) diagnostic strategy to isolate multiple failures in a physical system. That is, instead of generating the entire diagnostic tree, the on-line strategy only suggests the next test to be applied given the outcomes of previously applied tests. Our approach is to employ concepts from information theory and Lagrangian relaxation to solve this problem.

At each stage of diagnosis, we consider a set of available tests TA which can provide some information about the system. Initially, TA contains all tests except those that can detect all or no faults. Then, we recommend a test using a local, step-by-step optimization algorithm developed by Johnson [19]. In this approach, a test t_k from the set of available tests TA is selected, if it maximizes the information gain per unit cost of the test:

$$k = \arg\max_j \{\frac{IG(A,t_j)}{c_j}\} \tag{2}$$

where A is the ambiguity group at the current stage of diagnosis, and $IG(A,t_j)$ is the information gain given by:

$$IG(A,t_j) = -\{P'(A_{jp})\log_2 P'(A_{jp}) + P'(A_{jf})\log_2 P'(A_{jf})\} \tag{3}$$

In (3), $\{A_{jp}, A_{jf}\}$ are the subsets of the ambiguity group A corresponding to pass and fail outcomes of test t_j such that $A_{jp} \vee A_{jf} = A$, and $P'(A_{jp}) = P(A_{jp})/P(A)$, $P'(A_{jf}) = P(A_{jf})/P(A)$ are the conditional probabilities of the pass and fail outcomes of test t_j, and $P(A_{jp})$ and $P(A_{jf})$ are the probabilities of ambiguity groups A_{jp} and A_{jf}, respectively.

In general, $P(A = \Theta(L; F_1, F_2, ..., F_L; G))$, needed in the evaluation of information gain, can be computed as follows:

$$P(A) = P((\cup_{i=1}^{\hat{m}} \Psi_i) \cap \bar{G}) \tag{4}$$

where $\hat{m}$ is the number of minimal candidates, and Ψ_i for $i = 1, ..., \hat{m}$ are the minimal candidates of the ambiguity group A and $\bar{}$ denotes the logical NOT operator[20][23]. In addition, for notational clarity, we use the same notation for expressing a set and its Boolean expression. Furthermore, we use $\cap$ and $\cup$ as Boolean product and sum, i.e., conjunction (AND) and disjunction (OR) of at least two Boolean expressions, respectively [20][3]. We define $G_s \subseteq G$ as a set containing single failures of G, and G_w as its complement within G, i.e., $G_w = G_s^c = G - G_s$. Thus, by expanding (4) and using the associative law of Boolean algebra [21], we have:

$$\begin{aligned} P(A) &= P((\cup_{i=1}^{\hat{m}} \Psi_i) \cap \bar{G}_s \cap \bar{G}_w) \\ &= P(((\cup_{i=1}^{\hat{m}} \Psi_i) \cap \bar{G}_s) \cap \bar{G}_w) \end{aligned} \tag{5}$$

where $\bar{}$ denotes the logical NOT operator[20][23].
By defining $\Phi_i = \Psi_i \cap \bar{G}_s$ for $i = 1, ..., \hat{m}$, (5) reduces to:

$$P(A) = P((\cup_{i=1}^{\hat{m}} \Phi_i) \cap \bar{G}_w) \tag{6}$$

This further reduces to:

$$\begin{aligned} P(A) &= P(\cup_{i=1}^{\hat{m}} \Phi_i) - P((\cup_{i=1}^{\hat{m}} \Phi_i) \cap G_w) \\ &= P(\cup_{i=1}^{\hat{m}} \Phi_i) - P(\cup_{i=1 \ \& \ w_j \in G_w}^{\hat{m}} (\Phi_i \cap w_j)) \end{aligned} \tag{7}$$

Since $\Phi_i = \Psi_i \cap \bar{G}_s$, the second term of (7) should be considered only for those w_j that do not have any thing in common with G_s. Further, it is sufficient to evaluate the second term of (7) for $w_j \in I(G_w)$, i.e., irreducible set of G_w. This is because, if $w_k \subset w_j$, any set satisfying the Boolean expression $(\Phi_i \cap w_j)$ will satisfy $(\Phi_i \cap w_k)$.

The problem in (7) is equivalent to the problem of finding the probability of a sum of non-disjoint sets. This problem is known as the sums of products problem, and its computational complexity is NP-hard [22]. Veeraraghavan et al [23] considered the sum (product) of products (sums) problem and proposed an efficient Boolean algebraic algorithm, the so-called GKG-VT algorithm, for its solution. In this algorithm, the probability of the union of a set of events can be evaluated using the following equation [23]:

[3] This is in contrast to $\vee$ and $\wedge$, which are used to denote Boolean conjunction and disjunction of two sets.

$$\mathrm{P}\{\cup_i^p IP_i\} = \mathrm{P}\{(IP_1) \cup (\overline{IP}_1 IP_2) \cup \ldots (\overline{IP}_1 \ldots \overline{IP}_{p-1} IP_p)\} \quad (8)$$

Since these resulting sets are disjoint, their probabilities are added to obtain the probability of the desired event. Thus, the first term in (7) can be evaluated as follows:

$$\begin{aligned} P(((\cup_{i=1}^{\hat{m}} \Phi_i)) &= P(\cup_{i=1}^{\hat{m}} \Psi_i | \bar{G}_s) P(\bar{G}_s) \quad (9) \\ &= P(\cup_{i=1}^{\hat{m}} \Psi_i | \bar{G}_s) \prod_{s_k \in G_s} (1 - p(s_k)) \end{aligned}$$

where $P(\cup_{i=1}^{\hat{m}} \Psi_i | \bar{G}_s)$ is the probability of sum of disjoint products Ψ_i in $\mathbf{S} - G_s$ domain, i.e., the number of total variables reduces from m to $m - |G_s|$. In addition, in digraph models without AND nodes, $G_w = \emptyset$. Thus, the second term in (7) is zero, and P(A) reduces to (9). Furthermore, in these systems, since the set of minimal candidates is the minimal hitting set for the set $F = \{F_1, F_2, ..., F_L\}$ (see Lemma 1), (9) can be written as:

$$P(A) = P(\cap_{i=1}^{L} F_i | \bar{G}_s)) \prod_{s_k \in G_s} (1 - p(s_k)) \quad (10)$$

Thus, in systems having no redundancy, instead of evaluating the probability of ambiguity group A using minimal candidates at each stage of diagnosis, we can directly evaluate this probability using $F = \{F_1, ..., F_L\}$. Furthermore, since the GKG-VT algorithm evaluates the probability of $\{\cup_i^{p+1} IP_i\}$ (sum of products) and $\{\cap_i^{p+1} IP_i\}$ (product of sums) sequentially, i.e.,:

$$\mathrm{P}\{\cup_i^{p+1} IP_i\} = \mathrm{P}\{\cup_i^p IP_i\} + \mathrm{P}\{(\overline{IP}_1 \ldots \overline{IP}_p IP_{p+1})\} \quad (11)$$

$$\mathrm{P}\{\cap_i^{p+1} IP_i\} = \mathrm{P}\{\cap_i^p IP_i\} - \mathrm{P}\{(IP_1 \ldots IP_p \overline{IP}_{p+1})\} \quad (12)$$

the probability of $A_{jf} = \Theta(L+1; F_1, \ldots, F_L, TS_j \wedge G^c; G)$ and $A_{jp} = \Theta(L; (F_1 \wedge TS_j^c), \ldots, (F_L \wedge TS_j^c); (G \vee TS_j))$ are:

$$\begin{aligned} \mathrm{P}(A_{jf}) &= (P(\cap_{i=1}^{L} F_i | \bar{G}_s) - \mathrm{P}(F_1 \ \ldots \ F_L \ \overline{TS_j \wedge G^c} | \bar{G}_s)) \\ &\quad \prod_{s_k \in G_s} (1 - p(s_k)) \\ &= \mathrm{P}(A) - \mathrm{P}(F_1 \ \ldots \ F_L \ \overline{TS_j \wedge G^c} | \bar{G}_s) \prod_{s_k \in G_s} (1 - p(s_k)) \\ \mathrm{P}(A_{jp}) &= \mathrm{P}(A) - \mathrm{P}(A_{jf}) \\ &= \mathrm{P}(F_1 \ \ldots \ F_L \ \overline{TS_j \wedge G^c} | \bar{G}_s) \prod_{s_k \in G_s} (1 - p(s_k)) \end{aligned}$$

Note that when $s_0 \in F_1$ (see the fifth property of compact set notation), we should split F_1 into two disjoint sets $\{s_0\}$ and $F_1 - \{s_0\}$.

One of the advantages of this approach, compared to the one in [24], [25], is that the probability of an ambiguity group at the current stage of diagnosis is evaluated using the probability of ambiguity group at the previous stage. Furthermore, using this recursive approach, the probability of any hypothesis at the current stage of diagnosis can be evaluated. The computational complexity of this approach is strongly related to the structure of the B matrix.

In summary, at each stage of diagnosis, for a given set G_s, we generate the set of disjoint events for $F_1 \ F_2 \ \ldots \ F_L \ \overline{TS_j \wedge G^c}$, evaluate $P(A_{jf})$ and $P(A_{jp})$, and recommend a test with the highest information gain. Based on the test outcome, we update the set of available tests TA, i.e., we remove the recommended tests and those tests that do not give any information. This procedure is continued until: (1) at least a failure source is isolated, or (2) no test gives further information.

The former corresponds to the case when the cardinality of one or more F_i in the ambiguity group A is one. After repairing/replacing the failure sources in F_i's with cardinality one, we update the current ambiguity group and the set of available tests as follows: (i) add repaired/replaced components to the set of good components G, (ii) remove F_i's containing at least a repaired/replaced component from ambiguity group A. If all the F_i's are removed, we set the current ambiguity to A=$\Theta(1; F_1 = S - G; G)$, and (iii) update the set of available tests TA to all tests except previously applied passed tests and those tests that do not give any new information, i.e., those tests t_j such that $TS_j \wedge G^c$ is either an empty set, or a superset of one of F_i (see the second property of compact set notation). This procedure is continued until the set of good components G contains all the elements.

In the second case, we can select either block or sequential replacement. In block replacement, we repair all the suspected faults, i.e., S-G, and stop testing. In sequential replacement, we repair/replace most likely candidates and continue testing. The problem of finding the most likely candidates is as follows:

$$\begin{aligned} \text{maximize} \quad & \textstyle\prod_{i=1}^{m} p(s_i)^{x_i} (1 - p(s_i))^{(1-x_i)} && (13) \\ \text{subject to} \quad & \textstyle\sum_{i=1}^{m} F_{li} x_i \geq 1 \ ; \quad l = 1, ..., L && (14) \\ & \textstyle\sum_{i=1}^{m} \Gamma_{ki} x_i \leq |w_k| - 1 \ ; \quad w_k \in G_w && (15) \\ & x_i \in (0,1) \ ; \quad i = 1, ..., m && (16) \end{aligned}$$

where $F_{li} = 1$ if $s_i \in F_l$; otherwise $F_{li} = 0$; $\Gamma_{ki} = 1$ if $s_i \in w_k$; otherwise $\Gamma_{ki} = 0$. Constraints (14) and

(15) ensure that the most likely candidates are consistent with the failed and passed tests. Furthermore, using Lemmas 1 and 6, it is sufficient to consider (15) for $\hat{G}_w=\{w_k|w_k \in I(G_w)$ and $w_k \wedge G_s = \emptyset\}$, where $I(G_w)$ is the irreducible set of G_w. By taking the logarithm of the objective function in (13), an equivalent objective function is:

$$\text{maximize} \quad \sum_{i=1}^{m} x_i \log \frac{p(s_i)}{(1-p(s_i))} \qquad (17)$$

subject to (14), (15) and (16). Thus, the problem of finding the most likely candidates for redundant systems reduces to a *generalized set-covering* problem [26], [27]. Furthermore, in systems having no redundancy, $G_w = \emptyset$. Consequently, (15) is eliminated, and the problem reduces to a traditional set-covering problem [31].

The generalized set covering problem is solved via a Lagrangian relaxation technique. In this technique, the constraints (14) and (15) are relaxed via Lagrange multipliers. The solution of the relaxed problem is an upper bound for the covering problem. The multipliers are updated iteratively via subgradient optimization to minimize this upper bound. In addition, the upper bound and the relaxed solution can be used to develop the best feasible solution for the generalized set covering problem. A nice feature of the relaxation approach is that the difference between the upper bound and the best feasible solution, termed the approximate duality gap, provides a measure of suboptimality of the feasible solution. The details may be found in [30, 26, 27].

Using the solution of the generalized set-covering problem, the most likely candidate is $S_m = \{s_i|x_i = 1\}$. Note that, usually $p(s_i) < 0.5$ for $i = 1, .., m$. In this special case, the most likely candidate is one of the minimal candidates of the current ambiguity group. After repairing the most likely candidates, we update the ambiguity group and continue testing.

In order to solve multiple fault isolation problems in larger systems with as many as 10,000 failure sources, we employ the following simplified approach to compute information gain in (3). If the ambiguity group A at the current stage of diagnosis contains more than one single failure source; i.e., intersection of F_i's contains more than one fault, we select a test t_k that maximizes the information gain per unit cost of the test based on a single fault ambiguity group. That is, in this case, $IG(A, t_j)$ in (2) is the information gain based on single fault assumption, $p'(A_{jp})$ and $p'(A_{jf})$ in (3) are the conditional probabilities of the pass and fail outcomes of test t_j based on the single fault assumption in ambiguity group A. However, based on the test outcome, we update the multiple fault ambiguity group (see properties 1 and 2 of the compact set notation). This procedure is continued until: (1) at least a failure source is isolated, i.e., the cardinality of one or more of F_i's is one, (2) no test gives further information, (3) the cardinality of intersection of F_i's is one, i.e., there exists a set of masking sets for the single fault in that intersection, or (4) the set of good components G contains all the components, $G = \mathbf{S}$.

The first and second cases are the same as those in the previous approach. In the third case, we recommend a test based on a measure of information content in [28], and continue testing until the first or second condition is reached. In the fourth case, since all the components are good, no further action is needed. This approach has less computational complexity, but higher testing cost compared to the previous approach, based on sum (product) of disjoint products (sums).

In addition to a set of comprehensive synthetic problems, we have applied the algorithms presented in this paper to several real-world systems. These include: (1) the Space Shuttle Main Propulsion System with 7271 failure sources and 1292 AND nodes [6], (2) the F18-Flight control system with 148 failure sources and 78 AND nodes [29] with failure sources limited to singletins and doubletons, (3) the anticollision light control system of the Sea Hawk helicopter with 51 failure sources and 55 tests, (4) the stabilator system of the Black Hawk helicopter with 238 failure sources and 834 tests, and (5) the engine torque monitoring system used in CH-53E helicopter with 116 failure sources and 75 tests. In the latter three cases, static and dynamic multiple fault diagnostic strategies subject to various constraints on available resources, setup operations, and initial failure symptoms have been implemented, along with interfaces to interactive electronic technical manuals and multi-media documentation.

8 Conclusion

In this paper, we considered the problem of constructing near-optimal test sequencing algorithms for diagnosing multiple faults in systems modeled as digraphs. This problem involves two sequential steps: (1) generation of a binary test matrix, and (2) design of a multiple-fault testing strategy that unambiguously isolates the multiple failures with minimum expected testing cost (time).

In systems without redundancy, a binary test matrix denoting the full-order dependency among single failures and the tests forms the basis for diagnosing single, as well as multiple faults in the system. In order to diagnose multiple faults in systems with redundancy, this binary test matrix is augmented to capture the failure signatures of minimal-faults. Using a top-down recursive procedure,

we developed an algorithm to find all the minimal faults and their failure signatures in redundant systems, and using a bottom-up procedure, we presented an efficient algorithm to find minimal faults up to a limited size.

After generating the binary test matrix, the problem is to design a practical multiple fault test sequencing algorithm. The computational and storage complexity of an optimal multiple fault strategy are super-exponential in the number of failure sources, m. We presented several near-optimal algorithms that provide a trade-off between optimality and computational complexity. Firstly, we extended the single-fault strategy of our previous work [1, 10] to diagnose multiple faults by successively isolating the potential single-fault candidates, then double-fault candidates, and so on. This is one of the simplest multiple fault strategies that one can use. In this approach, the storage complexity at each OR node of the AND/OR graph is the same as that in a single fault strategy.

We then extended the single fault sequential testing strategies to a class of Sure strategies. The basic idea of these strategies is to find one or more definitely failed components, while not making an error when other coexisting faults are present. We explored three different approaches for the application of additional tests, resulting in Sure1-3 strategies.

Some of the advantages of using Sure strategies are: (1) the inherent combinatorial explosion that occurs in generating an optimal multiple fault strategy is reduced substantially, (2) the first iteration of the Sure strategies results in the same tree as in the single fault (minimal fault) strategy for the system without (with) redundancy, and therefore, these strategies isolate a single fault (minimal fault) with the smallest average cost, while not making an error when multiple faults are present. Computational complexity of this approach is strictly related to the structure of the system, i.e., the test matrix B.

In order to eliminate the problems associated with the size of the complete diagnostic strategy, the test strategy can be generated "on-line". That is, instead of generating the entire diagnostic tree, the interactive strategy only suggests the next test to be applied given the outcomes of previously applied tests. We employed concepts from information theory and Lagrangian relaxation to generate several on-line diagnostic strategies. In these strategies, at each stage of diagnosis, a test with the highest information gain is recommended. The computation of information gain associated with a test requires the probabilities of ambiguity groups corresponding to pass and fail outcomes of the test. An efficient computational approach based on sum (product) of disjoint products (sums) is used to evaluate these probabilities. However, the computational complexity of this approach is strongly related to the structure of the binary test matrix B and previously applied tests. In order to derive a practical (albeit suboptimal) on-line diagnostic strategy capable of diagnosing multiple faults in large scale systems, we estimated these probabilities via: (1) the probabilities of single failures at the ambiguity group, i.e., $\wedge_{1\leq i\leq L} F_i$, and (2) the probability of ambiguity group based on all the suspected faults, i.e., $\Theta(1; F_1 = \{S - G\}; G)$ [28]. Note that, these estimates constitute the lower and upper bounds for the probability of ambiguity group. We expect to investigate tighter bounds, as well as other measures for recommending a test, and compare their efficiencies in our future efforts.

References

[1] K.R. Pattipati and M.G. Alexandridis, "Application of heuristic search and information theory to sequential fault diagnosis," *IEEE Transactions on Systems, Man, and Cybernetics*, 20(4):872–887, July/August 1990.

[2] A. Rauzy, "New algorithm for fault trees analysis," *Reliability Engineering and System Safety*, 40:203–211, 1993.

[3] J. Vatn, "Finding minimal cut sets in a fault tree," *Reliability Engineering and System Safety*, 36:59–62, 1992.

[4] K. S. Brown, "Evaluating fault trees (and and or gates only) with repeated events," *IEEE Transactions on Reliability*, 39:226–235, June 1990.

[5] W. G. Schneeweiss and K. S. Brown, "Comments on: Evaluating fault trees (and and or gates only) with repeated events," *IEEE Transactions on Reliability*, 40:8–10, April 1991.

[6] D.L. Iverson and F. A. Patterson-Hine, "Digraph reliability model processing advances and applications," *Proceedings of the AIAA Computing in Aerospace Conference*, 1993.

[7] R. Davis, "Diagnostic reasoning based on structure and behavior," *Artificial Intelligence*, 24:347–410, 1984.

[8] R. Davis, "Retrospective on diagnostic reasoning based on structure and behavior," *Artificial Intelligence*, 59:149–157, 1993.

[9] J. de Kleer and B.C. Williams, "Diagnosing multiple faults," *Artificial Intelligence*, 32:97–130, 1987.

[10] K.R. Pattipati, S. Deb, M. Dontamsetty, and A. Maitra, "START: System Testability Analysis and Research Tool," *IEEE Aerospace and Electronic Systems Magazine*, pp.13–20, January 1991.

[11] M. Shakeri, K.R. Pattipati, V. Raghavan, , and S. Deb, "Near-optimal sequential testing algorithms for multiple fault isolation," *IEEE International Conference on Systems, Man and Cybernetics*, pp.1908–1914, 1994.

[12] K.R. Pattipati, V. Raghavan, M. Shakeri, S. Deb, and R. Shrestha, "TEAMS: Testability Engineering And Maintenance System," *American Control Conference*, pp.1989–1996, June 1994.

[13] D. P. Bertsekas, *Dynamic Programming: Deterministic and Stochastic Models*, Prentice-Hall, Englewood Cliffs, NJ, 1987.

[14] D.B. Grunberg, J.L. Weiss, and J.C. Deckert, "Generation of optimal and suboptimal strategies for multiple fault isolation," *Technical report TM-248*, 1987.

[15] R. Reiter, "A theory of diagnosis from first principles," *Artificial Intelligence*, 32(1):57–95, Apr. 1987.

[16] R. Greiner, B.A. Smith, and R. W. Wilkerson, "A correction to the algorithm in Reiter's theory of diagnosis," *Artificial Intelligence*, 41:79–88, 1989/90.

[17] M. Shakeri, K.R. Pattipati, V. Raghavan, and S. Deb, "Sure strategies for multiple fault isolation," *Technical Report TR-94-3*, June 1994.

[18] M. Shakeri, K.R. Pattipati, V. Raghavan, and A. Patterson-Hine, "Multiple fault isolation in redundant systems," to be presented at *IEEE International Conference on Systems, Man and Cybernetics*, Vancouver, British Columbia, Canada, Oct 1995.

[19] R. A. Johnson, "An information theory approach to diagnosis," *Proc. 6th Symp. Rel. Quality Contr.*, pp.102–109, 1960.

[20] K. D. Heidtmann, "Smaller sums of disjoint products by subproduct inversion," *IEEE Trans. Reliability*, 38:305–311, Aug. 1989.

[21] S. Koppelberg, *Hand Book of Boolean Algebra*, Elsevier Science, B.V., 1989.

[22] M. O. Ball and J. S. Provan, "Disjoint products and efficient computation of reliability," *Operations Research*, 36:703–715, Oct 1988.

[23] M. Veeraraghavan and K.S. Trivedi, "An improved algorithm for symbolic reliability analysis," *IEEE Transactions on Reliability*, 40(3):347–358, August 1991.

[24] F. Pipitone, "The FIS electronics troubleshooting system," *IEEE expert*, pp.68–76, 1986.

[25] R. Liu, *Testing and Diagnosis of Analog Circuits and Systems*, Van Nostrand Reinhold, 1991.

[26] J.E. Beasley, "A Lagrangian heuristic for set-covering problems," *Naval Research Logistics*, 37:151–163, 1990.

[27] J.E. Beasley, "A Lagrangian heuristic for set-covering problems," *European Journal of Operational Research*, 58:293–300, 1992.

[28] S. Deb, K.R. Pattipati, V. Raghavan, M. Shakeri, and R. Shrestha, "Multi-signal flow graphs: A novel approach for system testability analysis and fault diagnosis," *IEEE Aerospace and Electronic Systems Magazine*, 10(5):14–25, May 1995.

[29] S.A Doyle, J. B. Dugan, and A. Patterson-Hine, "A quantitative analysis of the F18 flight control system," *Proceedings of the AIAA Computing in Aerospace 9*, pp. 668-675, October 1993.

[30] M. Shakeri, "Advances in Fault Diagnosis, Testability and Reliability Analysis of Large-Scale Systems," Ph.D. Thesis, Dept. of Electrical and Systems Engineering, University of Connecticut, in preparation.

[31] M. M. Syslo, N. Deo, and J. S. Kowalik, *Discrete optimization algorithms with PASCAL programs*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1983.

[32] H.H. Dill, "Test program sets - A new approach," *IEEE Autotestcon, San Antonio, Texas*, pp. 63-69, 1990.

[33] C.C. Chang, and C.C. Yu, "On-line fault diagnosis using the signed directed graph," *Ind. Eng. Chem. Res.*, 29:1290-1299, 1990.

[34] W.S. Lee, D.L. Grosh, F.A. Tillman, and C.H. Lie, "Fault Tree Analysis, Methods, and Applications-a Review," *IEEE Transactions on Reliability*, 34(3):194-203, 1985.

[35] J.H. Sheppard, and W.R. Simpson, "Multiple failure diagnosis," *IEEE Autotestcon*,Anaheim, CA, pp.381-389, 1994.

Multiple Fault Isolation in Redundant Systems*

M. Shakeri, K.R. Pattipati and V. Raghavan
U-157, Department of Electrical and Systems Engineering,
University of Connecticut, Storrs, CT 06269-3157
E-mail: krishna@sol.uconn.edu

F.A. Patterson-Hine and D.L. Iverson
NASA-Ames Research Center, Mail Stop 269-4,
Moffett Field, CA 94035-1000
E-mail: Ann_Patterson-Hine@styx.arc.nasa.gov

Abstract

We consider the problem of sequencing tests to isolate multiple faults in redundant (fault-tolerant) systems with minimum expected testing cost (time). It can be shown that single faults and minimal faults, i.e., minimum number of failures with a failure signature different from the union of failure signatures of individual failures, together with their failure signatures, constitute the necessary information for fault diagnosis in redundant systems. In this paper, we develop an algorithm to find all the minimal faults and their failure signatures. Then, we extend the Sure diagnostic strategies [1] of our previous work to diagnose multiple faults in redundant systems. The proposed algorithms and strategies are illustrated using several examples.

1 Introduction

Diagnosis is fundamentally a process of identifying the cause of a malfunction by observing its effects at various monitoring points in a system. Fault diagnosis in large-scale systems that are products of modern technology present formidable challenges to manufacturers and users. This is due to the large number of failure sources and the need to quickly isolate and rectify such failures with minimal down time. In addition, for redundant systems and systems with little or no opportunity for repair or maintenance during the operation (e.g., Hubble telescope, space station), the assumption of at most a single failure in the system between consecutive maintenance actions is unrealistic.

In this paper, we consider the problem of constructing test sequencing algorithms for diagnosing multiple faults in redundant systems. Our approach is to: (1) generate all minimal faults and their failure signatures in the system, and (2) extend the multiple fault sequential testing strategies of our previous work [1] to fault-tolerant systems. In addition, the minimal fault analysis can be used for a quantitative evaluation of system dependability[3, 4].

*Research supported in part by the Department of Economic Development of the State of Connecticut, NASA-Ames Research Center, Sikorsky Aircraft and Qualtech Systems, Inc.

2 Problem Formulation

We assume that the system is modeled by a directed graph (digraph) $\mathbf{DG} = \{\mathbf{S}, \mathbf{T}, \mathbf{A}, \mathbf{E}\}$, where $\mathbf{E}$ denotes the set of directed edges specifying the functional information flow in the system, and

- $\mathbf{S} = \{s_1, ..., s_m\}$ is a finite set of independent failure sources associated with the system;
- $\mathbf{T} = \{t_1, t_2, ..., t_n\}$ is a finite set of n available binary outcome tests, where the integrity of system failure sources/components/modules can be ascertained;
- $\mathbf{A} = \{a_1, ..., a_K\}$ is a finite set of AND nodes representing system redundancies.

The input requirements of the various nodes of the digraph model are as follows:

1. Failure node: A priori probability vector of failure nodes $\mathbf{P} = [p(s_1), ..., p(s_m)]$, where $p(s_i)$ is the a priori probability of failure source s_i.
2. Test node: A set of test costs $\mathbf{C} = \{c_1, c_2, ..., c_n\}$, where c_j is the cost of applying test t_j.
3. AND node: Two sets $\mathbf{F} = \{f_1, ..., f_K\}$ and $\mathbf{G} = \{g_1, ..., g_K\}$, where f_k and g_k denote f_k-out-of-g_k logic for AND node a_k, i.e., AND node a_k has g_k inputs and a failure must occur in at least f_k inputs of this AND node for the faults to propagate to the output.

The problem is to design a testing strategy that unambiguously isolates the failure sources with minimum expected testing cost. The sequential test strategy is represented in the form of an AND/OR decision tree, where the OR nodes represent the suspect sets of failure sources, AND nodes are tests applied at various OR nodes, and the leaves are the isolated failure sources.

3 Minimal Fault Algorithm

In digraph models without AND nodes, i.e., having no redundancy, the test matrix (fault dictionary) denotes the full-order dependency among single failures and tests in

the system. Assuming that the failure signature of a multiple failure is the union of failure signatures of individual failures, the binary test matrix forms the knowledge base for diagnosing single faults, as well as multiple faults, in a system having no redundancy. However, this property is not valid for a digraph model with AND nodes. It can be shown that single faults and minimal faults, i.e., minimum number of failures with a failure signature different from the union of failure signatures of individual failures, together with their failure signatures, contain all the necessary information for fault diagnosis in digraph models with AND nodes [10].

In order to generate minimal faults and their failure signatures, we compute the following dependency matrices using the reachability analysis algorithm [2, 6]:

- Failure source-test dependency matrix $D = [d_{ij}]$ is a binary matrix of dimension $m \times n$, where $d_{ij} = 1$ if t_j monitors failure source s_i; otherwise, $d_{ij} = 0$;
- Failure source-AND node dependency matrix $B = [b_{ij(l)}]$ is a binary matrix of dimension $m \times z$, where $z = \sum_{j=1}^{K}(g_j + 1)$; $b_{ij(0)} = 1$ if a failure of s_i can reach the output of AND node a_j; otherwise, $b_{ij(0)} = 0$; and $b_{ij(l)} = 1$ if a failure of s_i can reach the lth input of AND node a_j; otherwise, $b_{ij(l)} = 0$;
- AND node-test dependency matrix $E = [e_{ij}]$ is a binary matrix of dimension $K \times n$, where $e_{ij} = 1$ if t_j monitors AND node a_i; otherwise, $e_{ij} = 0$;
- AND node-AND node dependency matrix $R = [r_{ij(l)}]$ is a binary matrix of dimension $K \times z$, where $r_{ij(0)} = 1$ if a failure at the output of AND node a_i can reach the output of AND node a_j; otherwise, $r_{ij(l)} = 0$; and $r_{ij(l)} = 1$ if a failure at the output of AND node a_i can reach the lth input of AND node a_j; otherwise, $r_{ij(l)} = 0$;
- AND node-AND node reachability matrix $Q = [q_{ij}]$ is a binary matrix of dimension $K \times K$, where $q_{ij} = 1$ if there is at least a path between a_i and a_j; otherwise, $q_{ij} = 0$.

Note that we can generate AND node-AND node reachability matrix Q by setting AND node's logic of f_k-out-of-g_k to 1-out-of-g_k in the reachability analysis algorithm, i.e., AND nodes devolve into OR nodes.

For convenience, given a binary matrix $X = [x_{ij}]$ of dimension $k_1 \times k_2$, we define Xr_i for $i = 1, ..., k_1$ as its ith row, and Xc_j for $j = 1, ..., k_2$ as its jth column. For example, Dr_i for $i = 1, ..., m$ lists all the tests that can detect failure source s_i, and Dc_j for $j = 1, ..., n$ indicates all failure sources detectable by test t_j.

Using these matrices, the minimal fault algorithm finds the minimal faults of the digraph model via the following steps:

1. Sort a subset of AND nodes $As \subseteq$ A to be processed.
2. Generate minimal combinations of AND nodes and failure nodes that propagate to the output of every AND node $a_k \in As$, i.e., $MC(a_k)$ for $a_k \in As$.
3. Generate minimal faults of each AND node in As, i.e., $MF(a_k)$ for $a_k \in As$.
4. Generate the minimal faults of the digraph model (MF_d) using the minimal faults of AND nodes.

3.1 Step 1 - Sorting the AND nodes

Since we need to process only those AND nodes $As \subseteq$ A for which there exists at least a path to a test, the algorithm sorts the AND nodes in As such that an AND node will be processed before any other AND nodes reachable from it. This step prevents the algorithm from performing similar operations repeatedly.

The procedure for finding and sorting As is as follows: (1) Find a subset of AND nodes $Ae \subseteq A$ such that an AND node $a_k \in Ae$ can be detected by at least a test, i.e., $Ae = \cup Ec_j$ for $j = 1, ..., n$, (2) Using AND node-AND node reachability matrix Q, find the subset of AND nodes As that reach Ae, i.e., $As = \cup_{a_k \in Ae} Qc_k$, and (3) Sort the AND nodes in As in the ascending order of the number of AND nodes reaching them. Note that the number of AND nodes reaching AND node a_j is $\sum_{k=1}^{K} q_{kj}$.

3.2 Step 2 - Generation of minimal combinations for each AND node

Using the binary matrices, the minimal fault algorithm generates minimal combinations of AND nodes and failure nodes for each AND node in As , i.e., $MC(a_k)$ for $a_k \in As$. The procedure is as follows: (1) For each AND node a_k, determine the failure sources and AND nodes that can reach the inputs and output of a_k, i.e., $Sc_k(l)$ for $l = 0, 1, , ..., g_k$, where $Sc_k(l) = Bc_k(l) \cup Rc_k(l)$, (2) Remove $Sc_k(0)$ from $Sc_k(l)$ for $l = 1, ..., g_k$, that is, remove single failures affecting the AND node output from its input signatures. (3) Because of the f_k-out-of-g_k logic, all combinations of $Sc_k(l)$ for $l = 1, ..., g_k$ containing sets of cardinality f_k are considered. For example, for an AND node, say a_k, with 2-out-of-3 logic, we consider the following combinations: ($Sc_k(1)$, $Sc_k(2)$), ($Sc_k(1)$, $Sc_k(3)$) and ($Sc_k(2)$, $Sc_k(3)$). Then, using $Sc_k(l)$'s combinations, generate the minimal combinations of AND nodes and

failure nodes for each AND node by one of the following three approaches: (a) Minimal hitting set[1] method using a breadth first search [7], (b) Minimal hitting set method using a depth first search, and (c) Binary decision diagrams [8]. When g_k=2 for $k = 1, ..., K$, the minimal combinations for each AND node can be found using the cross-product of two sets $Sc_k(1)$ and $Sc_k(2)$ for $k = 1, ..., K$ [9].

Note that we can consider a limit L for the number of failure sources in the minimal combinations of AND nodes. In the first and second approaches (i.e., (a) and (b)), those combinations with more than L failure sources are not expanded. In the third approach (i.e., (c)), at first the decision diagram is generated, and then, the combinations with more than L faults are eliminated. However, because of the small number of sets, i.e., g_k for $k = 1, ..., K$, usually 2 or 3, there is almost no difference in using any of these three approaches.

3.3 Step 3 - Minimal faults for each AND node

After generating minimal combinations for each AND node in *As*, we process one AND node at a time. The subroutine for this part is a recursive function, and for simplicity, we call it MFG (Minimal Fault Generator): MFG (*AND-node*, *fault-list*, *AND-list*, *level*, *solved-AND-nodes*). In order to find minimal faults for an AND node, say a_i, we call MFG as follows: MFG(a_i, $\emptyset$, $\emptyset$, 0, $\emptyset$). MFG adds a_i to *AND-list*, and considers it as a level-zero AND node. Then, if the minimal faults of this AND node have already been found, MFG adds one of the a_i's minimal faults to the *fault-list*. Otherwise, it adds one of the a_i's minimal combinations to the *fault-list*, removes *solved-AND-nodes* from that list, and sets the level of the new AND nodes in the *fault-list* to *level*+1. Then, it removes the AND node with the highest *level*, say a_j, from the *fault-list*, and calls MFG as follows: MFG(a_j, *fault-list*, *AND-list*, *level+1*, *solved-AND-nodes*). This procedure is continued until: (a) no AND node remains in the *fault-list*, or (b) the level of the selected AND node is less than or equal to the level of one or more AND nodes in the *AND-list*, or (c) the algorithm picks an AND node that has already been processed, i.e., the AND node is in the *AND-list*.

In the first case, this combination is compared with other combinations created for AND node a_i. If it is a super set of one of them, MFG does not do any thing, and returns. If it is a subset of one or more of them, the algorithm removes the super sets, and it stores this combination, as well as the set of AND nodes affected by this combination, i.e., *AND-list* without considering levels, and returns.

In the second case, the algorithm adds the AND nodes with levels greater than or equal to the level of the selected AND node in the *AND-list* to the *solved-AND-nodes*, resets their levels to zero, and removes these AND nodes from the *fault-list*, if any. Then, it processes the selected AND node.

In the third case, if the AND node has already been solved, i.e., it belongs to the *solved-AND-nodes*, it removes the AND node from the *fault-list*, if any, and picks another AND node and processes it. Otherwise, the algorithm returns without doing any thing. This latter step prevents the algorithm from entering an infinite loop, when a cycle is encountered.

Note that we can consider a limit L for the number of faults in the minimal faults of AND nodes. In this case, MFG checks whether the number of faults in the *fault-list* is greater than the limit L or not. If the number of faults in the *fault-list* is greater than L, it returns; otherwise, it expands the AND node as before.

3.4 Step 4 - Minimal faults of digraph models

After generating the minimal faults of the AND nodes, the algorithm generates the minimal faults of the digraph model (MF_d). Firstly, using the AND node-test dependency matrix E, the algorithm removes the minimal faults of those AND nodes that cannot be detected by any test. Secondly, if a set of faults belongs to more than one minimal combination of AND nodes, the algorithm considers only one of them, and stores the union of corresponding *AND-list* as the set of AND nodes affected by the set of faults. Then, using matrices D, E and *AND-list*, the algorithm generates the failure signatures of remaining faults, i.e., $\cup_{s_i \in w_j} Dr_i \cup_{a_k \in AND-list} Er_k$, where w_j is a minimal fault, and *AND-list* is the list of AND nodes affected by w_j. Note that the remaining faults may contain super sets, and because of the test points in the digraph model, a super set may/may not be a minimal fault of the digraph model. Those super sets which have the same failure signatures as the union of the failure signatures of their subsets are removed. For example, let us consider a digraph model with failure sources S=$\{s_1, s_2, s_3\}$, AND nodes A=$\{a_1, a_2\}$ and tests T=$\{t_1, t_2\}$. A failure of s_1 and s_2 can be propagated to the output of the first AND node, i.e., a_1, and can be detected by test t_1. A failure of s_1, s_2 and s_3 propagates to the output of second AND node, i.e., a_2, and is detected by test t_2. Therefore, $\{ s_1, s_2 \}$ and $\{ s_1, s_2, s_3 \}$ are minimal faults of AND nodes a_1 and a_2, respectively, and the minimal fault of a_2 is a super set of the minimal fault of a_1. However, because of the different failure signatures,

[1] A hitting set for a collection of sets C is a set $H \subseteq \cup_{X \in C} X$ such that $H \cap X \neq \emptyset$ for each $X \in C$

i.e., $\{s_1, s_2\}$ is detected by t_1 and $\{s_1, s_2, s_3\}$ is detected by t_1 and t_2, these two sets are the minimal faults of the digraph model.

In order to illustrate the minimal fault algorithm, we present the following examples.

Example 1: Consider the digraph model in Figure 1. It consists of four failure sources S = $\{s_1, s_2, s_3, s_4\}$, three AND nodes A = $\{a_1, a_2, a_3\}$ and one test point T = $\{t_1\}$. $\{s_1, s_2, s_3, s_4\}$ is the minimal fault for this digraph model.

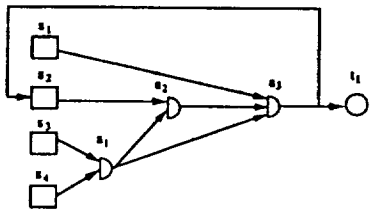

Figure 1: Digraph model of Example 1

The minimal fault algorithm works as follows:

- Step 1: The AND nodes are sorted as follows: a_1, a_2 and a_3.
- Step 2: The minimal combinations of failure sources and AND nodes for each AND node are as follows: MC(a_1) = $\{\{s_3, s_4\}\}$; MC(a_2) = $\{\{a_1, s_2\}$, $\{a_1, a_3\}\}$; MC(a_3) = $\{\{s_1, a_1, a_2\}\}$.
- Step 3: $\{s_3,s_4\}$, $\{s_2,s_3,s_4\}$ and $\{s_1,s_2,s_3,s_4\}$ are the minimal faults of AND nodes a_1, a_2 and a_3, respectively.
- Step 4: No test can detect $\{a_1, a_2\}$. Therefore, the minimal faults of these AND nodes should be eliminated. Thus, $\{s_1, s_2, s_3, s_4\}$ is the only minimal fault of the digraph model in Figure 1.

Example 2: Consider the digraph model of the F18 Flight Control System (FCS) for the left Leading Edge Flap (LEF) in Figure 2, which was used as an example in [3]. The minimal faults for this digraph model are {FCCA, FCCB}, {FCCA, CHNL3}, {FCCB, CHNL2}, and {CHNL2, CHNL3}.

Example 3: Consider the digraph model in Figure 3, which was used as an example in [9]. The minimal faults for this digraph model are $\{s_2, s_3\}$, $\{s_3, s_4\}$, $\{s_3, s_5\}$, and $\{s_9, s_{10}\}$.

3.5 Multiple Fault Strategy

It can be shown that the computational and storage complexity of designing an optimal multiple-fault diagnostic strategy are exponential in m [10]. In order to reduce the

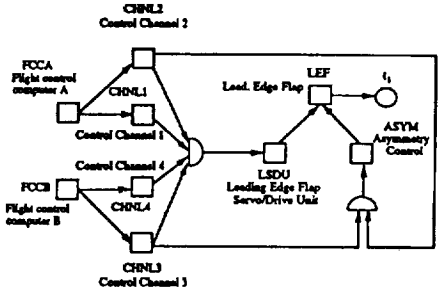

Figure 2: Digraph Model of F18 FCS LEF of Example 2

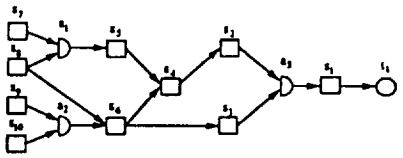

Figure 3: Digraph Model of Example 3

storage complexity, we use the compact multiple fault notation for the multiple fault ambiguity group at each OR node [1], [5]. Furthermore, in order to reduce computational complexity, we extend the Sure diagnostic strategies of our previous work [1] to redundant systems.

3.5.1 Compact Notation

We use the compact multiple fault notation $A = \Theta(L; F_1, ..., F_L; G)$ for the multiple fault ambiguity group at each OR node in systems without AND nodes [1], [5]. The F_i for $i = 1, ..., L$ and G are subsets of S =S $\cup \{s_0\}$ = $\{s_0, s_1, ..., s_m\}$, where s_0 is the fault-free condition; G is the set of known good failure sources (failure free sources), and F_i for $i = 1, ..., L$ are sets known to contain at least one definitely failed component each, i.e.,

$$\Theta(L; F_1, F_2, ..., F_L; G) = \{X \subseteq S | \\ X \cap F_i \neq \emptyset \text{ for } i = 1, ..., L, \text{ and } X \cap G = \emptyset$$

In the following, we summarize some of the properties of compact set notation:

- If $E \supseteq F_i$ for some i, then $\Theta(L+1; F_1, F_2, ..., F_L, E; G) = \Theta(L; F_1, F_2, ..., F_L; G)$.
- $\Theta(L; F_1, F_2, ..., F_L; G) = \Theta(L; F_1 \cap G^c, F_2 \cap G^c, ..., F_L \cap G^c; G)$, where superscript c denotes complement, i.e., $G^c = S - G$.
- the worst case storage complexity of compact set notation for an OR node is $O(mn)$.

- Multiple fault logic using the compact set notation is as follows: the initial hypothesis set is the set of all subsets of S, i.e., $A = \Theta(1;\ F_1 = S\ ;\ G = \emptyset)$. After performing a test, say t_j, the hypothesis set $A = \Theta(L;\ F_1, ..., F_L;\ G)$ is pruned as follows:

$$A \leftarrow \begin{cases} \Theta(L; (F_1 \cap TS_j^c), ..., (F_L \cap TS_j^c); (G \cup TS_j)) & \text{if } t_j \text{ passes} \\ \Theta(L+1; F_1, ..., F_L, TS_j \cap G^c; G) & \text{if } t_j \text{ fails} \end{cases}$$

- the failure sources belong to F_i with cardinality one are definitely faulty (*one-for-sure*).

In systems with AND nodes, each row of the test matrix corresponds to a subset of S= $\{s_1, ..., s_m\}$. For simplicity, let us assume that the new test matrix contains $m_n = m + m_f$ rows, where m_f is the number of minimal faults. We define W=$\{w_1, ..., w_{m_n}\}$, where $w_i = \{s_i\}$ for $i = 1, ..., m$, and $w_i \subseteq$ S for $m+1 \leq i \leq m_n$. Therefore, the F_i for $i = 1, ..., L$ and G are subsets of W=W$\cup \{w_0\}$ $= \{w_0, w_1, ..., w_{m_n}\}$, where $\{w_0\} = \{s_0\}$ and

$$\Theta(L; F_1, F_2, ..., F_L; G) = \{X \subseteq W | \\ X \cap F_i \neq \emptyset \text{ for } i = 1, ..., L, \text{ and } X \cap G = \emptyset\}$$

Based on these definitions, it can easily be shown that:

Lemma 1: Using *one-for-sure* condition, if the cardinality of any F_i is one, all the failure sources in $w_j \in F_i$ are faulty, and if the cardinality of F_i is greater than one, all the failure sources in $\cap_{w_j \in F_i} w_j$, are definitely faulty. It is obvious that these two conditions can be combined as follows: all the failure sources in $\cap_{w_j \in F_i} w_j$, for $i = 1, ..., L$, are definitely faulty.

Lemma 2: Let us assume that we repaired definitely failed components $X \subseteq S$, and there exist a $w_j \in G$ such that $|w_j - X| = 1$ and $s_k = |w_j - X|$. Therefore, s_k is good and should be added to the good component G.

Lemma 3: If we repair definitely failed components $w_i = \{s_i\}$, and there exists a w_j such that $s_i \in w_j$, then w_j should be added to the good subset G.

3.5.2 Sure Strategies

The basic idea of Sure strategies is to find one or more definitely failed components, while not making an error when other co-existing faults are present [1]. In these strategies, at each stage of diagnosis, we consider the minimal candidate set of the multiple fault suspect set corresponding to the OR node at that stage, and invoke the single fault strategy to isolate these candidates. Then, we propagate multiple fault suspect set through the resulting diagnostic tree. We repeat these procedures for each leaf node of the tree until: (1) the intersection of minimal candidates is not empty, i.e., the corresponding failure sources are definitely faulty, or (2) no test gives further information. The former corresponds to the case when the cardinality of one or more F_i in the ambiguity group is one. Note that in these strategies, we only repair definitely failed components.

Example 4: Consider the digraph model in Figure 4. The digraph model consists of failure source S = $\{s_1, s_2, s_3\}$, AND nodes A = $\{a_1, a_2, a_3\}$ and tests T = $\{t_1, t_2, t_3\}$. w_4=$\{s_1, s_2\}$, w_5=$\{s_2, s_3\}$ and w_6=$\{s_1, s_2, s_3\}$ are the minimal faults for this digraph model. The binary test matrix of the digraph model is shown in Figure 5. Figure 6 shows the multiple fault strategy for this system, where ACTION nodes represent the actions to be performed at that stage of diagnosis and A_i denotes the ambiguity group corresponding to the ith OR node, and $A_1 = \Theta(1; \{w_0, w_1, w_2, w_3, w_4, w_5, w_6\}; \emptyset)$; $A_2 = \Theta(1; \{w_0, w_1, w_2, w_3, w_5\}; \{w_4, w_6\})$; $A_3 = \Theta(1; \{w_4, w_6\}; \emptyset)$; $A_4 = \Theta(1; \{w_0, w_1, w_2, w_3\}; \{w_4, w_5, w_6\})$; $A_5 = \Theta(1; \{w_5\}; \{w_4, w_6\})$; $A_6 = \Theta(1; \{w_4\}; \{w_6\})$; and $A_7 = \Theta(1; \{w_6\}; \emptyset)$.

Note that we applied Lemma 2 to A_5 and A_6. For example, $A_5 = \Theta(1; \{w_5\}; \{w_4, w_6\})$. Thus, w_5 is definitely faulty, i.e., s_2 and s_3 are faulty. After repairing these failures, there is no need to apply additional tests. This is because w_6 belongs to G, and therefore, s_1 is good; G=S. One interesting point to note here is that we should not repair definitely failed components at intermediate nodes of the diagnostic strategy, because it may mask the failure of other faults. For example, $A_3 = \Theta(1; \{w_4, w_6\}; \emptyset)$. Using Lemma 1, $w_4 \cap w_6 = \{s_1, s_2\}$ are definitely faulty. If we repair s_1 and s_2 at this stage of diagnosis, a failure of s_3 will go undetected.

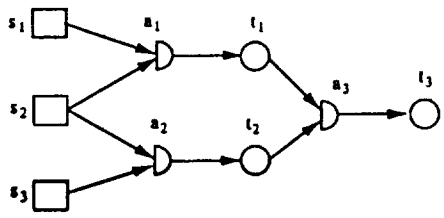

Figure 4: Digraph model with AND node

In addition to a set of comprehensive synthetic problems, we have applied the algorithms presented in this paper and those of [10] to several real-world systems. These include: (1) the Space Shuttle Main Propulsion System with 7271 failure sources and 1292 AND nodes [9], (2) the F18-Flight control system with 148 failure sources and 78 AND nodes [3] with failure sources limited to singletons and doubletons, (3) the anticollision light control system of the Sea Hawk helicopter with 51

Minimal faults	Tests		
	t_1	t_2	t_3
$w_0(s_0)$	0	0	0
$w_1(s_1)$	0	0	0
$w_2(s_2)$	0	0	0
$w_3(s_3)$	0	0	0
$w_4(s_1, s_2)$	1	0	0
$w_5(s_2, s_3)$	0	1	0
$w_6(s_1, s_2, s_3)$	1	1	1

Figure 5: Test matrix

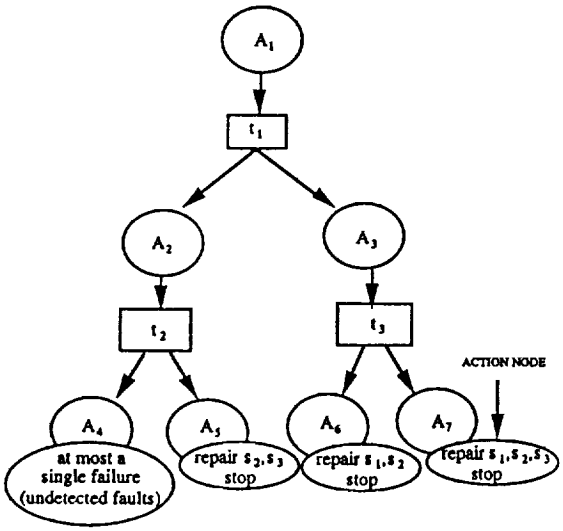

Figure 6: Diagnostic strategy

failure sources and 55 tests, (4) the stabilator system of the Black Hawk helicopter with 238 failure sources and 834 tests, and (5) the engine torque monitoring system used in CH-53E helicopter with 116 failure sources and 75 tests. In the latter three cases, static and dynamic multiple fault diagnostic strategies subject to various constraints on available resources, setup operations, and initial failure symptoms have been implemented, along with interfaces to interactive electronic technical manuals and multi-media documentation.

4 Conclusion

In this paper, we presented an algorithm to find all minimal faults in a digraph model and to generate their failure signatures. Further, we extended the multiple fault sequential testing strategies of our previous work [1] to redundant systems. Computational results indicate that these strategies can be used on systems with as many as 600 failure sources and 600 tests. Furthermore, using Sure strategies, a test strategy can be generated "on-line" to diagnose multiple faults in larger systems. That is, instead of generating the entire diagnostic tree, the interactive test generation program only suggests the next test to be applied given the outcomes of previously applied tests, and generates the path leading to the isolation of multiple failures in a system.

References

[1] M. Shakeri, K.R. Pattipati, V. Raghavan, , and S. Deb. Near-optimal sequential testing algorithms for multiple fault isolation. *IEEE International Conference on Systems, Man and Cybernetics*, pages 1908–1914, 1994.

[2] S. Deb, K.R. Pattipati, V. Raghavan, M. Shakeri, and R. Shrestha. Multi-signal flow graphs: A novel approach for system testability analysis and fault diagnosis. *IEEE Autotestcon*, 1994.

[3] S.A Doyle, J. B. Dugan, and A. Patterson-Hine. A quantitative analysis of the F18 flight control system. *American Institute of Aeronautics and Astronautics Computing in Aerospace 9 Conference Proceedings*, pages 668–675, October 1993.

[4] M. Veeraraghavan and K. S. Trivedi. An improved algorithm for symbolic reliability analysis. *IEEE Transactions on Reliability*, 40(3):347–358, August 1991.

[5] D.B. Grunberg, J.L. Weiss, and J.C. Deckert. Generation of optimal and suboptimal strategies for multiple fault isolation. *Technical report TM-248*, 1987.

[6] K.R. Pattipati, V. Raghavan, M. Shakeri, S. Deb, and R. Shrestha. TEAMS: Testability engineering and maintenance system. *American Control Conference*, pages 1989–1996, June 1994.

[7] R. Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32(1):57–95, Apr. 1987.

[8] A. Rauzy. New algorithm for fault trees analysis. *Reliability Engineering and System Safety*, 40:203–211, 1993.

[9] D.L. Iverson and F. A. Patterson-Hine. Digraph reliability model processing advances and applications. *Proceedings of the AIAA Computing in Aerospace Conference*, 1993.

[10] M. Shakeri, "Advances in Fault Diagnosis, Testability and Reliability Analysis of Large-Scale Systems", Ph.D Dissertation, August, 1995.

Test Sequencing Algorithms with Unreliable Tests

Vijaya Raghavan[†], Mojdeh Shakeri[†], and Krishna Pattipati[††]

Qualtech Systems Inc.[†]
Box-407, Mansfield Center, CT 06250
e-mail: vijay@sol.uconn.edu

Department of Electrical and Systems Engineering[††]
University of Connecticut, Storrs, CT 06269-3157
e-mail: krishna@sol.uconn.edu

Abstract

In this paper, we consider imperfect test sequencing problems under single fault assumption. This is a partially observed Markov decision problem (POMDP), a sequential multi-stage decision problem wherein the states are the set of possible failure sources and information regarding the states is obtained via the results of imperfect tests. The optimal solution for this problem can be obtained by applying a continuous state Dynamic Programming (DP) recursion. However, the DP recursion is computationally very expensive owing to the continuous nature of the state vector comprising the probabilities of faults. In order to alleviate this computational explosion, we present an efficient implementation of the DP recursion. We also consider various problems with special structure (parallel systems) and derive closed form solutions/index-rules without having to resort to DP. Finally, we consider various top-down graph search algorithms for problems with no special structure, including multi-step DP, multi-step information heuristics and certainty equivalence algorithms. We compare these near-optimal algorithms with DP for small problems to gauge their effectiveness.

1 Introduction

An important issue in the field maintenance of systems is the imperfect nature of tests due to improper setup, operator error, electromagnetic interference, environmental conditions, or aliasing inherent in the signature analysis of built-in-self-tests. Typically, a user complaint, which is a subjective measure of system performance, can also be considered as an imperfect test because it does provide some insight into the malfunction. Imperfect testing introduces an additional element of uncertainty into the diagnostic process: the pass outcome of a test does not guarantee the integrity of components under test (because the test may have missed a fault), or a failed test outcome does not mean that one or more of the implicated components are faulty (because the test outcome may have been a false alarm).

The consequences of a test error depend on the disposition of the system after repair. If a test results in a false alarm, a functioning component is replaced, and a failed component may be left in place. If the system is then returned to service, the system fails immediately. In the case of missed detection by a test, the overall test could indicate that no item has failed. In this case, the system might be returned to service where it fails immediately or it might be scrapped. Either choice implies a cost. Relatively little attention has been given to imperfect testing. Most research efforts were directed at finding test strategies for systems with special structure (parallel systems). The most complete treatment for parallel systems with imperfect tests is by Firstman and Gluss [1] in which a two level testing is studied with both false alarms and missed detections in tests. However, it is assumed that test errors are ultimately recovered by repeating the tests until a proper repair is made. The test sequence is then determined in the same manner as for perfect testing. The perfect-test rechecks assures test termination with proper repair and thus fails to capture the fact that test errors are often unrecoverable. For many systems, imperfect test results cannot be recognized either because of the test design or because retesting is economically infeasible. In these cases, the consequences of test

errors occur outside of the repair facility. Nachlas and Loney [2] presented the problem of test sequencing for fault diagnosis using unreliable tests for parallel systems. The objective of the fault diagnosis problem is to minimize the expected cost required to diagnose and repair the failed component. They present heuristic algorithms based on efficient enumeration of permutations of test sequences, which are not suitable for large problems with arbitrary structures. These problems belong to a class of hypothesis testing problems with dynamic information seeking. Problems in dynamic search arise in a wide variety of applications [10] [11] [12] [13] [14]. Dynamic search in the context of sequential detection was extensively treated by Wald [18] In [15],[16],[17], different search problems in the presence of false alarms were considered.

In this paper, we consider a generalized formulation of the test sequencing problem in the presence of imperfect tests for systems of arbitrary structure. The test sequencing problem in this case is a partially observed Markov decision problem (POMDP) [8] [9], a sequential multi-stage decision problem wherein the states are the set of possible failure sources and information regarding the states is obtained via the results of imperfect tests. The optimal solution for this problem can be obtained by applying a continuous state Dynamic Programming (DP) recursion. However, the DP recursion is computationally very expensive owing to the continuous nature of the state vector comprising the probabilities of faults. In order to alleviate this computational explosion, we present an efficient implementation of the DP recursion. We also consider various problems with special structure (parallel systems) and derive closed form solutions/index-rules without having to resort to DP. Finally, we consider various top-down graph search algorithms for problems with no special structure, including multi-step DP, multi-step information heuristics and certainty equivalence algorithms. We compare these near-optimal algorithms with DP for small problems to gauge their effectiveness.

2 Optimal Test Sequencing with Imperfect Tests

In its simplest form, the test sequencing problem with imperfect tests is as follows:

1. A system with a finite set of failure sources $S = \{s_0, s_1, s_2, \ldots, s_m\}$ is given. We make the standard assumption that the system is tested frequently enough that only one or none of the faults has occurred. The "no-fault" condition is denoted by a dummy failure source s_0;

2. The a priori probability of each failure source, $p(s_i)$ is known;

3. A finite set of n available tests $T = \{t_1, t_2, \ldots, t_n\}$ are given, where each test t_j checks a subset of failure sources. The relationship between the set of failure sources and the set of tests is represented by a reachability matrix $R = [r_{ij}]$, where $r_{ij} = 1$ if test t_j monitors failure source s_i;

4. The reliability of each test t_j is characterized by the detection-false-alarm probability pair (P_{dj}, P_{fj})[1], where P_{dj} = Prob{test t_j fails | any of the failure sources monitored by t_j has failed}, and P_{fj} = Prob{test t_j fails | none of the failure sources monitored by t_j has failed};

5. Each test $t_j (1 \leq j \leq n)$ costs an amount c_j measured in terms of time, or other economic factors;

6. Each failure source $s_i (1 \leq i \leq m)$, once identified has repair/replacement cost f_i, false repair/replacement cost C_{Ri}, and missed repair/replacement cost C_{Mi} associated with it.

The problem is to design a test algorithm with minimum expected diagnostic cost to isolate the failure source, if any, with a specified level of confidence α (typically, $\alpha \in [0.95, 0.99]$). Employing the single fault assumption, the reachability matrix R, and the test reliabilities (P_{dj}, P_{fj}) can be combined into a single matrix of "likelihoods", $D = [d_{ij}]$, where d_{ij} is given by

$$d_{ij} = r_{ij} P_{dj} + (1 - r_{ij}) P_{fj}, \tag{1}$$

where d_{ij} = Prob{ test t_j fails | failure source s_i has occurred }.

When tests are perfect, that is, $P_{dj} = 1 - P_{fj} = 1$ for all tests, we have $d_{ij} = r_{ij}$. This corresponds to a perfectly observed Markov decision problem, and has been discussed extensively in [7]. The solution to

[1] Extension to the case when (P_{dj}, P_{fj}) are functions of failure source s_i is straightforward.

this problem is a diagnostic decision tree, wherein the root corresponds to the state of complete ignorance, the intermediate nodes relate to states of residual ambiguity and the leaves correspond to individual failure sources. The test algorithm terminates when the failed element is isolated with complete certainty (that is, $\alpha = 1$).

When the tests are imperfect, the test sequencing problem is a partially observed Markov decision problem (POMDP), a sequential multi-stage decision problem wherein the states are the set of possible failure sources and information regarding the states is obtained via the results of imperfect tests. It can be shown [4] that the probabilities of failure sources conditioned on all the previous test results constitute a sufficient statistic (i.e., contain all the necessary information) for deciding the next test to be applied. Formally, let $j(k) \in \{1, 2, \ldots, n\}$ be the test applied at stage k and let $O(k) \in \{1(= pass), 0(= fail)\}$ be the outcome of test $t_{j(k)}$. Further, let I_{k-1} be the information available to decide on test $t_{j(k)}$ to be applied at stage k. This information includes all the past tests applied and their outcomes given by:

$$I_{k-1} = \{(t_{j(l)}, O(l)\}_{l=0}^{k-1}. \tag{2}$$

Using Bayes' rule, the conditional probabilities of hypotheses $\{\pi_i(k) = p(s_i|I_k)\ i = 0, 1, \ldots, m\}$, which are the information states of the decision process at each stage k, can be shown to evolve as

$$\pi_i(k+1) = \frac{[O(k) + (-1)^{O(k)} d_{ij(k)}]\pi_i(k)}{\sum_{l=0}^{m}[O(k) + (-1)^{O(k)} d_{lj(k)}]\pi_l(k)]}. \tag{3}$$

The above recursion is initiated with $\pi_i(0) = p(s_i), i = 0, 1 \ldots, m$, the *a priori* probability distribution of failure sources. The optimal test $t_{j(k)}$ is given by the dynamic programming (DP) recursion [4]:

$$h^*(\{\pi_i(k)\}) = \min_{j(k)\in\{1,2,\ldots,n\}} \left[c_{j(k)} + \left(\sum_{l=0}^{m} d_{lj(k)}\pi_l(k) \right) h^* \left(\frac{d_{ij(k)}\pi_i(k)}{\sum_{l=0}^{m} d_{lj(k)}\pi_l(k)} \right) + \left(\sum_{l=0}^{m} (1 - d_{lj(k)})\pi_l(k) \right) h^* \left(\frac{(1 - d_{ij(k)})\pi_i(k)}{\sum_{l=0}^{m}(1 - d_{lj(k)})\pi_l(k)} \right) \right] \tag{4}$$

where $c_{j(k)}$ is the cost of test $t_{j(k)}$, $h^*(\{\pi_i(k)\})$ is the optimal expected cost-to-go from the information state $\{\pi_i(k) : i = 1, 2, \ldots, m\}$, the terms involving h^* inside the brackets are the optimal costs-to-go from the information states corresponding to the fail and pass outcomes, respectively. The terminal states of this recursion have known cost :

$$h^*(\{\pi_i\}) = f_{i'} + (1 - \pi_{i'})C_{Ri'} + \sum_{i=1, i\neq i'}^{m} C_{Mi}\pi_i \tag{5}$$

where

$$i' = \arg\max_i \pi_i \tag{6}$$

This definition of terminal cost function corresponds to the policy of repairing the most likely fault. Since $\{\pi_i\}$ are continuous, the above DP recursion is continuous. Thus, the consideration of imperfect tests in the test sequencing problem formulation converts a finite (albeit large) dimensional search problem of the perfect test case into an infinite dimensional stochastic control problem.

3 Systems of Parallel Structure

Parallel systems are characterized by a reachability matrix R with ones on the diagonal and zeros everywhere else, for some permutation of tests. That is, every failure state is detected by one, and only one test. For parallel systems, we can explicitly characterize the optimal policy in the perfect test case: ***at each state of ambiguity, test a module with the highest ratio of probability of failure and the cost of testing the module.*** For the imperfect testing case, such a closed form solution cannot be obtained without making additional assumptions. However, in the following subsections, we derive closed-form solutions for some special cases.

3.1 Special Case 1:

Let us specialize the above to a parallel system with the following assumptions:

- all test costs are equal
- a test can be applied more than once
- a fault is implicated if its posterior probability at any stage of testing exceeds a given threshold γ (typically, $\gamma \in [0.95, 0.99]$)

Given this problem context, we first consider a greedy one-step lookahead strategy that maximizes the posterior probability of correct decision assuming that a decision would be taken at the next stage implicating the failure with the maximum posterior probability (MAP decision rule).

Let α and β denote the false alarm and missed detection probabilities of tests. Let $\pi_i(k)$ denote the conditional probability of the failure source s_i at $k-$th stage of testing (stage 0 is when no tests have been applied). Let $f_{ij}(O_j|s_i)$ denote the conditional probability density of the outcome $O_j \in \{0, 1\}$ of test t_j given that s_i is present. Given the nature of the tests, we further know that,

$$f_{ij}(O_j|s_i) = \begin{cases} g(O_i) = \beta\delta(O_i) + (1-\beta)\delta(O_i - 1) & \text{for } i = j \\ h(O_i) = (1-\alpha)\delta(O_i) + \alpha\delta(O_i - 1) & \text{for } i \neq j \end{cases} \tag{7}$$

where $\delta(.)$ is the Dirac Delta function.[2]

Let us assume that test t_{j_k} is the next test to be applied at stage k. Then, since the greedy approach corresponds to the assumption that the next test is the final one, the decision rule at the next stage is to implicate the failure source s_d such that:

$$\pi_d(k+1) = \max_i \pi_i(k+1) \tag{8}$$

which translates to,

$$\pi_d(k) f_{dj_k}(O(j_k, k)|s_d) = \max_i \{\pi_i(k) f_{ij_k}(O(j_k, k)|s_i)\} \tag{9}$$

Let us define:

$$P(C|s_i, j_k) = \text{Prob(Correct Decision } |s_i, j_k) \tag{10}$$

$$P(C|s_i, j_k) = \Pr\left\{\pi_i(k) g(O(j_k, k)) \geq \max_{l \neq i} \{\pi_l(k) h(O(j_k, k))\}\right\} \tag{11}$$

We can simplify the above equation as,

$$P(C|s_i, j_k) = \Pr\{\pi_i(k) h(O(j_k, k)) \geq \max\left[\max_{l \neq i, l \neq j_k} \{\pi_l(k) h(O(j_k, k))\}, \pi_{j_k}(k) g(O(j_k, k))\right]\} \text{ for } j_k \neq i \tag{12}$$

In order to simplify the above two equations, we define:

$$\overline{m} = \arg\max_i \pi_i(k) \tag{13}$$

$$\hat{m} = \arg\max_{i \neq \overline{m}} \pi_i(k) \tag{14}$$

[2]The function $\delta(x)$ is defined via:

$$\delta(x) = 0 \quad \forall\ x \neq 0$$
$$\int_{-\infty}^{\infty} \delta(x)dx = 1$$

Then we have for $i = \overline{m}$:

$$P(C|s_{\overline{m}}, j_k) = Q_g(\frac{\pi_{\hat{m}}(k)}{\pi_{\overline{m}}(k)}) \text{ for } j_k = \overline{m}$$
$$= 1 - Q_h(\frac{\pi_{\overline{m}}(k)}{\pi_{\hat{m}}(k)}) \text{ for } j_k \neq \overline{m} \quad (15)$$

and for $i \neq \overline{m}$,

$$P(C|s_i, j_k) = Q_g(\frac{\pi_{\overline{m}}(k)}{\pi_{\hat{m}}(k)}) \text{ for } j_k = i$$
$$= 0 \text{ for } j_k \neq i \quad (16)$$

where

$$Q_g(a) = \Pr(g(x) > ah(x) \mid \text{PDF of } x \text{ is } g(x)) \quad (17)$$
$$Q_h(a) = \Pr(g(x) > ah(x) \mid \text{PDF of } x \text{ is } h(x)) \quad (18)$$

Therefore, we have:

$$P(C|j_k) = \pi_{\overline{m}}(k)Q_g(\frac{\pi_{\hat{m}}(k)}{\pi_{\overline{m}}(k)}) + \pi_{\hat{m}}(k)(1 - Q_h(\frac{\pi_{\hat{m}}(k)}{\pi_{\overline{m}}(k)})) \text{ for } j_k = \overline{m}$$
$$= \pi_{\hat{m}}(k)Q_g(\frac{\pi_{\hat{m}}(k)}{\pi_{\overline{m}}(k)}) + \pi_{\overline{m}}(k)(1 - Q_h(\frac{\pi_{\hat{m}}(k)}{\pi_{\overline{m}}(k)})) \text{ for } j_k = \hat{m}$$
$$= \pi_{j_k}(k)Q_g(\frac{\pi_{\overline{m}}(k)}{\pi_{j_k}(k)}) + \pi_{\overline{m}}(k)(1 - Q_h(\frac{\pi_{\overline{m}}(k)}{\pi_{j_k}(k)})) \text{ for all other } j_k$$

Now an index j_k is to be chosen so that the above expression is maximized. In the following, we will show that $P(C|j_k)$ for j_k other than $\overline{m}$ and $\hat{m}$ is less than $P(C|\hat{m})$. Let us transform $Q_g(.)$ and $Q_h(.)$ by forming the likelihood ratio:

$$\lambda = \frac{g(x)}{h(x)} \quad (19)$$

Assume that if x has a PDF $h(x)$, then λ has a probability density function $f_h(\lambda)$, distribution function $F_h(\lambda)$, and integral of the distribution function $\Psi_h(\lambda)$. That is,

$$\frac{d}{d\lambda}\Psi_h(\lambda) = F_h(\lambda), \frac{d}{d\lambda}F_h(\lambda) = f_h(\lambda) \quad (20)$$

Similarly, assume that if x has a PDF of $g(x)$, then λ has a probability density $f_g(\lambda)$, distribution function $F_g(\lambda)$, and integral of the distribution function $\Psi_g(\lambda)$. It follows that:

$$Q_h(a) = 1 - F_h(a), \text{ and } Q_g(a) = 1 - F_g(a) \quad (21)$$

It can be easily shown that:

$$\frac{f_g(\lambda)}{f_h(\lambda)} = \lambda \quad (22)$$

and that

$$Q_g(a) = 1 - \int_{-\infty}^{a} \lambda f_h(\lambda)d\lambda = 1 - aF_h(a) + \Psi_h(a) \quad (23)$$

Now, let us define

$$\varphi(y) = 1 - Q_h(\frac{1}{y}) + yQ_g(\frac{1}{y}) \quad (24)$$

and note that,

$$P(C|j_k) = \pi_{\overline{m}}(k)\varphi(\frac{\pi_{j_k}(k)}{\pi_{\overline{m}}(k)}) \quad \text{for } j_k \neq \overline{m} \text{ and } j_k \neq \hat{m} \quad (25)$$

We have,

$$\varphi(y) = y + y\Psi_h(\frac{1}{h}) \tag{26}$$

and $\varphi(0) = 1$. Now,

$$\frac{d}{dy}\varphi(y) = 1 - \frac{1}{y}F_h(\frac{1}{y}) + \Psi_h(\frac{1}{y}) = Q_g(\frac{1}{y}) \geq 0 \tag{27}$$

Hence, we know that $\varphi(y)$ is a non-decreasing function, and that its minimum value is at $y = 0$ which is unity.

Therefore,

$$P(C|j_k) = \pi_{\overline{m}}(k)\varphi(\frac{\pi_{j_k}(k)}{\pi_{\overline{m}}(k)}) \text{ for } j_k \neq \overline{m} \text{ and } j_k \neq \hat{m} \tag{28}$$

Since $\varphi(y) \geq Q_g(x)$ for $0 \leq x, y \leq 1$, the conditional probability $P(C|j_k)$ is maximized by choosing $j_k = \hat{m}$. However, $P(C|\overline{m})$ can be greater than $P(C|\hat{m})$, and we have to check this condition. Now suppose we define the Binary Bayesian Equal-cost hypothesis test, where the hypotheses are that the measurement x has come either from a probability distribution $f(.)$ or from $g(.)$ with π_h and π_g being the associated priors. We can write the corresponding minimal probability of error as,

$$r^*(\pi_h) = \pi_h Q_h(\frac{\pi_h}{(1-\pi_h)}) + (1-\pi_h)(1 - Q_g(\frac{\pi_h}{(1-\pi_h)})) \tag{29}$$

Note that

$$P(C|\overline{m}) = \left[1 - r^*\left\{\frac{\pi_{\hat{m}}(k)}{\pi_{\overline{m}}(k) + \pi_{\hat{m}}(k)}\right\}\right](\pi_{\overline{m}}(k) + \pi_{\hat{m}}(k)) \tag{30}$$

and

$$P(C|\hat{m}) = \left[1 - r^*\left\{1 - \frac{\pi_{\hat{m}}(k)}{\pi_{\overline{m}}(k) + \pi_{\hat{m}}(k)}\right\}\right](\pi_{\overline{m}}(k) + \pi_{\hat{m}}(k)) \tag{31}$$

Hence, the optimal strategy is to set $j_k = \hat{m}$ if

$$r^*\left\{\frac{\pi_{\hat{m}}(k)}{\pi_{\overline{m}}(k) + \pi_{\hat{m}}(k)}\right\} > r^*\left\{1 - \frac{\pi_{\hat{m}}(k)}{\pi_{\overline{m}}(k) + \pi_{\hat{m}}(k)}\right\}$$

Otherwise, set $j_k = \overline{m}$.

The above decision rule can be further simplified by substituting the exact expressions for $Q_g(.)$ and $Q_h(.)$. That is, the optimal strategy is to set $j_k = \hat{m}$ if

$$\pi_{\overline{m}}(k)\left[Q_g(\frac{\pi_{\hat{m}}(k)}{\pi_{\overline{m}}(k)}) + Q_h(\frac{\pi_{\hat{m}}(k)}{\pi_{\overline{m}}(k)}) - 1\right] + \pi_{\hat{m}}(k)\left[1 - Q_g(\frac{\pi_{\hat{m}}(k)}{\pi_{\overline{m}}(k)}) - Q_h(\frac{\pi_{\hat{m}}(k)}{\pi_{\overline{m}}(k)})\right] > 0 \tag{32}$$

By combining the terms, and expanding $Q_g(.)$ and $Q_h(.)$, the above comparison can be written as,

$$1 - \{(1-\alpha+\beta)I(\frac{1-\beta}{\alpha} > \frac{\pi_{\hat{m}}(k)}{\pi_{\overline{m}}(k)}) + (1+\alpha-\beta)I(\frac{\beta}{1-\alpha} > \frac{\pi_{\hat{m}}(k)}{\pi_{\overline{m}}(k)})\} > 0 \tag{33}$$

where the indicator function $I(E) = 1$, when the logical expression E is true, and zero otherwise. This implies that, when both missed detections and false alarms are present, the optimal policy is to test the fault with highest posterior probability if the above expression is not greater than zero. Otherwise, the fault with the second highest posterior probability should be tested. Note that, when the probability of missed detection β is zero, then the above expression is always greater than zero, implying that the decision rule is to choose $j(k) = \hat{m}$, i.e. test the fault with second highest posterior probability.

3.2 Special Case 2:

Let us consider another special case involving a parallel system with the following assumptions:

- the test costs $\{c_1, c_2, \ldots, c_m\}$ are known

- a test cannot be applied more than once
- a fault is implicated when a test detecting it fails
- the tests have no false alarms
- the missed detection probabilities $\{\beta_i\}$ are known

Given this scenario, we now proceed to show that the optimal test sequence is an index rule.

Let $S^* = \{j(1), j(2), \ldots, j(m)\}$ represent the index set of the optimal test sequence that minimizes the expected testing cost. Before writing down the expression for the expected testing cost, let us consider the testing strategy described above in detail. The first test to be applied is $t_{j(1)}$, and this test is always applied. The second test $t_{j(2)}$ is applied under one of the following two situations:

1. the component $s_{j(1)}$ is not faulty (hence $t_{j(1)}$ would not fail), or
2. the component $s_{j(1)}$ is faulty but the test $t_{j(1)}$ missed detecting it (the probability of this event is $\beta_{j(1)}$)

Similarly, the third test $t_{j(3)}$ in the sequence is applied if $s_{j(1)}$ and $s_{j(2)}$ are not faulty or if the tests $t_{j(1)}$ and $t_{j(2)}$ missed detecting them.

Now, the above discussion lets us write the expression for the expected testing cost as,

$$\begin{aligned} E[J(S^*)] &= c_{j(1)} + c_{j(2)}(1 - p(s_{j(1)}) + p(s_{j(1)})\beta_{j(1)}) + \ldots \\ &= \sum_{k=1}^{m} \left\{ c_{j(k)}(1 - \sum_{i=1}^{k-1} p(s_{j(i)})(1 - \beta_{j(i)})) \right\} \end{aligned} \tag{34}$$

Let $S' = \{j'(1), j'(2), \ldots, j'(m)\}$ be another sequence of tests obtained from S^* by interchanging terms k and $k+1$. That is,

$$\begin{aligned} j'(i) &= j(i) \qquad \text{for } i \neq k \text{ and } i \neq k+1 \\ j'(k) &= j(k+1) \\ j'(k+1) &= j(k) \end{aligned}$$

If S^* is the optimal sequence, then for any k, the expected testing cost of S' should be greater than or equal to that of S^*. Hence, by expanding and simplifying the logical expression $E[J(S')] \geq E[J(S^*)]$, we get

$$\begin{aligned} c_{j(k+1)}(1 - \sum_{i=1}^{i=k-1} p(s_{j(i)})(1 - \beta_{j(i)}) + \\ c_{j(k)}(1 - p(s_{j(k+1)})(1 - \beta_{j(k+1)}) - \sum_{i=1}^{i=k-1} p(s_{j(i)})(1 - \beta_{j(i)}) \geq \\ c_{j(k)}(1 - \sum_{i=1}^{i=k-1} p(s_{j(i)})(1 - \beta_{j(i)}) + \\ c_{j(k+1)}(1 - \sum_{i=1}^{i=k} p(s_{j(i)})(1 - \beta_{j(i)}) \end{aligned}$$

Simplifying,

$$p(s_{j(k)})(1 - \beta_{j(k)})/c_{j(k)} \geq p(s_{j(k+1)})(1 - \beta_{j(k+1)})/c_{j(k+1)} \tag{35}$$

That is, the optimal sequence satisfies the above ordering relation. To prove the converse, observe that the inequality

$$p(s_{j'(k)})(1 - \beta_{j'(k)})/c_{j'(k)} \geq p(s_{j(k)})(1 - \beta_{j(k)})/c_{j(k)} \tag{36}$$

implies that $E[J(S^*)] \leq E[J(S')]$ and that therefore any sequence that is different from S^* can be transformed to S^* by successive exchanges of neighboring indices and the result is a reduction in cost. Therefore, the ordering relation (35) defines an optimal sequence of tests.

4 Near-optimal Test Sequencing using Information Heuristics and Certainty Equivalence

An alternative to DP-based test sequencing algorithms is the class of approximation techniques that employ *greedy* heuristics based on information theory. For example, in a one-step lookahead information heuristic algorithm, if $\{\pi_i(k)\}$ is the current information state at stage k, we select a test $t_{j(k)}$ if it maximizes the information gain per unit cost of the test. The selection rule is:

$$j(k) = \arg \max_{j \in \{1,2,\ldots,n\}} \left\{ \frac{\mathrm{IG}(\{\pi_i(k)\}, t_{j(k)})}{c_j} \right\} \tag{37}$$

where $\mathrm{IG}(\{\pi_i(k)\}, t_{j(k)})$ is the information gain given by:

$$\mathrm{IG}(\{\pi_i(k)\}, t_{j(k)}) = H(\{\pi_i(k)\}) - H(\{\pi_i(k+1)\} | t_j \text{is applied}) \tag{38}$$

We can write the expression for the information gain explicitly as:

$$\begin{aligned} \mathrm{IG}(\{\pi_i(k)\}, t_{j(k)}) = & \sum_{i=1}^{m} \pi_i(k) \left(d_{ij} \log d_{ij} + (1 - d_{ij}) \log (1 - d_{ij})\right) \\ & - \left(\sum_{i=1}^{m} (1 - d_{ij}) \pi_i(k)\right) \log \left(\sum_{i=1}^{m} (1 - d_{ij}) \pi_i(k)\right) \\ & - \left(\sum_{i=1}^{m} d_{ij} \pi_i(k)\right) \log \left(\left(\sum_{i=1}^{m} d_{ij} \pi_i(k)\right)\right) \end{aligned} \tag{39}$$

Another alternative to DP-based algorithms is the Certainty Equivalence technique. In this approach, we compute the best test to be applied at every stage, assuming that the tests are reliable, using AO*algorithm [7] that uses the current posterior probabilities of failures for prior probabilities. Once the test result is known, the posterior probabilities of the failures are updated using (3) and the best test is computed again as above.

Both of the above approaches do not mandate that a test cannot be repeated. Hence, a suitable stopping criterion is necessary in order to terminate the testing process. One stopping criterion is to compute the expected cost incurred on applying the chosen test and stop testing at the current stage if the computed cost is greater than the expected cost at the current stage. Another stopping rule would involve pruning the ambiguity group at every stage based on the posterior probabilities and stop when the ambiguity group of faults contains a single fault. A reasonable pruning rule can be devised by the following consideration: if the tests are only very slightly imperfect, then after the application of a fairly large number of tests, the posterior probabilities of non-existent failure states are reduced to a tiny fraction of their prior probabilities before testing. Hence, a failure source s_i could be removed from the ambiguity group at stage k, if

$$\pi_i(k) \leq \pi_i(0)/N_p \tag{40}$$

where N_p is a factor suitably chosen (e.g., $N_p > 100$).

5 Implementation of Dynamic Programming Solution

The sequential testing problem formulated earlier via DP cannot be solved in its original form, since the state space (consisting of the posterior probability vector) is continuous. Hence, some form of discretization is necessary for the computer implementation of the DP method for this problem. Even with this discretization, we will see that problems having more than 20 failure sources cannot be solved optimally owing to the non-polynomial time complexity of DP. However, DP can serve as a benchmark against which the performance of near-optimal algorithms can be compared, at least for problems of small size. In the following, we present an efficient technique to implement the DP recursion that makes use of "lean" data structures. These data structures circumvent the explosive storage requirements of DP, while guaranteeing fast access to states.

5.1 Outline of the technique

Before we get into the details of state space quantization, let us consider a rough overview of the solution procedure. Suppose that quantization is already performed and we have a set $X = \{x_i : (1 \leq i \leq n_s)\}$ of states at hand. Note that every element x_i represents a vector posterior probabilities of failure sources. For example, a two failure source problem with $n_s = 3$ uniform quantization levels results in $x_1 = (1.0, 0.0)$, $x_2 = (0.5, 0.5)$, and $x_3 = (0.0, 1.0)$. Let us also define an appropriate terminal cost function $f(.)$, such that $f(x_i)$ is the cost incurred if no further testing is carried out at state x_i. Note that $f(.)$ depends on the maintenance/repair philosophy followed. Let $J_k(x_i)$ represent the optimal cost-to-go for state x_i at stage k ($k = 1, 2, \ldots$) of testing. For an $N-$stage DP problem (i.e., no more than N tests would be used before diagnosis/repair), by definition,

$$J_N(x_i) = f(x_i), \qquad \forall \quad 1 \leq i \leq n_s \tag{41}$$

Now suppose there are n tests in the system, and it is desired to determine the optimal test to be performed for every state-stage $\{(x_i, k) : 1 \leq i \leq n_s, 1 \leq k \leq N-1\}$. Let us define the state-mapping functions for the n tests, $T_{jp}(x_i), T_{jf}(x_i), 1 \leq j \leq n, 1 \leq i \leq n_s$, The definition of the state-mapping functions is as follows: when test j is applied at state x_i, the pass outcome takes the posterior probability state to $T_{jp}(x_i) \in X$, and the fail outcome transforms it to $T_{jf}(x_i) \in X$. Let $P_{jp}(x_i)$ and $P_{jf}(x_i)$ be the associated probabilities of these events conditioned on state x_i.

The recursive DP formulation of (4) can be adapted to the above quantized version as follows:

$$J_k(x_i) = \min_j \{c_j + P_{jp}(x_i) J_{k+1}(T_{jp}(x_i)) + P_{jf}(x_i) J_{k+1}(T_{jf}(x_i))\}$$
$$1 \leq i \leq n_s, 1 \leq k \leq N-1 \tag{42}$$

The index j that maximizes the above recursion is the best test to apply at stage k. Thus, we initialize this recursion at $k = N$ with,

$$J_N(x_i) = f(x_i) \quad 1 \leq i \leq n_s \tag{43}$$

and carry through backwards from stage $k = N-1$ to $k = 1$.

However, the computer implementation of this recursion requires consideration of the following important issues that directly affect the size of the problems that can be solved:

1. **Quantization Scheme:** We need to determine the optimal quantization scheme to map floating point probabilities (that can lie anywhere in $[0,1]$) to discrete levels. We will see that any simplistic rule to quantize the probabilities may result in quantization levels that do not map valid probability states.

2. **State Space Data Structures:** If the storage is not an issue, then the above recursions can be solved very easily by precomputing the mapping functions $T_{jp}(x_i), T_{jf}(x_i), 1 \leq j \leq n, 1 \leq i \leq n_s$. However, we will see that the storage requirements are prohibitively high for even small problems with not too many quantization levels. Hence, we need to determine efficient ways of storing the discrete probability states and computing the test-mapping functions on the fly.

In the following, we consider the above issues and present effective solutions that let us push the envelope in solving such an intractable problem.

5.2 Quantization

The problem of probability state quantization is formulated as follows. Consider a posterior probability state space P^m of m dimensions. That is, a valid state $p \in P^m$ is a vector of m elements $\{p_1, p_2, \ldots, p_m\}$ such that,

$$\sum_{i=1}^{m} p_i = 1.0 \qquad 0 \leq p_i \leq 1 \quad \forall \quad 1 \leq i \leq m \tag{44}$$

Suppose we want to uniformly divide the interval $[0, 1]$ into n_q divisions, i.e., we ordain that the only valid probabilities are $\{0, \delta, 2\delta, \ldots, n_q\delta\}$, where $\delta = 1/n_q$ is the quantization interval, and $n_q + 1$ is the number of quantization levels. For a specified n_q, the objective is to determine a set of m non-negative integers

$\{q_1, q_2, \ldots, q_m\}$ such that the vector $\hat{p} = \{q_1\delta, q_2\delta, \ldots, q_m\delta\}$ represents the quantized probability state. Clearly, it is necessary to have,

$$\sum_{i=1}^{m} q_i = n_q \tag{45}$$

Note that various simplistic scalar quantization rules such as $\{q_i = \lceil p_i/\delta \rceil\}$, or $\{q_i = \lfloor p_i/\delta \rfloor\}$, or even $\{q_i = \lfloor p_i/\delta + 0.5 \rfloor\}$, will result in quantized states that do not satisfy (45) for most choices of n_q. Hence, we need to devise a vector quantization scheme, that transforms any given probability state to a valid quantized probability state. A suitable criterion to choose the integers $\{q_1, q_2, \ldots, q_m\}$ could be to minimize the Euclidean distance between the quantized and unquantized probability states.

Formally, the optimal choice of the quantization vector $q = \{q_1, q_2, \ldots, q_m\}$ minimizes the Euclidean distance measure between the absolute and quantized probability states defined by,

$$d(q) = \sum_{i=1}^{m} (p_i - q_i\delta)^2 \tag{46}$$

subject to the constraint,

$$\sum_{i=1}^{m} q_i = n_q \tag{47}$$

This is a resource allocation problem with quadratic cost function which has a well-known optimal solution procedure via greedy approach [5]. This approach starts by assigning zeros to all q_i, and incrementing one q_i at a time by 1, that results in the maximum decrease of the cost function in (46). However, a direct application of this algorithm requires mn_q computations of cost function decrements (mn_q multiplications). In the following, we present a technique that converges to the optimal solution requiring at most m^2 computations of cost function decrements and m divisions. Our technique results in substantial computational savings for large values of n_q.

The basic idea involved in our technique is to compute a fast, but accurate first estimate of the quantization levels, and then use the greedy algorithm from that point on, instead of starting from an all-zero q vector. With this in mind, let us now consider the following version of the above problem with a tighter constraint set:

$$\text{Minimize} \quad d(q) = \sum_{i=1}^{m} (p_i - q_i\delta)^2 \tag{48}$$

subject to the constraint,

$$q_i\delta \leq p_i \qquad \forall \quad 1 \leq i \leq m, \quad 0 \leq q_i \tag{49}$$

Suppose, the solution to the above version is given by $\hat{q} = \{\hat{q}_1, \hat{q}_2, \ldots, \hat{q}_m\}$. The original problem can be reformulated in terms of this partial solution as follows:

$$\text{Minimize } d(r) = \sum_{i=1}^{m} (p_i - \hat{q}_i\delta - r_i\delta)^2 \tag{50}$$

subject to the constraint,

$$\sum_{i=1}^{m} r_i = n_q - \sum_{i=1}^{m} \hat{q}_i \tag{51}$$

If $\{r_i\}$ are constrained to be positive, then an appropriate change of variables results in the same resource allocation problem as in (46), but with a reduced resource constraint. It can be easily shown that the resource constraint $n_q - \sum_{i=1}^{m} \hat{q}_i$ in the reduced problem can never exceed m. Then, a quick solution of the problem in (48) would reduce the number of cost function computations from mn_q to m^2. In the following, we present the optimal solution to the modified problem and show that $\{r_i\}$ are all positive for the optimal solution, allowing us to use the greedy approach to solve the reduced resource allocation problem.

Lemma 1 *The optimal solution to the modified problem in (48) is given by,*

$$\hat{q}_i = \lfloor p_i/\delta \rfloor \qquad \forall \quad 1 \leq i \leq m \tag{52}$$

Proof: Clearly $\{\hat{q}_i\}_{i=1}^m$ is feasible. Increasing any $\{\hat{q}_i\}$ results in an infeasible solution. Decreasing any $\{\hat{q}_i\}$ results in cost increase. Hence, $\{\hat{q}_i\}_{i=1}^m$ is an optimal feasible solution to the problem in (48) and (49).

Lemma 2 *The resource variables $\{r_i\}_{i=1}^m$ of the reduced problem in (50) are non-negative for the optimal solution.*

Proof: It suffices to show that the optimal solution vector q satisfies

$$q_i \geq \hat{q}_i \qquad \forall \quad 1 \leq i \leq m \tag{53}$$

Suppose this is not true, and that for some k, $q_k = \hat{q}_k - 1$. Since the elements in q satisfy (45) and since the elements in $\hat{q}$ sum to an integer less than or equal to n_q, there must exist some index $\overline{m}$ such that $q_{\overline{m}} = \hat{q}_{\overline{m}} + 1$. Assume without loss of generality that $\overline{m} = (k+1)$. Now, consider an alternative quantization vector q^a where,

$$q_i^a = \begin{cases} \hat{q}_k & i = k \\ \hat{q}_{\overline{m}} & i = k+1 \\ q_i & \text{otherwise} \end{cases} \tag{54}$$

The difference between the cost functions induced by the above two quantization vectors can be written as,

$$\begin{aligned} d(q) - d(q^a) &= (p_k - \hat{q}_k\delta - \delta)^2 + (p_{k+1} - \hat{q}_{k+1}\delta + \delta)^2 \\ &\quad -(p_k - \hat{q}_k\delta)^2 - (p_{k+1} - \hat{q}_{k+1}\delta)^2 \\ &= \delta[p_{k+1} - \hat{q}_{k+1}\delta + (\hat{q}_k + 1)\delta - p_k)] \\ &\geq 0 \end{aligned} \tag{55}$$

The final inequality in the above equation follows directly from the definition of $\hat{q}_i = \lfloor p_i/\delta \rfloor$, implying $p_i \geq \hat{q}_i$ and $(\hat{q}_i + 1)\delta \geq p_i$. Thus we see that a solution q violating the statement of the lemma cannot be optimal. Hence, it follows that the optimal solution always contains $\hat{q}$ thereby forcing the variables r_i in (50) to be non-negative.

It is instructive to determine the total number of distinct discrete probability states resulting from such a quantization scheme. This can be formally written as the number of distinct solutions in non-negative integers for the following equation:

$$\sum_i^m q_i = n_q \tag{56}$$

Lemma 3 *The total number of distinct discrete probability states arising out of quantization of an m-dimensional probability space (m failure sources) into n_q divisions along each probability coordinate is given by $\binom{n_q+m-1}{m-1}$.*

Proof: Consider a line segment of length n_q, with points $P_0, P_1, \ldots, P_{n_q}$ marked out at integer intervals. Any solution (in positive integers) of (56) corresponds to a decomposition of this segment into m pieces whose lengths are positive integers. The $m-1$ end points of these pieces (other than P_0 and P_{n_q}) must be chosen from among the $n_q - 1$ points $P_1, P_2, \ldots, P_{n_q-1}$. This can be done in $\binom{n_q-1}{m-1}$ ways. However, note that we are looking for all non-negative solutions of the problem. Adding m to both sides of (56), we get

$$\sum_i^m (q_i + 1) = n_q + m \tag{57}$$

Now the variables $y_i = q_i + 1$ are strictly positive if q_i are non-negative, and there are $\binom{n_q+m-1}{m-1}$ ways of choosing distinct, positive y_i variables. Thus, the number of non-negative integral solutions is identical and is $\binom{n_q+m-1}{m-1}$.

5.3 Data Structures for State Space Representation

In the previous section, we presented the quantization procedure for the discretization of posterior probability space. The DP recursion described in (42) requires the following fundamental operations to be performed repeatedly:

- Given a quantized state x_i, compute the resulting states $T_{jp}(x_i)$ and $T_{jf}(x_i)$ due to pass and fail outcomes of an admissible test j.
- Access the cost-to-go estimates at the states $T_{jp}(x_i)$ and $T_{jf}(x_i)$ obtained in the earlier cycle of computation and revise the cost-to-go estimate at x_i.

A naive approach to address the above operations is to precompute the mapping functions $T_{jp}(.)$ and $T_{jf}(.)$, and store the appropriate pointers in each x_i, so that $T_{jp}(x_i)$ and $T_{jf}(x_i)$ states can be accessed directly from x_i for any given test j. This requires an extra storage of $n\binom{n_q+m-1}{m-1}$ pointer variables (which require 4 bytes each on most computer systems), where n_q is the number of quantization divisions of each probability coordinate, m is the number of failure sources and n is the number of tests. Clearly, a runtime calculation of these mapping functions and efficient data structures that enable fast access of the transformed states, would free up so much valuable memory space that we would be able to solve a much larger dimensional problem than is possible with the above simplistic approach.

However, this approach requires us to devise methods to:

- enumerate and store the quantized states in efficient data structures.
- access the cost-to-go for a given state.

These are no simple tasks, since a simplistic table storage of states (each state is a collection of m integers) takes up $\binom{n_q+m-1}{m-1}m\lceil\log_{10} n_q\rceil$ bytes of memory space on conventional computer systems (assuming that the integers are concatenated to form a string). And random access of a state in such a table requires an average of $\binom{n_q+m-1}{m-1}/2$ comparisons.

In the following, we present a highly storage-efficient, fast-access data structure tuned for this purpose. We first need to introduce some notation in order to give a formal description of the data structures involved. Consider a directed graph $T = (V, E)$ where, V is the set of vertices (nodes) and E is the set of edges. In addition, let T be a directed rooted tree having one vertex which is the head of no edges (called the root) and each vertex except the root is the head of exactly one edge. The relation (v, w) is an edge of T denoted by $v \rightarrow w$. If $v \rightarrow w$, then v is termed parent of w and w is the child of v. Let the function $d(v)$ represent the depth of the node v in the rooted tree.

In order to illustrate why rooted tree is chosen to represent the set of discretized probabilities, let us consider an example system of $m = 4$ failure sources and $n_q = 3$ quantization intervals. We then obtain following quantization vectors (shown in the next page.)

Blanks are used whenever q_i remained unchanged from its previous value in order to bring out the similarity of the enumerated state space to a rooted tree. Also, note that q_3 and q_4 are intentionally bunched together, since the last coordinate (in this case q_4) is fixed when the first $m-1$ coordinates are defined, hence its storage can be eliminated. By placing a node at every non-blank entry in the above table and connecting nodes from left to right, i.e., q_1 nodes to q_2 nodes, q_2 nodes to q_3 nodes, we can form a directed rooted tree, where every node is a child of just one parent. The nodes in the first layer (q_1 nodes) can be assumed to be emanating from a single dummy node q_0 for the sake of completeness.

q_1	q_2	q_3, q_4
0	0	0,3
		1,2
		2,1
		3,0
	1	0,2
		1,1
		2,0
	2	0,1
		1,0
	3	0,0
1	0	0,2
		1,1
		2,0
	1	0,1
		1,0
	2	0,0
2	0	0,1
		1,0
	1	0,0
3	0	0,0

The data structure based on the above directed rooted tree would consist of the following elemental structures: The elemental data structures for representing the above rooted tree and the pseudo-code for the associated state-access routines are described in Appendix A.

The total number of nodes in such a rooted tree structure is bounded by $2\binom{n_q+m-1}{m-1}$, and the combined memory requirement for a DP scheme utilizing these data structures is no more than $7\binom{n_q+m-1}{m-1}$ bytes on most conventional computer systems. In addition to being inexpensive in terms of storage, note that the access to the cost and policy corresponding to a given quantized state does not take more than $m-1$ operations, making it attractive for runtime computation of test mapping functions.

5.4 Terminal Cost Function

As defined earlier, the terminal cost function is the probabilistic cost incurred when the testing is stopped at a given quantized probability state. We define the following cost function for our DP implementation:

$$f(q) = f_{i'} + (1 - q(i')/n_q)C_{Ri'} + \sum_{i=1, i\neq i'}^{m} C_{Mi}q(i)/n_q \tag{58}$$

where

$$i' = \arg\max_i q(i) \tag{59}$$

This definition corresponds to repairing the component with the highest posterior probability value. It makes sense to choose this terminal cost because of the following reason: when the test costs are significantly lower than the false repair costs and missed repair costs (which is usually the case in practice), then there should be an incentive to apply another test and skew the probability distribution to reduce the entropy of the state. For instance, the uncertainty in the state (0.1,0.9) is less than that of the state (0.2,0.8) and hence should have a lower terminal cost. However, the terminal cost difference between the states (0.2,0.8) and (0.21,0.79) should not be sizable. The above definition conforms to this principle and also gives us a consistent stopping rule: testing should be stopped when the average cost incurred after applying any test is higher than the cost of stopping at the present state.

5.5 Simulation Results

5.5.1 Comparison with DP

In order to compare the performance of the information heuristics and certainty equivalence with DP, we considered two small systems; one with 3 failures and 3 tests and another with 5 failures and 5 tests. For these systems, it is possible to quantize the posterior proabilities into very small intervals, thus resulting in an accurate implementation of the Dynamic Programming recursion. Specifically the two systems we considered are described below:

Table 1: Parameters of System 1

Number of Faults = 3
Number of Tests = 3
Number of DP Quantization Levels = 500
D-Matrix
1 1 0
0 1 1
0 0 1
Test Costs 1.0 1.0 1.0
False Repair Costs 100.0 100.0 100.0
Missed Repair Costs 100.0 100.0 100.0
Prior Probs of Faults 0.25 0.35 0.4

Table 2: Parameters of System 2

Number of Faults = 5
Number of Tests = 5
Number of DP Quantization Levels = 40
D-Matrix
1 1 0 0 0
0 1 1 0 0
0 0 1 1 0
0 0 0 1 1
0 0 0 0 1
Test Costs 1.0 1.0 1.0 1.0 1.0
False Repair Costs 100.0 100.0 100.0 100.0 100.0
Missed Repair Costs 100.0 100.0 100.0 100.0 100.0
Prior Probs of Faults 0.25 0.2 0.3 0.15 0.1

Tables 3-6 show the comparitive performance of various algorithms (multi-step look-ahead DP, multi-step information heuristics and Certainty Equivalence) for system 1 for various values of test unreliabilities. Tables 7-10 show the comparitive performance of these algorithm for System 2. Tables 11-12 show the comparative performance of the information heuristic and certainty equivalence techniques for Graham and Garey's pathological example [6] with $m = 10$. Note that INFO(k) denotes information heuristics with k-step look-ahead, DP(k) denotes dynamic programming with k-step look-ahead, and CE denotes the Certainty Equivalence technique. It is observed that for low values of test unreliabilities, the heuristic techniques have resulted in near-optimal solutions. However, their performance degrades as the probabilities of false alarm and missed detection were increased. Also, it is interesting to note that there is not much difference between the performances of info-heuristic technique and certainty equivalence approach for systems 1 and 2 for low values of test unreliabilities. Another interesting observation is that CE resulted in consistently lower probability of error compared to information heuristics. However, for the worst case example of Graham and Gary, CE was always better than the information heuristics.

Metrics	DP(1)	DP(3)	DP(7)	DP(15)
Ave. Test Length	2.29521	2.93849	2.94194	2.9416
Ave. Testing Cost	11.3974	7.88139	7.48886	7.1705
Prob. of Error	0.0455903	0.0247351	0.0227501	0.021155
Ave. Info Gain	0.678008	0.533618	0.534378	0.535792

Table 3: Comparison of Various DP Methods for Pf=0.05, Pm=0.05(System 1)

Metrics	INFO(1)	INFO(2)	INFO(3)	CE
Ave. Test Length	2.29273	2.30669	2.54352	2.29553
Ave. Testing Cost	11.515	11.1669	11.4625	11.3658
Prob. of Error	0.0461903	0.0443203	0.0446153	0.0454103
Ave. Info Gain	0.678128	0.677073	0.649883	0.677529

Table 4: Comparison of Various Heuristic Methods for Pf=0.05, Pm=0.05(System 1)

Metrics	DP(1)	DP(3)	DP(7)	DP(15)
Ave. Test Length	2.96716	3.63474	3.63336	4.10859
Ave. Testing Cost	17.9693	8.04204	7.95276	8.39895
Prob. of Error	0.0750394	0.02205	0.0216099	0.0214649
Ave. Info Gain	0.392182	0.451167	0.45221	0.433815

Table 5: Comparison of Various DP Methods for Pf=0.10, Pm=0.10(System 1)

Metrics	INFO(1)	INFO(2)	INFO(3)	CE
Ave. Test Length	3.07231	3.08141	3.09564	3.08723
Ave. Testing Cost	12.0591	12.6102	12.3495	12.6811
Prob. of Error	0.0449601	0.0476701	0.0462951	0.0479951
Ave. Info Gain	0.456213	0.45486	0.453465	0.453524

Table 6: Comparison of Various Heuristic Methods for Pf=0.10, Pm=0.10(System 1)

Metrics	DP(1)	DP(3)	DP(7)	DP(15)
Ave. Test Length	2.41679	3.38965	3.53795	3.52495
Ave. Testing Cost	21.4411	13.1638	13.4766	13.8521
Prob. of Error	0.0952055	0.0490145	0.0497751	0.0517699
Ave. Info Gain	0.651796	0.563471	0.562667	0.562444

Table 7: Comparison of Various DP Methods for Pf=0.05, Pm=0.05(System 2)

Metrics	INFO(1)	INFO(2)	INFO(3)	CE
Ave. Test Length	2.56769	2.56358	2.56213	4.41764
Ave. Testing Cost	19.4079	18.9453	19.1493	10.6517
Prob. of Error	0.0842089	0.0819036	0.0829637	0.0312653
Ave. Info Gain	0.667016	0.669341	0.668698	0.519862

Table 8: Comparison of Various Heuristic Methods for Pf=0.05, Pm=0.05(System 2)

Metrics	DP(1)	DP(3)	DP(7)	DP(15)
Ave. Test Length	3.7586	4.23867	4.30183	6.97485
Ave. Testing Cost	24.868	17.8838	21.7445	16.1277
Prob. of Error	0.105529	0.0682713	0.0872903	0.0458495
Ave. Info Gain	0.426733	0.440533	0.482891	0.285431

Table 9: Comparison of Various DP Methods for Pf=0.10, Pm=0.10(System 2)

Metrics	INFO(1)	INFO(2)	INFO(3)	CE
Ave. Test Length	3.09077	3.0923	3.08582	5.10162
Ave. Testing Cost	29.0598	29.1433	28.8392	19.0721
Prob. of Error	0.129892	0.130332	0.128832	0.0698944
Ave. Info Gain	0.48225	0.482189	0.483567	0.405789

Table 10: Comparison of Various Heuristic Methods for Pf=0.10, Pm=0.10(System 2)

Metrics	INFO(1)	INFO(2)	INFO(3)	CE
Ave. Test Length	17.5606	17.5607	18.7088	15.1227
Ave. Testing Cost	17.5606	17.5607	18.7088	15.1227
Prob. of Error	0	0	0	0
Ave. Info Gain	0.0241986	0.0241994	0.0226679	0.0282088

Table 11: Comparison of Various Methods for Pf=0.10, Pm=0.10(Gary' model with m=10)

Metrics	INFO(1)	INFO(2)	INFO(3)	CE
Ave. Test Length	16.4974	16.4957	17.4856	9.92901
Ave. Testing Cost	16.4974	16.4957	17.4856	9.92901
Prob. of Error	0	0	0	0
Ave. Info Gain	0.0257201	0.0257262	0.0242757	0.0424652

Table 12: Comparison of Various Methods for Pf=0.05, Pm=0.05(Gary' model with m=10)

6 Top-Down Graph Search Algorithms

The top-down algorithms described in [7] can be readily applied even when the tests are imperfect. This is because, the HEFs (required for AO*based algorthms) and the information gain expressions depend only on the posterior probability distribution of the failure sources at the current ambiguity node. These posterior probabilities can be computed via the Bayes rule given in (3). However, we found that the AO*based algorithms are not useful due to the explosion of the diagnostic strategy even for moderately sized systems. On the other hand, the top-down information heuristic algorithms coupled with the ambiguity pruning technique described earlier, enabled us to solve large systems. Tables 13-18 demonstrate the performance of top-down information heuristic algorithm for various randomly generated systems of different sizes and for various values of false alarm and missed detection probabilities of tests. Note that, α denotes the false alarm probability, and β denotes the missed detection probability. The following performance indicators were collected and listed in these tables:

- J_c is the expected testing cost
- J_r is the expected repair cost composed of the missed repair and false repair costs
- J_N is the average ambiguity group size
- n_l is the number of leaf nodes in the diagnostic strategy
- n_e is the total number of nodes in the decision tree

We can see that even the slightest uncertainty in the test outcomes results in large diagnostic trees with increased testing and repair costs, albeit with tolerable values of average ambiguity group sizes. Table 19 shows the performance of the top-down information heuristic algorithm for random systems of various sizes with fixed test uncertainties ($\alpha = 0.01, \beta = 0.01$). We can see that a system containing as many as 2000 failures and 2000 imperfect tests is solved in less than 30 minutes.

(α,β)	J_c	J_r	J_N	n_l	n_e	Time(secs)
(0.00,0.00)	6.468	0.000	1.000	100	199	0.29
(0.01,0.00)	8.047	0.093	1.002	433	865	1.35
(0.02,0.00)	8.184	0.013	1.000	452	903	1.36
(0.03,0.00)	8.197	0.010	1.000	448	895	1.35
(0.04,0.00)	9.608	1.236	1.029	938	1875	3.26
(0.05,0.00)	9.722	1.678	1.035	965	1929	3.31

Table 13: Performance of Top-Down Algorithm for a (100,100) system with false alarms only

(α,β)	J_c	J_r	J_N	n_l	n_e	Time(secs)
(0.00,0.00)	6.468	0.000	1.000	100	199	0.29
(0.00,0.01)	7.893	0.084	1.001	424	847	1.33
(0.00,0.02)	8.000	0.003	1.000	439	877	1.32
(0.00,0.03)	8.055	0.002	1.000	437	873	1.33
(0.00,0.04)	9.275	1.231	1.030	873	1745	3.08
(0.00,0.05)	9.391	1.582	1.033	919	1837	3.19

Table 14: Performance of Top-Down Algorithm for a (100,100) system with missed detections only

(α,β)	J_c	J_r	J_N	n_l	n_e	Time(secs)
(0.00,0.00)	6.468	0.000	1.000	100	199	0.29
(0.01,0.01)	9.013	0.274	1.005	889	1777	2.95
(0.02,0.02)	9.129	0.037	1.001	947	1893	3.01
(0.03,0.03)	9.155	0.025	1.000	952	1903	3.03
(0.04,0.04)	11.410	8.603	1.266	2650	5299	13.40
(0.05,0.05)	11.434	10.964	1.286	2908	5815	14.27

Table 15: Performance of Top-Down Algorithm for a (100,100) system with false alarms and missed detections

(α,β)	J_c	J_r	J_N	n_l	n_e	Time(secs)
(0.00,0.00)	7.417	0.000	1.000	200	399	1.26
(0.01,0.00)	8.677	0.906	1.016	785	1569	6.43
(0.02,0.00)	8.874	0.311	1.005	926	1851	6.64
(0.03,0.00)	8.984	0.227	1.003	953	1905	6.67
(0.04,0.00)	10.157	2.159	1.073	1874	3747	16.39
(0.05,0.00)	10.208	3.006	1.088	1910	3819	16.63

Table 16: Performance of Top-Down Algorithm for a (200,200) system with false alarms only

(α,β)	J_c	J_r	J_N	n_l	n_e	Time(secs)
(0.00,0.00)	7.417	0.000	1.000	200	399	1.27
(0.00,0.01)	8.640	0.841	1.015	763	1525	6.08
(0.00,0.02)	8.883	0.333	1.006	886	1771	6.27
(0.00,0.03)	9.011	0.169	1.003	924	1847	6.34
(0.00,0.04)	10.235	2.029	1.068	1790	3579	15.46
(0.00,0.05)	10.284	2.744	1.079	1848	3695	15.65

Table 17: Performance of Top-Down Algorithm for a (200,200) system with missed detections only

(α,β)	J_c	J_r	J_N	n_l	n_e	Time(secs)
(0.00,0.00)	7.417	0.000	1.000	200	399	1.26
(0.01,0.01)	9.518	2.538	1.048	1462	2923	13.39
(0.02,0.02)	9.917	0.859	1.015	1907	3813	14.28
(0.04,0.04)	12.062	11.625	1.540	4839	9677	64.50
(0.05,0.05)	12.122	16.071	1.635	5148	10295	66.62

Table 18: Performance of Top-Down Algorithm for a (200,200) system with false alarms and missed detections

m,n	n_l	n_e	Time(secs)
500	2069	4137	90.26
1000	3094	6187	378.89
1500	4136	8271	891.50
2000	5267	10533	1653.06

Table 19: Performance of Top-Down Algorithm for systems of various sizes with $\alpha = 0.01, \beta = 0.01$

7 Summary

In this paper, we considered the problem of test sequencing in the presence of imperfect tests. The test sequencing problem is a partially observed Markov decision problem (POMDP), a sequential multi-stage decision problem wherein the states are probabilities of the set of possible failure sources and information regarding the states is obtained via the results of imperfect tests. The optimal solution for this problem can be obtained by applying a continuous state Dynamic Programming (DP) recursive equation. However, the DP recursion is computationally very expensive owing to the continuous nature of the state vector comprising the probabilities of faults. In order to alleviate this computational explosion, we presented an efficient approach for implementing the DP recursion for this problem. In addition, we presented multi-step DP, multi-step information heuristics and certainty equivalence algorithms for interactive diagnosis of systems with imperfect tests. We also considered various problems with special structure (parallel systems) and derived closed form solutions/index-rules without having to resort to DP. We also presented computational results demonstrating the effectiveness of the information heuristic based top-down graph search algorithm.

A Data Structures and Pseudo-code for DP Implementation

RootedTreeNode
{
    NumberOfChildNodes (Integer)
    ArrayOfChildNodes (Pointer to RootedTreeNode)
    IndexIntoCostVector (Integer)
}
Data Structure to Represent a Node in the Rooted Tree

CostVectorNode
{
    EstimateOfCostToGo (Floating Point Variable)
    Policy (Integer)
}
Data Structure to Represent a Node in the Cost Vector

Procedure RootedStateTreeConstructor

```
Inputs:
RootedTreeNode CurrentNode By Reference
Integer SumTillNow By Value
Integer FaultNum By Value
Integer StateIndex By Reference
CostVectorArray CostVector By Reference
Integer NumLevels By Value
IntegerArray CurrentState By Reference
{
     if(FaultNum = NumFaults-1)
     {
          CostVector[StateIndex].Cost =
               TerminalCostFunction(CurrentState)
          CurrentNode.IndexIntoCostvector = StateIndex
          StateIndex = StateIndex+1
          return
     }

     CurrentNode.NumberOfChildNodes = NumLevels - SumTillNow +1
     (create storage for childnodes too)
     for i=1 to CurrentNode.NumberOfChildNodes
     {
          CurrentState[FaultNum+1] = i
          Invoke RootedStateTreeConstructor() with following inputs:
               CurrentNode.ArrayOfChildNodes[i]
               SumTillNow+i
               FaultNum+1
               StateIndex
               CostVector
               NumLevels
               CurrentState
     }
}
```

Algorithm for Rooted Tree Construction

```
Procedure GetCostAndPolicyForQuantizedState
Inputs:
RootedTreeNode CurrentNode By Reference
CostVectorArray CostVector By Reference
Integer NumFaults By Value
IntegerArray CurrentState By Reference
Outputs:
Cost
Policy
{
     for i=1 to NumFaults-1
     {
          CurrentNode =
               CurrentNode.ArrayOfChildNodes[QuantizedStateVector[i]]
          return
     }

     StateIndex = CurrentNode.IndexIntoCostVector
```

```
    Cost = CostVector[StateIndex].EstimateOfCostToGo
    Policy = CostVector[StateIndex].Policy
}
```

Algorithm for Accessing Cost and Policy of a Quantized State

References

[1] Gluss, B., "An Optimum Policy for Determining a Fault in a Complex System," *Operations Research*, Vol.8, 1960, pp.512-523.

[2] Nachlas, J.A., Loney, S.R., and Binney, B.A., "Diagnostic Strategy Selection for Series Systems," *IEEE Transactions on Reliability*, Vol. 39, August 1990.

[3] Bellman, R.E., *Dynamic Programming*, Princeton: Princeton University Press, 1957.

[4] Bertsekas, D.P., *Dynamic Programming: Deterministic and Stochastic Models*, Prentice-Hall, Englewood Cliffs, NJ, 1987.

[5] Ibaraki, T., and Katoh, N., *Resource Allocation Problems: Algorithmic Approaches*, The MIT Press, Cambridge, MA, 1988.

[6] Loveland, D.W., "Performance bounds for Binary Testing with Arbitrary Weights," *Acta Inform.* , no. 22, pp.101-114, 1985.

[7] Raghavan, V., Pattipati, K.R., Shakeri, M., "Optimal and Near-Optimal Test Sequencing Algorithms with Realistic Test Models", submitted to *IEEE Transactions on Systems, Man, and Cybernetics.*

[8] Monahan, G., "A survey of partially observable Markov decision processes," *Mgmt. Sci.*, vol. 28, pp. 1-16, 1982.

[9] Smallwood, R.D., and Sondik, E.J., "Optimal control of partially observable processes over the finite horizon," *Operations Research*, vol. 21, pp. 1071-1088, 1973.

[10] Benkoski, S.J., Monticino, M.G., and Weisinger, J.R., "A surver of search theory literature," *Naval research Logistics*, vol. 38, no. 4, pp. 469-494, 1991.

[11] Kadane, J.B., "Optimal whereabouts search," *Operations Research*, vol. 19, pp. 894-904, 1971.

[12] Tognetti, K.P., "An optimal strategy for whereabouts search," *Operations Research*, vol. 16, pp. 209-211, 1968.

[13] Raghavan, V., Willett, P., Pattipati, K., Kleinman, D., "Optimal measurement sequencing in M-ary hypothesis testing problems," in *Proc. 1992 American Control Conference*, Chicago, IL, June 1992.

[14] Sheridan, T.B., "On how often a supervisor shoudl sample," *IEEE Trans. Syst. Man Cyber.*, vol. 13, no. 1, pp. 37-46, Jan/Feb 1965.

[15] Dobbie, J.M., "Some search problems with false contacs," *Operations Research*, vol. 21, pp. 907-925, 1973.

[16] Stone, L.D., and Stanshine, J.A., "Optimal search using uninterrupted contact investigation," *SIAM J. Appl. Math.*, vol. 20, pp. 241-263, 1971.

[17] Stone, L.D., and Stanshine, J.A., "Optimal search in the presence of poisson-distributed false targets," *SIAM J. Appl. Math.*, vol. 23, pp. 6-27, 1972.

[18] Wald, A., *Sequential Analysis*, New York: John Wiley, 1947.

Submitted to IEEE Transactions on Systems, Man, and Cybernetics

Sequential Testing Algorithms for Multiple Fault Diagnosis

Mojdeh Shakeri[†], Vijaya Raghavan[†], Krishna Pattipati[††], and Ann Patterson-Hine[†††]

Qualtech Systems Inc.[†]
Box-407, Mansfield Center, CT 06250
e-mail: mojdeh@sol.uconn.edu

Department of Electrical and Systems Engineering[††]
University of Connecticut, Storrs, CT 06269-3157
e-mail: krishna@sol.uconn.edu

NASA-Ames Research Center, Mail Stop 269-4[†††]
Moffett Field, CA 94035-1000
e-mail: Ann_Patterson-Hine@styx.arc.nasa.gov

Abstract

In this paper, we consider the problem of constructing optimal and near-optimal test sequencing algorithms for multiple fault diagnosis. The computational complexity of solving the optimal multiple-fault isolation problem is super-exponential, that is, it is much more difficult than the single-fault isolation problem, which, by itself, is NP-hard[1][7]. By employing concepts from information theory and AND/OR graph search, we present several test sequencing algorithms for the multiple fault isolation problem. These algorithms provide a trade-off between the degree of suboptimality and computational complexity. Furthermore, we present novel diagnostic strategies that generate a diagnostic directed graph (digraph), instead of a diagnostic tree, for multiple fault diagnosis. Using this approach, the storage complexity of the overall diagnostic strategy reduces substantially. The algorithms developed herein have been successfully applied to several real-world systems. Computational results indicate that the size of a multiple fault strategy is strictly related to the structure of the system.

[1]This means that the computational requirements of an optimal algorithm cannot be bounded by a polynomial function of the number of failure sources and/or the number of tests.

1 Introduction

The complexity associated with the maintenance of large integrated systems, such as the space shuttle or a modern aircraft consisting of mechanical, electro-mechanical and hydraulic subsystems, presents formidable challenges to manufacturers and end users. This is due to the large number of failure sources and the need to quickly isolate and rectify such failures with minimal down time. In addition, for redundant (fault-tolerant) systems and for systems with little or no opportunity for repair or maintenance during their operation (e.g., Hubble telescope, space station), the assumption of at most a single failure in the system between consecutive maintenance actions is unrealistic. Thus, the efficient maintenance of complex systems requires advanced diagnostic algorithms for multiple fault isolation.

A review of existing literature [13] showed that multiple-fault diagnosis using artificial intelligence techniques is too expensive and slow for large systems. Davis [2, 3] described a fault diagnosis system that reasons from the knowledge of structure and behavior. Failure candidate generation in this approach occurs in three basic steps: circuit simulation and discrepancy collection, potential candidate determination, and global consistency determination using constraint suspension techniques. The approach of Davis [2, 3] can be extended to diagnose multiple faults. However, this approach would require the application of constraint suspension to all possible combinations of components, and consequently, suffers from computational explosion. De Kleer and Williams [4] presented a model-based approach to fault diagnosis. By keeping track of multiple sets of consistent and inconsistent components, their algorithm generates minimal sets of faulty candidates, rather than generating all possible candidates. This approach requires the complete specification of system components, the state and observed variables associated with each component, and the functional relationships among the state variables. However, the precise information required by these models is typically not available for complex systems and is too costly to obtain. In addition,

because of extensive use of functional simulation, this approach is extremely slow, and, thus, is not appropriate for fault diagnosis in large scale systems with the complexities of many orders of magnitude more than the examples presented in [4]. Sheppard and Simpson [19] provided a formal analysis of the multiple failure problem in the context of information flow model. They discussed the computational complexity of several algorithms for diagnosing multiple failures, and developed algorithms to generate multiple fault diagnoses for a given ambiguity group. However, this method does not take into account the failure probabilities of components or test costs.

In this paper, we present several multiple fault test sequencing algorithms. First, we extend the single-fault strategy of our previous work [7, 8, 9, 11] to diagnose multiple faults by successive replacement of single fault candidates. Using this strategy, we seek to isolate the potential single-fault candidates, then double-fault candidates, and so on. Since a component may be repaired/replaced before confirming that it is indeed faulty, the probability of false alarm error or RTOK (retest OK) is higher than that with multiple fault strategies that use all informative tests before repairing a component in the system. Then, we focus on developing a class of Sure strategies [14] for diagnosing multiple faults that employ all informative tests before diagnosis. The basic idea of these strategies is to find one or more definitely failed components, while not making an error when other co-existing faults are present. Using these algorithms, the storage and computational complexity of the multiple fault diagnostic strategy are reduced substantially.

The paper is organized as follows. In section 2, we formulate the test sequencing problem. Because of extensive use of single fault test sequencing algorithms in solving the multiple fault diagnosis problem, we describe single fault test sequencing algorithms in section 3. In section 4, we present the problem of diagnosing multiple failures using a single fault diagnostic strategy. In section 5, we present an extended single fault strategy to diagnose multiple failures. Near-optimal multiple fault strategies are discussed in section 6. In section 7, we summarize the results and

discuss future research issues. Throughout, an example from [7] will be used to illustrate the concepts and the proposed diagnostic strategies. In addition, we apply our algorithms to several real-world examples.

2 Problem Formulation

The multiple fault test sequencing problem, in its simplest form, is defined by the five-tuple (S, P, T, C, B), where

1. $S = \{s_1, ..., s_m\}$ is a set of independent failure sources associated with the system;
2. $P = [p(s_1), ..., p(s_m)]$ is the a priori probability vector associated with the set of failure sources S;
3. $T = \{t_1, t_2, ..., t_n\}$ is a finite set of n available binary outcome tests, where each test t_j checks a subset of S;
4. $C = \{c_1, c_2, ..., c_n\}$ is a set of test costs measured in terms of time, manpower requirements, or other economic factors, where c_j is the cost of applying test t_j;
5. $B = [b_{ij}]$ is a binary matrix of dimension $m \times n$ which represents the relationship between the set of failure sources S and the set of tests T, where $b_{ij} = 1$ if test t_j monitors failure source s_i; otherwise, $b_{ij} = 0$.

The problem is to design a testing strategy that unambiguously isolates the failure sources with minimum expected testing cost $f = \sum_{S_I \subseteq S} \sum_{t_j \in PT_I} p(S_I) c_j$, where PT_I is the set of applied tests (performed tests) in the path leading to the isolation of the set of failure sources S_I, and $p(S_I)$ is the probability of the set of failure sources S_I (see Appendix A). The AND/OR sequential test strategy is represented in the form of a tree or a graph, where the OR nodes represent the suspect

sets of failure sources, AND nodes are tests applied at various OR nodes, and the leaves are the isolated failure sources.

For notational convenience, we define failure signature FS_i to denote a set associated with failure source s_i that indicates all the tests that monitors failure source s_i, i.e., $FS_i = \{t_j | b_{ij} = 1 \text{ for } 1 \leq j \leq n\}$. Furthermore, we assume that the failure signature of a multiple-failure is the union of failure signatures of individual failures.

3 Single Fault Testing Strategies

In a single fault strategy, it is assumed that the system is tested frequently enough that at most one component has failed. The single fault diagnosis problem, in its simplest form, is the five-tuple $(\mathbf{S}, \mathbf{P}, T, C, D)$, where

- $\mathbf{S} = S \vee \{s_0\} = \{s_0, s_1, ..., s_m\}$ is a set of failure sources, where s_0 is a dummy failure source denoting fault-free condition and $\vee$ denotes the union of two sets;

- $\mathbf{P} = [p_0, p_1, ..., p_m]$ is the *conditional* probability vector associated with the set of failure sources S based on a single fault assumption, where p_0 is the probability of fault-free condition, s_0. In Appendix A, we show that the conditional probability p_i is related to the unconditional prior probabilities $\{p(s_i)\}$ via:

$$p_0 = \frac{1}{1 + \sum_{k=1}^{m} \frac{p(s_k)}{1-p(s_k)}} \tag{1}$$

$$p_i = \frac{\frac{p(s_i)}{1-p(s_i)}}{1 + \sum_{k=1}^{m} \frac{p(s_k)}{1-p(s_k)}} \quad \text{for} \quad i = 1, ..., m$$

- T and C are as defined in Section 2;

- $D = [d_{ij}]$ is a binary test matrix of dimension $(m+1) \times n$, where $d_{0j} = 0$ for $1 \leq j \leq n$, and $d_{ij} = b_{ij}$ for $1 \leq i \leq m$ and $1 \leq j \leq n$.

The algorithms for designing optimal single-fault diagnostic strategies are based on dynamic programming (DP) [1], and AND/OR graph search procedures. The DP technique is based on a bottom-up procedure, and has storage and computational requirements of $O(3^n)$ for even the simplest test sequencing problem. The AND/OR graph search algorithms are top-down heuristic graph search procedures that employ a cost-to-go estimate to speed up the solution search process [7]. A novel feature of this approach is that the cost-to-go estimate (termed the Heuristic Evaluation Function (HEF)) is derived from Huffman coding and entropy. These information theoretic lower bounds ensure that an optimal solution is found using the AO*, HS, and CF search algorithms [9]. In addition, because of the top-down nature of the AND/OR graph search algorithms, several near-optimal search algorithms have been derived: (1) AO^*_ϵ algorithm, (2) limited search AO*, and (3) Multi-step information heuristics. Furthermore, because of their top-down nature, these algorithms extend naturally to: (1) modular diagnosis, (2) precedence constraints, setup operations, and resources and (3) rectification. The algorithms have been implemented in a software package, termed TEAMS (Testability Engineering And Maintenance System[9]). For convenience, these algorithms are referred to as the **TEAMS-S** algorithms [11].

Example 1.a: In this example, we consider the same system as in [7]. In this system, there are five failure sources s_1,..., s_5. The set of five tests, labeled t_1,..., t_5, may be used to identify the unknown failure sources. The test matrix, along with the a priori probabilities of failure sources and test costs, is shown in Table 1. Based on the assumption of at most a single fault in the system, the set of failure aspects $\mathbf{S} = \{s_0, s_1, ..., s_5\}$, with the concomitant conditional probability vector **P** =[0.700, 0.01, 0.020, 0.100, 0.050, 0.120]. An optimal single fault test strategy for this example is shown in Figure 1. For this test strategy, the average test cost is $J = \sum_{i=0}^{m} \sum_{t_j \in PT_i} p_i.c_j$=2.18, where PT_i is the set of applied tests (performed tests) in the path leading to the isolation of failure source $s_i \in \mathbf{S}$.

FAILURE SOURCES	TESTS TEST COSTS c_j 1 1 1 1 1 t_1 t_2 t_3 t_4 t_5	FAULT PROBABILITIES $p(s_i)$
s_1	0 1 0 0 1	0.014
s_2	0 0 1 1 0	0.027
s_3	1 0 0 1 1	0.125
s_4	1 1 0 0 0	0.068
s_5	1 1 1 1 0	0.146

Table 1: Test Matrix, *a Priori* Probabilities and Test Costs for Example 1.a

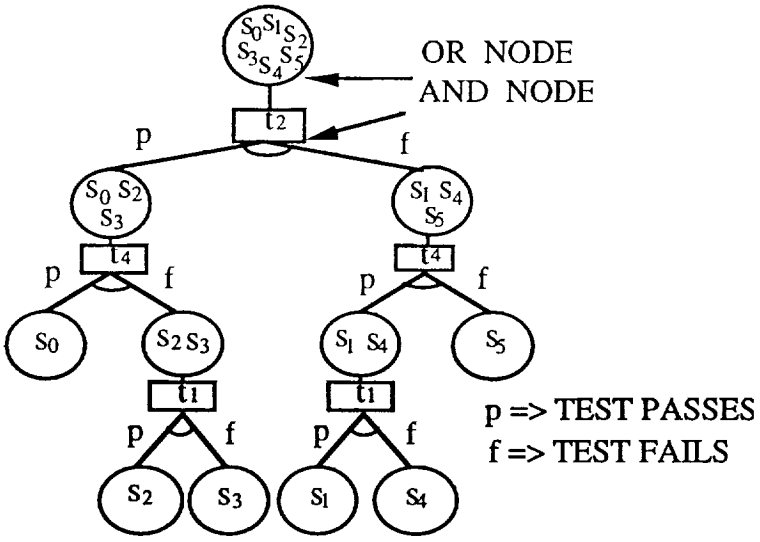

Figure 1: Single-fault Test Strategy for the System of Example 1.a

The single fault assumption may not be valid in situations where the opportunity for frequent maintenance does not exist. In such cases, the single fault strategies can give wrong diagnosis when multiple faults occur. For example, consider a system with S = $\{s_1, s_2, s_3\}$, T = $\{t_1, t_2\}$, TS_1 = $\{s_1, s_3\}$ and $TS_2 = \{s_2, s_3\}$, where test signature TS_j is a set associated with test t_j that indicates

all the failure sources detectable by test t_j, i.e., $TS_j = \{s_i | b_{ij} = 1 \text{ for } 1 \leq i \leq m\}$. Suppose that we perform both tests and that they both fail. Under the single-fault logic, we would conclude that s_3 is faulty. However, if s_1 and s_2 were both faulty, we would observe the same test results. Consequently, the single-fault strategy would make an incorrect diagnosis, when s_1 and s_2 are both faulty.

In the following, we define hidden and masking false failures, which are possible multiple fault candidates at each leaf node of the single fault diagnostic tree. The set of hidden failures HF_i for failure source s_i is given by:

$$HF_i = \{s_j | j \neq i \text{ and } (FS_i \cap PT_i) \cup (FS_j \cap PT_i) = (FS_i \cap PT_i)\}$$

In words, HF_i consists of those failure sources whose failure signatures corresponding to the set of applied tests PT_i in the path leading to the isolation of failure source s_i are masked by the failure of s_i, i.e., subset of the failure signature of s_i restricted to PT_i. The set of masking false failures MS_i for failure source s_i consists of those sets whose failure signatures corresponding to PT_i add up to mask the failure of s_i, i.e.,

$$MS_i = \{X | X \subseteq (S - s_i), \cup_{\forall s_k \in X}(FS_k \cap PT_i) = (FS_i \cap PT_i)\}$$

The multiple fault ambiguity group at a leaf node of the single-fault diagnostic strategy where failure source s_i is isolated consists of masking faults MS_i and any combination of masking faults MS_i and hidden faults HF_i with s_i, i.e., $MS_i \cup (MS_i \times \{s_i\}) \cup (2^{HF_i} \times \{s_i\})$, where $\times$ denotes cross product function and 2^{HF_i} is the power set of HF_i. The problem of identifying the set of hidden failures is relatively easy to solve. In contrast, the problem of enumerating the masking false failures for each failure source s_i is computationally expensive. Typically, it requires $O(|PT_i|2^m)$ or $O(2^m)$ operations [10].

4 Multiple-Fault Isolation Using Single-Fault Strategy

One often stated premise is that one can apply single fault strategy repeatedly, until all the faults are isolated. This strategy works well when there are no masking false failures at the leaf nodes of the single-fault diagnostic tree. However, if the set of masking false failures at the leaf nodes are not empty, the single fault strategy will give wrong diagnosis. In order to illustrate this case, let us assume that s_1 and s_3, in Example 1.a, are faulty. Based on single fault diagnostic tree, $t_2 = f$ and $t_4 = f$; and we would assume that s_5 is faulty. After repairing/replacing s_5, we would perform more tests from the root OR node, $t_2 = f$ and $t_4 = f$, i.e., the same test results as before. This is because $\{s_1, s_3\} \in MS_5 = \{\{s_1, s_2\}, \{s_1, s_3\}, \{s_2, s_4\}, \{s_3, s_4\}, \{s_1, s_2, s_3\}, \{s_2, s_3, s_4\}, \{s_1, s_2, s_4\}, \{s_1, s_3, s_4\}, \{s_1, s_2, s_3, s_4\}\}$. In this example, 18 failures out of $32(= 2^5)$ multiple failures can not be isolated by repeatedly using the single fault diagnostic tree. This is because $|MS_5| + |MS_5 \times \{s_5\}| = 18$. The occurrence of masking false failure sets is fairly common. In order to illustrate this, we generated 10 random systems with five components, five tests, P =[0.5, 0.5, 0.5, 0.5, 0.5], and C = {1,1,1,1,1}. Only 2 systems did not have masking sets, and the average size of masking sets based on all systems was 6. Therefore, on the average, 12 multiple failures out of 32 failures can not be isolated in these systems via repetitive application of single fault logic.

In addition to this set of synthetic problems, we have considered several real-world systems. These include:

1. Anticollision Light Control System of the Sea Hawk helicopter with 43 failure sources and 53 tests,

2. An amplifier-filter with 80 failure sources and 25 tests,

3. 1553 Data Bus with 176 faults and 53 test points,

System	m	n	% Leaves with Masking False Failures
Anticollision system	43	53	47.50%
Amplifier Filter	80	25	17.86%
1553 Bus	176	53	20.59%
Goodrich (EDIF)	898	250	0%
Phase Decoder (EDIF)	1644	2147	3.02%

Table 2: Percentage of Leaf Nodes with Masking False Failures

4. A circuit board model (courtesy of Goodrich Aerospace) generated from an EDIF (Electronic Design Interchange Format)[2] netlist containing 898 faults and 250 tests,

5. Phase Decoder model (public domain test circuit for EDIF parsers) with 1644 faults and 2147 tests.

Table 2 shows the percentage of leaf nodes which contain at least one masking false failure. In conclusion, single fault diagnostic tree can be used to isolate multiple failures in systems with no masking sets. However, as the above results show, the masking sets in most systems are not empty. Consequently, practical multiple fault diagnosis algorithms are needed.

5 Multiple Fault Diagnosis Using an Extended Single Fault Testing Strategy

In this approach, we invoke a single fault strategy, and repair/replace the identified component at each leaf node, if any. Then, we check whether the repaired/replaced component at each leaf node is definitely faulty or not. If for any test t_j that failed previously, the cardinality of $TS_j - G$ is one, i.e., $TS_j - G$ contains only one failure source, then the corresponding failure source is definitely faulty, where G is the union of test signatures of previously passed tests. If the repaired/replaced

[2]http://www.cs.man.ac.uk/cad/EDIFTechnicalCentre.

component is definitely faulty, we apply additional tests, if necessary, to isolate the remaining faults. Additional tests can be applied from either the root OR node, or from the first failed test in the path leading to the identification of previous faults. This process ensures that we do not come back to the same leaf node twice.

Alternatively, if the replaced module is not definitely faulty, there exist other sets of components which have the same failure signature as the failure signature of replaced module, i.e., masking false failures. In this case, if we start from the root OR node or the first failed test in the path, we may reach the same leaf node. In order to solve this problem, we remove the replaced modules from the ambiguity group at the current stage of diagnosis, and invoke the single fault strategy TEAMS-S to isolate the remaining suspected components. Then, we repair/replace the identified modules at each leaf node. If the repaired/replaced module at a leaf node of this tree is definitely faulty, we apply additional tests from the root OR node or from the first failed test *after last repair*. On the other hand, if the identified module at a leaf node is not definitely faulty, we update the ambiguity group and invoke single fault strategy as before. This procedure is continued until no test gives further information or the system is fault-free. The extended single fault algorithm is formalized in the next subsection.

Example 1.b: In this example, we consider the same system as in Example 1.a. The extended single fault diagnostic strategy for this example is shown in Figure 2, where the ACTION nodes represent the actions to be performed at the corresponding OR node. Note that the shaded parts of the tree are the same as those in a single fault diagnostic tree of Figure 1. The average testing cost for this case is J =2.780. The joint probability that s_5 is good, and is repaired/replaced is 0.0103.

We applied the extended single fault strategy to several real-world systems. Table 3 shows the times taken to construct an extended single fault diagnostic strategy for several real-world systems.

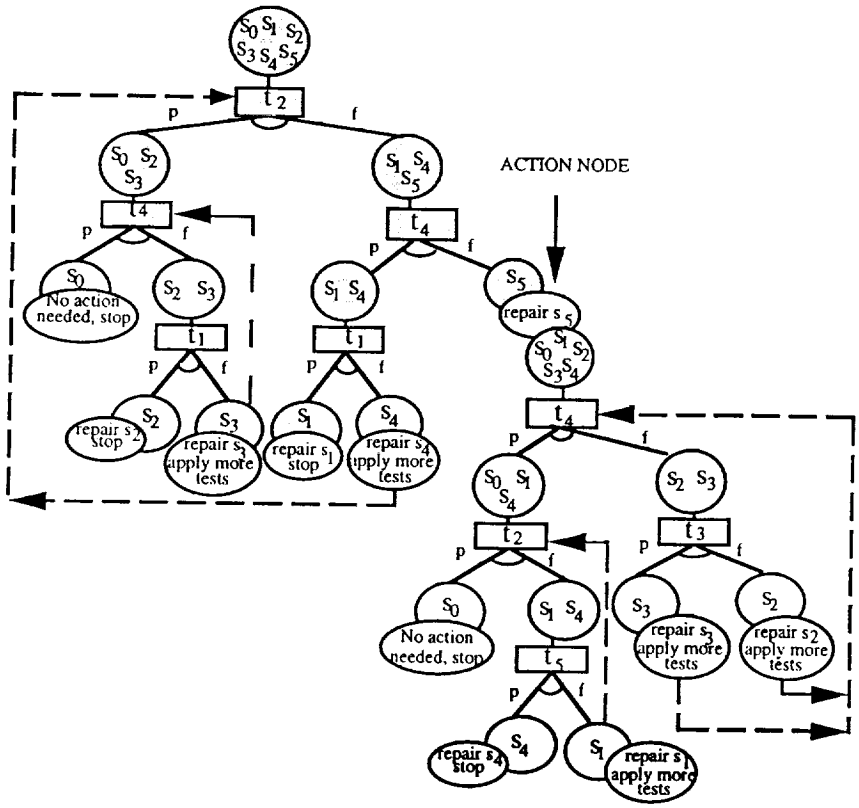

Figure 2: Extended Single-fault Strategy to Diagnose Multi-faults in Example 1.a

System	(m, n)	Time (sec)			
		# Repair Limits			
		1	2	3	All
Anticollision system	(43, 53)	0.27	2.41	2.93	6.93
Amplifier-filter	(80, 25)	0.18	0.63	0.88	0.90
1553 Bus	(176,53)	0.27	1.81	1.81	1.81
Goodrich (EDIF)	(898, 250)	0.88	0.88	0.88	0.88
Phase Decoder (EDIF)	(1644, 2147)	41.59	370.00	691.94	1132.55

Table 3: Solution Times in Seconds Based on Extended Single Fault Strategy for Various Real-world Systems on a SPARC-10

Table 4 shows the number of nodes in the extended single fault diagnostic strategies for these real-world systems. [3]

One drawback of the extended single-fault strategy is that the probability of repairing/replacing a good component, i.e., false alarm error or RTOK (retest OK), is higher than that with multiple fault strategies that employ all informative tests before repairing a component in the system (see section 6.2). Furthermore, in the case of very large systems, it is practical to solve multiple fault isolation problems up to a certain cardinality $\hat{L} \geq 1$, e.g., single or double failures. This is based on the premise that multiple faults of large cardinality are much less likely to occur. However, in an extended single fault strategy, if we stop expanding the diagnostic tree after limited repair actions, say $\hat{L}$, it does not mean that we can diagnose multiple faults up to size $\hat{L}$ using the same tree. This is because a component may be repaired/replaced before confirming that it is indeed faulty.

[3] In order to reduce the search space, the TEAMS-S algorithms preprocess the binary D-matrix as follows: (1) they collapse all the failure sources with the same failure signature to create a new representative failure source, and (2) they eliminate the redundant tests [10] (see section 6.3).

System	(m, n)	# Nodes			
		# Repair Limits			
		1	2	3	All
Anticollision system	(43, 53)	79	897	1065	2371
Amplifier-filter	(80, 25)	55	231	347	347
1553 Bus	(176,53)	67	417	417	417
Goodrich (EDIF)	(898, 250)	37	37	37	37
Phase Decoder (EDIF)	(1644, 2147)	993	8919	16389	25663

Table 4: Number of Nodes in Extended Single Fault Strategy for Various Real-world Systems

5.1 Extended Single-fault Algorithm

Extended single fault algorithm is a recursive function, and must be invoked as **Extended-Single-Fault(** *OR node*, SS), where

- *OR node* is the current OR node,
- SS denotes the suspected faults at the current OR node.

Global variables:

- the root OR node of the diagnostic tree,
- the set of failure sources $S = \{s_1, ..., s_m\}$,
- the a prior probability vector $P = [p(s_1), ..., p(s_m)]$,
- the set of available tests $T = \{t_1, t_2, ..., t_n\}$,
- the set of test costs $C = \{c_1, c_2, ..., c_n\}$,
- the binary test matrix $B = [b_{ij}]$.

Initialization:

- *OR node*=root OR node,
- $SS = \mathrm{S} = S \cup \{s_0\}$.

Algorithm: Extended-Single-Fault(*OR node*, SS)

step 1: Evaluate the conditional probability of the faults in SS using P, P_s.
step 2: Expand the diagnostic tree from the *OR node* by invoking **TEAMS-S** (SS, P_s, T, C, D_s), where D_s contains the failure signatures of the failures in SS.
step 3: DO for each UNSOLVED leaf node,
step 3.1: Action: repair/replace the identified component, if any.
step 3.2: $G \leftarrow$ {repaired/replaced failure sources} $\cup_{t_i=p} TS_i$ for t_i in the path from *OR node* to the leaf node.
step 3.3: $SS \leftarrow SS - G$.
step 3.4: IF for any failed test t_j in the path from *OR node* to the leaf node, $|TS_j - G| = 1$ THEN
- IF $SS = \{s_0\}$,
Action: stop.
ELSE
Action: Apply additional tests from the root OR node or the first failed test after the *OR node*.
END
ELSE
- $SS \leftarrow SS \cup \{s_0\}$.
- **Extended-Single-Fault**(leaf node, SS).
END
END

6 Multiple Fault Testing Strategies

One approach that employs all informative tests before repairing/replacing a component is to consider all possible combinations of failure sources, i.e., 2^S, and generate an optimal multiple fault diagnostic strategy using the single-fault test sequencing algorithm **TEAMS-S**. However, the storage and computational complexity of optimal multiple-fault isolation problem is super-exponential in m. In order to reduce storage complexity, we use a compact set notation [6], and in order to reduce the computational complexity, we present a class of Sure diagnosis strategies for multiple fault isolation.

6.1 Compact Set Notation

Following Grunberg et al. [6], we use the compact notation $A = \Theta(L;\ F_1, ..., F_L;\ G)$ to denote the multiple fault ambiguity group at each OR node. The F_i for $i = 1, ..., L$ and G are subsets

of S = $\{s_0, s_1, ..., s_m\}$; G is the set of known good failure sources (failure free sources), and F_i for $i = 1, ..., L$ are sets that are known to contain at least one definitely failed failure source each, i.e.,

$$\Theta(L; F_1, F_2, ..., F_L; G) = \{X \subseteq \mathrm{S} |$$

$$X \wedge F_i \neq \emptyset \text{ for } i = 1, ..., L, \text{ and } X \wedge G = \emptyset\} \tag{2}$$

where $\wedge$ denotes the intersection of two sets. In the following, we summarize some of the properties of compact set notation [14, 15, 16]:

1. Multiple fault logic using the compact set notation is as follows: the initial hypothesis set is the set of all subsets of S, i.e., $A = \Theta(1; F_1 = \mathrm{S}\ ;\ G = \emptyset)$. After performing a test, say t_j, the hypothesis set $A = \Theta(L; F_1, ..., F_L; G)$ is decomposed as follows:

$$A \leftarrow \begin{cases} \Theta(L; (F_1 \wedge TS_j^c), ..., (F_L \wedge TS_j^c); (G \vee TS_j)) & \text{if } t_j \text{ passes} \\ \Theta(L+1; F_1, ..., F_L, TS_j \wedge G^c; G) & \text{if } t_j \text{ fails} \end{cases}$$

   where superscript c denotes the set complement, i.e., $G^c = \mathrm{S} - G$.

2. If $E \supseteq F_i$ for some i (that is, E is a superset of F_i), then $\Theta(L+1; F_1, ..., F_L, E; G) = \Theta(L; F_1, ..., F_L; G)$ [6].

3. $A = \Theta(L; F_1, ..., F_L; G) = \Theta(L; F_1 \wedge G^c, ..., F_L \wedge G^c; G)$ [6].

4. Given a set of previously applied passed tests $T_p \subseteq T$ and failed tests $T_f \subseteq T$, the multiple fault ambiguity group at the current stage of diagnosis can be generated directly as follows: $\Theta(L; F_1, ..., F_L; G)$, where $G = \vee_{t_i \in T_p} TS_i$, $L = |T_f| + 1$, $F_1 = \mathrm{S}$ (see the first property), and $F_{i+1} = TS_j \wedge G^c$ for $i = 1, ..., |T_f|$ and $t_j \in T_f$; and then, employ property 2 to remove super sets from the set $F = \{F_1, ..., F_L\}$.

5. If $|T_f| = 0$, then $L = 1$ and $s_0 \in F_1$. If $|T_f| > 0$, none of the F_i's contains s_0 (see the first property).

6. The worst case storage complexity of compact set notation for an OR node is O(mn). This is because the ambiguity group $\Theta(L; F_1, F_2, ..., F_L; G)$ contains all solutions of the following constraint equations:

$$W\underline{y} \geq \underline{e}$$

$$y_i = 0 \quad \text{if } g_i = 1, \text{ for } i = 0, ..., m$$

where $\underline{y} = [y_0, y_1, ..., y_m]'$ is a binary vector; $\underline{e}$ is the L-dimensional vector of 1's; $W = [w_{ij}]$ is a binary matrix of dimension $L \times (m+1)$, and $w_{ij} = 1$ if $s_j \in F_i$, otherwise $w_{ij} = 0$; and $g = [g_0, g_1, ..., g_m]'$ is a binary vector such that $g_j = 1$ if $s_j \in G$, otherwise $g_j = 0$. Using this notation, we need to store the binary matrix W and binary vector g at each stage of diagnosis. Therefore, the storage complexity of this approach is $O(mn)$ at each OR node, since $L \leq n$ and each test is applied at most once in each path of the diagnostic tree.

7. The failure sources belonging to F_i with cardinality $|F_i| = 1$ are definitely faulty (*one-for-sure* condition). This can easily be shown using equation (2).

6.2 Sure Strategies for Multiple Fault Diagnosis

In this section, we present three diagnostic strategies, Sure 1-3, that seek to find definitely failed components, even though there may be others still undiagnosed. Thus, these strategies isolate failures one (or more) at a time, while not making an error when multiple faults are present. The framework for Sure strategies is sketched in Figure 3.

The three basic ingredients of Sure 1-3 are: (i) *minimal candidate generation*, (ii) *minimal candidate isolation*, and(iii) *multiple fault propagation*. The minimality property implies that a particular candidate includes the minimum number of failure sources that explains all test results observed so far (if any). Consequently, the inherent combinatorial explosion that occurs in generating an optimal multiple fault strategy is reduced substantially. Before describing the algorithms, we define minimal (irreducible) set and hitting set of a set of subsets:

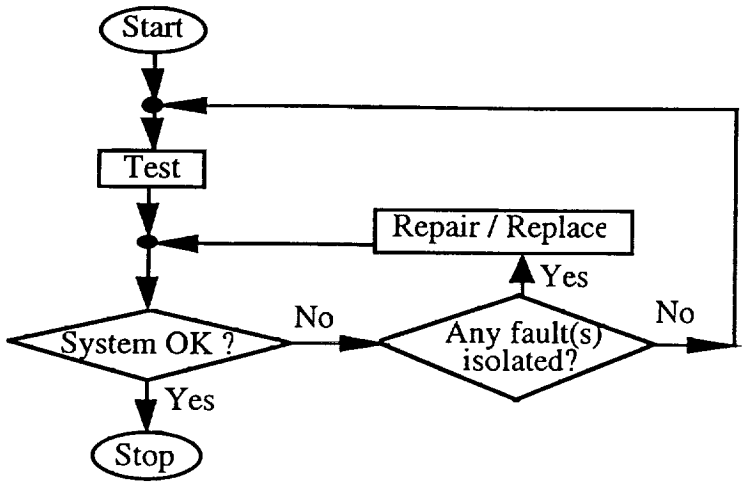

Figure 3: Framework of Sure Strategies in a Test-and-repair Cycle

Definition 1: A minimal or irreducible set for a collection of subsets $Q = \{Q_1, ..., Q_k\}$ is a set $I(Q) \subseteq Q$ such that $I(Q) = Q - \{Q_i | \exists Q_j \in Q \text{ and } Q_j \subseteq Q_i\}$, i.e., $I(Q)$ is equal to set Q without any super set.

Definition 2: A hitting set for a collection of sets $Q = \{Q_1, ..., Q_k\}$ is a set $H(Q) = \{H_1, ..., H_q\}$ such that $H_j \subseteq \vee_{1 \le i \le k} Q_i$ for $j = 1, ..., q$, and $H_j \wedge Q_i \neq \emptyset$ for $i = 1, ..., k$.

Based on these definitions, it can be shown that [12]:

Lemma 1: The minimal set of a multiple fault ambiguity group $A = \Theta(L; F_1, ..., F_L; G)$ is the minimal hitting set for the collection of sets $F = \{F_1, ..., F_L\}$, i.e., $I(A) = I(H(F))$.

Sure1-Sure3 algorithms are recursive procedures. At each iteration, we consider the minimal candidate set of the multiple fault suspect set corresponding to the OR node at that stage. Reiter [12] has derived an algorithm to determine the minimal hitting set of a collection of sets, and Greiner et al. [5] have presented a correction to the Reiter's algorithm. We use this technique to determine the minimal hitting set of $F = \{F_1, ..., F_L\}$ at an OR node. After determining the minimal candidates of a multiple fault suspect set at the current stage, we evaluate the conditional probabilities of minimal candidates using Bayes' rule. Then, we invoke the single fault strategy

TEAMS-S to isolate these candidates, and propagate multiple fault suspect set through the resulting diagnostic tree. Note that using the fourth property of compact set notation, it is sufficient to generate and store multiple fault ambiguity group at the leaf nodes of this tree only. We repeat these procedures for each leaf node of the tree until: (1) the intersection of minimal candidates is not empty, i.e., the corresponding failure sources are definitely faulty, or (2) no test provides further information. The former corresponds to the case when the cardinality of one or more F_i in the ambiguity group is one.

After repairing/replacing the components isolated by Sure strategies, we apply additional tests, if necessary, to isolate the remaining failure sources. We explore three different approaches for the application of additional tests: (1) start from the root OR node of the diagnostic tree; (2) start from the first failed test in the path leading to the isolation of previous faults; (3) update the multiple fault suspect set at the leaf node by integrating previous test results using the fourth property of the compact set notation, removing repaired/replaced failure sources from the ambiguity group at the leaf node, and invoking Sure strategies for the updated ambiguity group. Sure 1-3 algorithms correspond to the first, second and third approaches for applying additional tests, respectively. These are presented in detail in the next subsection.

The Sure1 diagnostic strategy is simple and the resulting diagnostic tree is very similar to the single fault diagnostic tree. However, the expected testing cost using this strategy is usually high. The expected testing cost using Sure2 diagnostic strategy is less than the first one, but the next test to be performed after repairing/replacing each failure source will be different. Furthermore, the diagnostic tree will change to a digraph (directed graph). The expected testing cost for the third approach is the smallest, but the size of the diagnostic tree will be considerably larger than the others. This is because the number of leaves of the diagnostic tree is the same as the number of distinguishable multiple-fault failure signatures. For example, in the worst case, i.e., when the test

matrix B is diagonal, the number of leaves is 2^m. This is because there are 2^m possible multiple-fault failure signatures. But, the number of leaf nodes in Sure1 and Sure2 diagnostic strategies in this case are the same as in a single-fault strategy, i.e., $m + 1$.

One of the interesting features of Sure strategies is that the starting point for all three algorithms is the same tree as in a single fault strategy for the system under consideration. This is because the minimal candidate set for 2^S is $\{s_0, s_1, ..., s_m\}$. Therefore, these strategies isolate a single fault with the smallest average cost, while not making an error when multiple faults are present. Furthermore, in the case of very large systems, instead of generating all minimal candidates, we can generate minimal candidates of size less than a certain threshold, $\hat{L}$, and diagnose multiple faults up to that size.

Example 1.c: Figure 4, without (with) the dashed lines, shows the multiple fault strategy for the system in Example 1.a, based on Sure1(Sure2) algorithm, where A_i denotes the ambiguity group corresponding to the OR node i, and A_1= $\Theta(1; \{s_0, s_1, s_2, s_3, s_4, s_5\}\ ;\emptyset)$; A_2= $\Theta(1; \{s_0, s_2, s_3\}; \{s_1, s_4, s_5\})$; $A_3 = \Theta(1;\{s_1, s_4, s_5\};\emptyset)$; A_4= $\Theta(1;\{s_0\};\{s_1, s_2, s_3, s_4, s_5\})$; $A_5 = \Theta(1;$ $\{s_2,s_3\}; \{s_1, s_4,s_5\})$; $A_6 = \Theta(1; \{s_1, s_4\}; \{s_2, s_3, s_5\})$; $A_7 = \Theta(2; \{s_1, s_4, s_5\}, \{s_2, s_3, s_5\};\emptyset)$; $A_8 =$ $\Theta(1; \{s_2\}; \{s_1, s_3, s_4, s_5\})$; $A_9 = \Theta(1; \{s_3\}; \{s_1, s_4, s_5\})$; $A_{10} = \Theta(1; \{s_1\}; \{s_2, s_3, s_4, s_5\})$; A_{11} $= \Theta(1; \{s_4\}; \{s_2, s_3, s_5\})$; A_{12}= $\Theta(2;\{s_4, s_5\},\{s_2, s_5\};\{s_1, s_3\})$; $A_{13} = \Theta(3; \{s_1, s_4, s_5\}, \{s_2, s_3, s_5\},$ $\{s_1, s_3\}; \emptyset)$; $A_{14} = \Theta(2; \{s_3\}, \{s_1, s_4\}; \{s_2, s_5\})$; $A_{15} = \Theta(3; \{s_1, s_4, s_5\}, \{s_1, s_3\}, \{s_2, s_5\}; \emptyset)$; A_{16} $= \Theta(2; \{s_1\}, \{s_2\}; \{s_3, s_4, s_5\})$; $A_{17} = \Theta(4; \{s_1, s_3\}, \{s_2, s_5\}, \{s_3, s_4, s_5\}, \{s_1, s_4, s_5\}; \emptyset)$

Note that the shaded parts of the tree are the same as those in the single fault diagnostic tree of Figure 1. The average testing cost for the optimal multiple fault strategy is $J = 2.411$, and the average testing cost for the first (Sure1) and second (Sure2) approaches using the diagnostic strategy of Figure 4 are $J = 2.715$ and $J = 2.616$, respectively.

Example 1.d: The Sure3 strategy for Example 1.a is shown in Figure 5, where $A_{18} = A_{20} =$

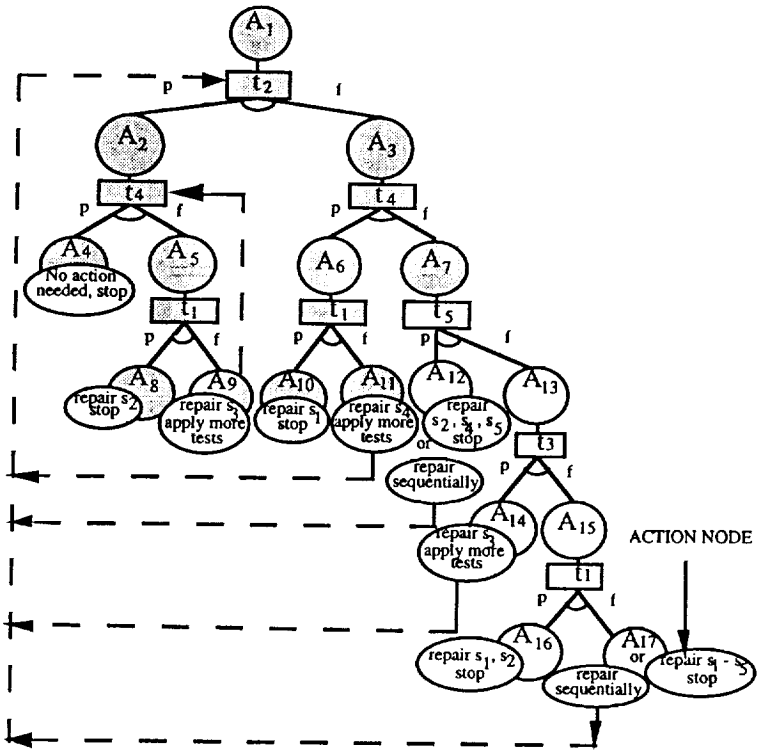

Figure 4: Sure1 and Sure2 Test Strategies for Example 1.a

$A_{24} = \Theta(1; \{s_0\}; \{s_1, s_2, s_3, s_4, s_5\})$; $A_{19} = \Theta(1;\{s_2\};\{s_1, s_3, s_4, s_5\})$; $A_{23} = \Theta(1;\{s_4\}; \{s_2, s_3, s_5\})$;

$A_{21} = A_{22} = A_{25} = \Theta(1;\{s_1\};\{s_2, s_3, s_4, s_5\})$;

Note that the shaded and dashed parts of the tree in Figure 5 are the same as those in Figure 4. For this test strategy, the average test cost $J = 2.535$. In this example, we considered a block replacement strategy when no test gives further information, for example, see ambiguity groups A_{12} and A_{17}.

We applied Sure algorithms to several real-world systems. Table 5 shows the times taken to construct diagnostic strategies based on Sure1 and Sure2 diagnostic strategies for several real-world systems. Table 6 shows the number of nodes in the Sure1 and Sure2 diagnostic strategies for these real-world systems.

We applied Sure3 diagnostic algorithm to several real-world systems. Table 7 shows the times taken to construct a diagnostic strategy based on Sure3 strategy for several real-world systems.

System	(m, n)	Time (sec)			
		# Fault Limits			
		1	2	3	All
Anticollision system	(43, 53)	0.27	1.71	5.98	26.28
Amplifier-filter	(80, 25)	0.18	0.23	0.26	0.27
1553 Bus	(176,53)	0.27	0.50	0.82	1.05
Goodrich (EDIF)	(898, 250)	0.88	0.88	0.88	0.88
Phase Decoder (EDIF)	(1644, 2147)	41.59	461.24	1194.16	(>2400)

Table 5: Solution Times in Seconds Based on Sure1 and Sure2 Strategies for Various Real-world Systems on a SPARC-10

System	(m, n)	# Nodes			
		# Fault Limits			
		1	2	3	All
Anticollision system	(43, 53)	79	521	1889	7257
Amplifier-filter	(80, 25)	55	75	83	89
1553 Bus	(176,53)	67	123	225	289
Goodrich (EDIF)	(898, 250)	37	37	37	37
Phase Decoder (EDIF)	(1644, 2147)	993	8843	21347	out of memory(> 79000)

Table 6: Number of Nodes in Sure1 and Sure2 Strategies for Various Real-world Systems

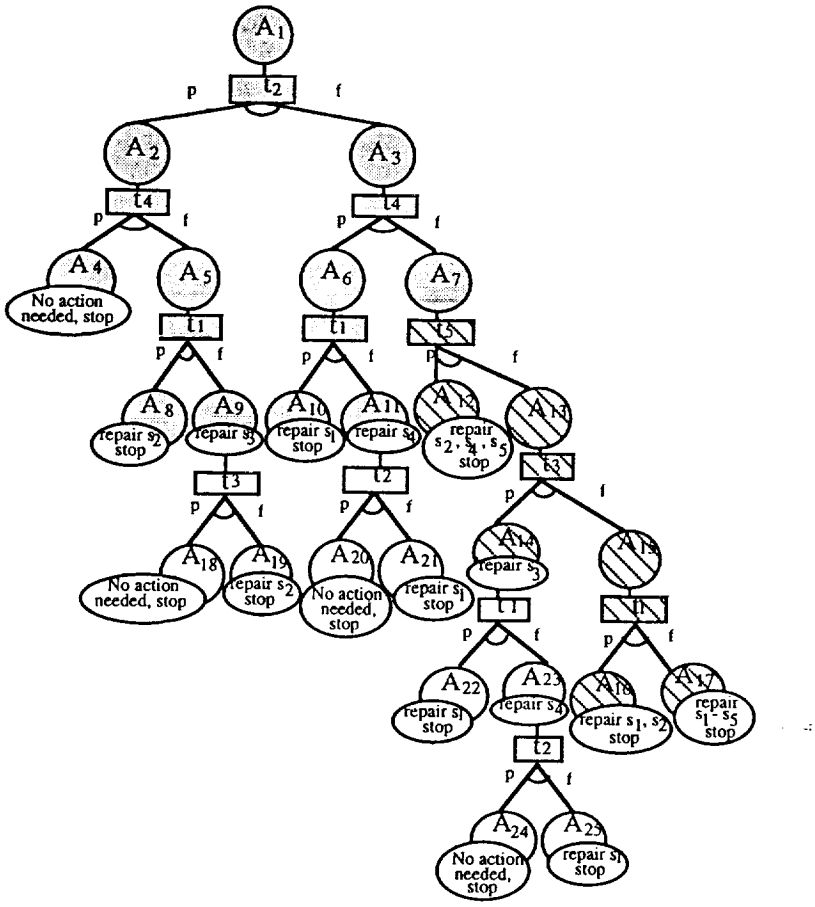

Figure 5: Sure3 Test Strategy for Example 1.a

Table 8 shows the number of nodes in the Sure3 diagnostic strategy of these real-world systems.

The computational results indicate that the size of the diagnostic strategy based on Sure3 is considerably larger than the others, and consequently, Sure3 diagnostic strategy cannot be applied to large-scale systems.

6.2.1 Sure Algorithms

Sure algorithms are recursive functions, and must be invoked as **Sure(** *OR node*, A_m, *surei*), where

- *OR node* is the current OR node,
- $A_m = \Theta(L; F_1, F_2, ..., F_l; G)$ is the multiple fault ambiguity group at the *OR node*,
- *surei* denotes the Sure1-Sure3 diagnostic strategies.

Global variables:

System	(m, n)	Time (sec)			
		# Fault Limits			
		1	2	3	All
Anticollision system	(43, 53)	0.27	6.09	89.74	>7200
Amplifier-filter	(80, 25)	0.18	2.16	6.05	803.27
1553 Bus	(176,53)	0.27	3.76	16.72	>5000
Goodrich (EDIF)	(898, 250)	0.88	4.92	7.72	19.69
Phase Decoder (EDIF)	(1644, 2147)	41.59	> 3600	—	—

Table 7: Solution Times in Seconds Based on Sure3 Strategy for Various Real-world Systems

System	(m, n)	# Nodes			
		# Fault Limits			
		1	2	3	All
Anticollision system	(43, 53)	79	1195	13275	>100000
Amplifier-filter	(80, 25)	55	601	1619	24204
1553 Bus	(176,53)	67	773	3433	>100000
Goodrich (EDIF)	(898, 250)	37	343	553	1463
Phase Decoder (EDIF)	(1644, 2147)	993	>100000	—	—

Table 8: Number of Nodes in Sure3 Strategy for Various Real-world Systems

- the set of failure sources $S = \{s_1, ..., s_m\}$,
- the a prior probability vector $P = [p(s_1), ..., p(s_m)]$,
- the set of available tests $T = \{t_1, t_2, ..., t_n\}$,
- the set of test costs $C = \{c_1, c_2, ..., c_n\}$,
- the binary test matrix $B = [b_{ij}]$.

Initialization:

- *OR node*=root OR node,
- $A_m = \Theta(1; F_1 = \mathrm{S}; G = \emptyset)$.

Algorithm: Sure(*OR node*, A_m, *surei*)

step 1: Generate the minimal (or irreducible) set of the multiple fault ambiguity group A_m, $A_s = I(A_m)$.

step 2: Evaluate the conditional probability of faults in A_s using P, P_s.

step 3: Generate the binary test matrix D_s using B for the faults in A_s; failure signature of each fault in A_s is the union of failure signatures of individual failures.

step 4: IF no test gives any information, THEN
step 4.1: Action: repair/replace all faults in $\cup_{1 \le i \le L} F_i - \{s_0\}$.
step 4.2: $G \leftarrow G \cup$ {repaired/replaced failure sources}.
step 4.3: $SS \leftarrow \mathrm{S} - G$.
step 4.4: IF $SS = \{s_0\}$, THEN
- Action: stop.
- label the *OR node* SOLVED, and RETURN.

END
step 4.5: IF *surei* is Sure1, THEN
- Action: apply more tests from root OR node.
- label the *OR node* SOLVED, and RETURN.

ELSE IF *surei* is Sure2, THEN
- Action: apply more tests from the first failed test on the path from root OR node to the *OR node*.
- label the *OR node* SOLVED, and RETURN.

ELSE IF *surei* is Sure3, THEN
- $A_m = \Theta(1; F_1 = SS; G)$.
- Invoke **Sure(** *OR node*, A_m, *surei*).

END

END

step 5: Expand the diagnostic tree from the *OR node* by invoking TEAMS-S (A_s, P_s, T, C, D_s).

step 6: Propagate the multiple fault ambiguity group A_m of the *OR node* along the tree.

step 7: DO for each UNSOLVED leaf node,

step 7.1: IF the multiple fault ambiguity group of the leaf node has guaranteed failures identified (i.e., F_i (s) with one member),
THEN

step 7.1.1: IF any $F_i = \{s_0\}$, THEN
- Action: stop.
- label the *OR node* SOLVED, and CONTINUE.

END

step 7.1.2: Action: repair/replace the faults in F_i(s) with one member.

step 7.1.3: $G \leftarrow G \cup$ {repaired failure sources}.

step 7.1.4: $SS \leftarrow S - G$.

step 7.1.5: IF $SS = \{s_0\}$, THEN
- Action: stop.
- label the *OR node* SOLVED, and CONTINUE.

- END

step 7.1.6: IF *surei* is Sure1, THEN
- Action: apply more tests from root OR node.
- label the *OR node* SOLVED, and CONTINUE.

ELSE IF *surei* is Sure2, THEN
- Action: apply more tests from the first failed test on the path from the root OR node to the *OR node*.
- label the *OR node* SOLVED, and CONTINUE.

ELSE IF *surei* is Sure3, THEN
- $A_m = \Theta(1; F_1 = SS; G)$.
- Invoke **Sure**(leaf OR node, A_m, *surei*).

END

ELSE

step 7.1.7: $A_m \leftarrow$ Multiple fault ambiguity groups of the leaf node.

step 7.1.8: Invoke **Surei**(leaf OR node, A_m, *surei*).

END

END

6.3 Computational Issues

In order to make the algorithm efficient, we find all the failure sources with the same failure signature in the test matrix. That is, we generate the set N=$\{N_1, N_2,...,N_\beta\}$ such that $N_l \subseteq S$ for $l = 1,...,\beta$ and $\forall s_i \in N_l$ have the same failure signatures in the binary test matrix. Thus, instead of invoking Sure strategies for the set S, we can invoke them for the set N. In this case, the probability that none of $s_i \in N_l$ is faulty, i.e., $\hat{p}(N_l)$, and only one of $s_i \in N_l$ is faulty, i.e., $p(N_l)$, can be evaluated as follows:

$$\hat{p}(N_l) = \prod_{s_i \in N_l} (1 - p(s_i)) \tag{3}$$
$$p(N_l) = \sum_{s_i \in N_l} p(s_i) \prod_{s_j \in N_l \& i \neq j} (1 - p(s_j))$$

Thus, using $p(N_l)$ and $\hat{p}(N_l)$, the conditional probabilities of minimal candidates can be evaluated. For example, the conditional probabilities associated with the set N at the starting point, i.e., based on a single fault assumption, can be evaluated as follows:

$$p_0 = \frac{1}{1 + \sum_{k=1}^{\beta} \frac{p(N_k)}{\hat{p}(N_k)}} \tag{4}$$
$$p_{N_l} = \frac{\frac{p(N_l)}{\hat{p}(N_l)}}{1 + \sum_{k=1}^{\beta} \frac{p(N_k)}{\hat{p}(N_k)}} \quad \text{for} \quad l = 1,...,\beta$$

In the case, when $|N_l| = 1$ for $l = 1,...,\beta$ and $\beta = m$, (4) reduces to (1).

7 Summary

The computational and storage complexity of an optimal multiple fault strategy are super-exponential in the number of failure sources, m. We presented several near-optimal algorithms that provide a trade-off between optimality and computational complexity. Firstly, we extended the single-fault strategy of our previous work [7, 8, 11] to diagnose multiple faults by successively isolating the potential single-fault candidates, then double-fault candidates, and so on. This is one

of the simplest multiple fault strategies that one can use. In this approach, the storage complexity at each OR node of the AND/OR graph is the same as that in a single fault strategy. However, using this approach, the probability of false alarm error or RTOK is very high.

We then extended the single fault sequential testing strategies to a class of Sure strategies. The basic idea of these strategies is to find one or more definitely failed components, while not making an error when other co-existing faults are present. We explored three different approaches for the application of additional tests, resulting in Sure1-3 strategies.

Some of the advantages of using Sure strategies are: (1) the inherent combinatorial explosion that occurs in generating an optimal multiple fault strategy is reduced substantially. (2) the first iteration of the Sure strategies results in the same tree as in the single fault strategy, and therefore, these strategies isolate a single fault with the smallest average cost, while not making an error when multiple faults are present. Computational complexity of this approach is strictly related to the structure of the system, i.e., the structure of test matrix B.

In order to overcome the problems associated with the size of the complete diagnostic strategy, the test strategy should be generated "on-line". That is, instead of generating the entire diagnostic tree, the interactive strategy only suggests the next test to be applied given the outcomes of previously applied tests. In addition, we assumed that the failure signature of each multiple failure is the union of the failure signatures of individual failures. However, this assumption does not hold for fault-tolerant (redundant) systems. In order to solve this problem, a binary test matrix based on minimal candidates, i.e., minimum number of failures with a failure signature different from the union of failure signatures of individual failures, should be generated. We expect to investigate these challenging issues in our future efforts [17, 18].

Appendix A: Conditional Probabilities of Failure Sources

Let us assume that hypothesis $S_I \subseteq S = \{s_1, ..., s_m\}$ denotes a set of failure sources such that $\{s_i \in S_I\}$ are faulty and $\{s_j \in S_I\}$ are fault-free. Thus, S_I can be represented as an m-dimensional hypothesis vector $x = [x_1, ..., x_m]'$ where $x_i = 1$ if $s_i \in S_I$; otherwise, $x_i = 0$. From the failure independence assumption, the probability of hypothesis vector $\underline{x}$ is;

$$\begin{aligned} p(\underline{x}) &= \prod_{i=1}^{m} p(s_i)^{x_i}(1 - p(s_i))^{1-x_i} \\ &= \prod_{i=1}^{m} (\frac{p(s_i)}{1 - p(s_i)})^{x_i}(1 - p(s_i)) \\ &= p(\underline{x} = \underline{0}) \prod_{i=1}^{m} (\frac{p(s_i)}{1 - p(s_i)})^{x_i} \end{aligned} \tag{5}$$

where $\underline{0}$ is a zero vector of dimension m, and $p(\underline{x} = \underline{0})$ is the probability of fault-free state of the system. Using Bayes' rule, the conditional probability of failure hypothesis $\underline{x}$ based on a single-fault assumption, i.e., $\underline{P} = \{p_0, p_1, ..., p_m\}$, is as follows:

$$p(\underline{x}|SF) = \begin{cases} \frac{\frac{p(s_i)}{(1-p(s_i))}}{1+\sum_{k=1}^{m} \frac{p(s_k)}{(1-p(s_k))}} & \text{if } x_i = 1 \text{ and } x_j = 0 \ \forall \ j \neq i \\ \frac{1}{1+\sum_{k=1}^{m} \frac{p(s_k)}{(1-p(s_k))}} & \text{if } \underline{x}=\underline{0} \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

Thus, the conditional probability of failure source s_i given the single fault assumption, p_i, is the conditional probability of hypothesis vector $\underline{x} = \underline{e_i}$, where $\underline{e_i}$ is the i-th unit vector, i.e., $x_i = 1$ and $x_j = 0 \ \forall \ j \neq i$, and $p_0 = p(\underline{x} = 0|SF)$.

Note that ***a priori*** probability of failure source s_i, i.e., $p(s_i)$, can be derived from the distribution function $F_i(t)$ as $p(s_i) = F_i(t_0)$, where $F_i(t)$ is the probability that failure source s_i has failed at or before time t, and t_0 is the UPTIME. In the following, we consider two special cases: (1) Exponential distribution, and (2) Weibull distribution.

A.1: Exponential Distribution

In this case, $F_i(t) = (1 - e^{-\lambda_i t})$ for $i = 1, ..., m$, where $\lambda_i = 1/\text{MTTF}_i$ is the failure rate and MTTF_i is the mean time between failures. Thus,

$$p(s_i) = (1 - e^{-\lambda_i t_0}) \quad \text{for} \quad i = 1, ..., m \tag{7}$$
$$p(s_0) = \prod_{i=1}^{m} (1 - p(s_i)) = e^{-\lambda_T t_0}$$

where $\lambda_T = \sum_{i=1}^{m} \lambda_i$. The conditional probability of each failure source s_i, using equation (6), is as follows:

$$p_i = \frac{e^{\lambda_i t_0} - 1}{1 + \sum_{j=1}^{m} (e^{\lambda_j t_0} - 1)} = \frac{e^{\lambda_i t_0} - 1}{1 - m + \sum_{j=1}^{m} e^{\lambda_j t_0}} \quad \text{for} \quad i = 1, ..., m \tag{8}$$
$$p_0 = \frac{1}{1 - m + \sum_{j=1}^{m} e^{\lambda_j t_0}}$$

if $\lambda_i t_0 << 1$ for $i = 1, ..., m$, then $e^{\lambda_i t_0} \approx 1 + \lambda_i t_0$. Thus, equation (8) reduces to:

$$p_i \approx \frac{\lambda_i t_0}{1 + \lambda_T t_0} = \frac{\lambda_i}{\frac{1}{t_0} + \lambda_T} = \frac{\lambda_i}{\lambda_T}(1 - p_0) \quad \text{for} \quad i = 1, ..., m$$
$$p_0 \approx \frac{1}{1 + \lambda_T t_0} = \frac{\frac{1}{t_0}}{\frac{1}{t_0} + \lambda_T}$$

A.2: Weibull Distribution

In this case, $F_i(t) = (1 - e^{-(\lambda_i t)^\alpha})$ where $\frac{1}{\lambda_i}$ is the characteristic life and α is a shape parameter that changes the shape of the distribution compared with the exponential. Thus,

$$p(s_i) = 1 - e^{-(\lambda_i t)^\alpha} \quad \text{for} \quad i = 1, ..., m \tag{9}$$
$$p_0 = \prod_{i=1}^{m} (1 - p(s_i)) = e^{-(\lambda_w t_0)^\alpha}$$

where $\lambda_w = (\sum_{i=1}^{m} \lambda_i^{\alpha})^{\frac{1}{\alpha}}$. Therefore, the conditional probability of each failure source s_i is as follows:

$$p_i = \frac{e^{(\lambda_i t_0)^{\alpha}} - 1}{1 + \sum_{j=1}^{m}(e^{(\lambda_i t_0)^{\alpha}} - 1)} = \frac{e^{(\lambda_i t_0)^{\alpha}} - 1}{1 - m + \sum_{j=1}^{m} e^{(\lambda_i t_0)^{\alpha}}} \quad \text{for} \quad i = 1, ..., m \tag{10}$$

$$p_0 = \frac{1}{1 - m + \sum_{j=1}^{m} e^{(\lambda_i t_0)^{\alpha}}}$$

If $\lambda_i t_0 << 1$ for $i = 1, ..., m$, then $e^{(\lambda_i t_0)^{\alpha}} \approx 1 + (\lambda_i t_0)^{\alpha}$. Thus, equation (10) reduces to:

$$p_i \approx \frac{(\lambda_i t_0)^{\alpha}}{1 + \sum_{j=1}^{m}(\lambda_j t_0)^{\alpha}} = \frac{{\lambda_i}^{\alpha}}{\frac{1}{{t_0}^{\alpha}} + {\lambda_w}^{\alpha}} = \frac{{\lambda_i}^{\alpha}}{{\lambda_w}^{\alpha}}(1 - p_0) \quad \text{for} \quad i = 1, ..., m$$

$$p_0 \approx \frac{1}{1 + (\lambda_w t_0)^{\alpha}} = \frac{\frac{1}{t_0^{\alpha}}}{\frac{1}{t_0^{\alpha}} + {\lambda_w}^{\alpha}} \tag{11}$$

References

[1] D.P. Bertsekas. *Dynamic Programming: Deterministic and Stochastic Models.* Prentice-Hall, Englewood Cliffs, NJ, 1987.

[2] R. Davis. Diagnostic reasoning based on structure and behavior. *Artificial Intelligence*, 24:347–410, 1984.

[3] R. Davis. Retrospective on diagnostic reasoning based on structure and behavior. *Artificial Intelligence*, 59:149–157, 1993.

[4] J. de Kleer and B.C. Williams. Diagnosing multiple faults. *Artificial Intelligence*, 32:97–130, 1987.

[5] R. Greiner, B.A. Smith, and R.W. Wilkerson. A correction to the algorithm in Reiter's theory of diagnosis. *Artificial Intelligence*, 41:79–88, 1989/90.

[6] D.B. Grunberg, J.L. Weiss, and J.C. Deckert. Generation of optimal and suboptimal strategies for multiple fault isolation. *Technical report TM-248*, 1987.

[7] K.R. Pattipati and M.G. Alexandridis. Application of heuristic search and information theory to sequential fault diagnosis. *IEEE Transactions on Systems, Man, and Cybernetics*, 20(4):872–887, July/ August 1990.

[8] K.R. Pattipati, S. Deb, M. Dontamsetty, and A. Maitra. Start: System testability analysis and research tool. *IEEE AES Magazine*, pages 13–20, January 1991.

[9] K.R. Pattipati, V. Raghavan, M. Shakeri, S. Deb, and R. Shrestha. TEAMS: Testability Engineering And Maintenance System. *American Control Conference*, pages 1989–1996, June 1994.

[10] V. Raghavan. *Algorithms for Sequential Fault Diagnosis.* Ph.D Thesis, Dept. of Electrical and Systems Engineering, University of Connecticut, Storrs, CT 06269-3157, 1996.

[11] V. Raghavan, M. Shakeri, and K.R. Pattipati. Optimal and near-optimal test sequencing algorithms with realistic test models. *Submitted to IEEE Transactions on Systems, Man, and Cybernatics.*

[12] R. Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32(1):57–95, Apr. 1987.

[13] M. Shakeri. *Advances in System Fault Modeling and Diagnosis.* Ph.D Thesis, Dept. of Electrical and Systems Engineering, University of Connecticut, 1996.

[14] M. Shakeri, K.R. Pattipati, V. Raghavan, , and S. Deb. Near-optimal sequential testing algorithms for multiple fault isolation. *IEEE International Conference on Systems, Man and Cybernetics*, pages 1908–1914, 1994.

[15] M. Shakeri, K.R. Pattipati, V. Raghavan, and A. Patterson-Hine. Multiple fault isolation in redundant systems. *IEEE International Conference on Systems, Man and Cybernetics*, 1995.

[16] M. Shakeri, K.R. Pattipati, V. Raghavan, A. Patterson-Hine, and T. Kell. Sequential test strategies for multiple fault isolation. *IEEE Autotestcon*, 1995.

[17] M. Shakeri, V. Raghavan, and K.R. Pattipati. Multiple fault isolation in redundant systems. *In Preparation.*

[18] M. Shakeri, V. Raghavan, and K.R. Pattipati. On-line multiple fault diagnosis strategies. *In Preparation.*

[19] J.H. Sheppard and W.R. Simpson. Multiple failure diagnosis. *IEEE Autotestcon*, pages 381–389, 1994.

1996 IEEE Autotest Conference, Dayton, Ohio, September 1996

Optimal and Near-Optimal Algorithms for Multiple Fault Diagnosis with Unreliable Tests*

M. Shakeri and V. Raghavan
Qualtech Systems Inc.,
119 Storrs Rd, Mansfield, CT 06250

K.R. Pattipati
U-157, Dept. of Electrical and Systems Engineering,
University of Connecticut, Storrs, CT 06269-3157
E-mail: krishna@sol.uconn.edu

A. Patterson-Hine
NASA-Ames Research Center, Mail Stop 269-4,
Moffett Field, CA 94035-1000
E-mail: Ann_Patterson-Hine@styx.arc.nasa.gov

Abstract - In this paper, we consider the problem of constructing optimal and near-optimal multiple fault diagnosis (MFD) in bipartite systems with unreliable (imperfect) tests. It is known that exact computation of conditional probabilities for multiple fault diagnosis is NP-hard. The novel feature of our diagnostic algorithms is the use of Lagrangian relaxation and subgradient optimization methods to provide: (1) near optimal solutions for the MFD problem, and (2) upper bounds for an optimal branch-and-bound algorithm. The proposed method is illustrated using several examples. Computational results indicate that: (1) our algorithm has superior computational performance to the existing algorithms (approximately three orders of magnitude improvement over the algorithm in [3]), (2) the near optimal algorithm generates the most likely candidates with a very high accuracy, and (3) our algorithm can find the most likely candidates in systems with as many as 1000 faults.

I. Introduction

With the increased recognition of importance of design for testability, there is an increasing trend towards the use of smart sensors for on-board system health management. The results of on-board tests are available to the ground test systems and operators as a *block* of symptoms. Due to improper set up, operator error, electromagnetic interference, environmental conditions, or aliasing inherent in the signature analysis of on-board tests, the nature of tests may be unreliable (imperfect). Imperfect tests introduce an additional element of uncertainty into the diagnostic process: the pass outcome of a test does not guarantee the integrity of components under test (because the test may have missed a fault), or a failed test outcome does not mean that one or more of the implicated components are faulty (because the test outcome may have been a false alarm). Consequently, the diagnostic procedures must hedge against this uncertainty in test outcomes.

In this paper, we consider the problem of constructing optimal and near-optimal multiple fault diagnosis in bipartite digraphs with unreliable tests. This problem is a central and long-standing concern in system fault diagnosis, and medical decision making [10]. When the false alarm probabilities of all tests are zero, the problem simplifies to the parsimonious covering theory (or probabilistic causal model) discussed in [16]. Peng and Reggia [15] proposed a competition based connectionist method to subdue the problem of combinatorial explosion in computing the posterior probabilities of all possible combinations of failure sources in probabilistic causal models. However, this method does not guarantee a global optimum and suffers from large computation times even for problems with small numbers of failure sources, m=26.

Genetic algorithms are offered as an alternative to the connectionist methods [3, 9]. Genetic algorithms are based on an analogy with Darwin's biological evolutionary theory in which a group of solutions evolves via natural selection. It emulates the rules of biological evolution-

*Research supported by NASA-Ames Research Center and the Department of Economic Development of the State of Connecticut.

ary process, such as reproduction, crossover, mutation, and natural selection, etc. At each iteration, a population of individuals is established, where each individual corresponds to a point in the search space. The objective function is evaluated for each individual to rate its fitness. Then, a next generation is formed based on the survival of the fittest. Therefore, the evolution of individuals from generation to generation tends to result in fitter individuals (i.e., solutions) in the search space. These algorithms converge extremely slowly, and have been applied to small problems with m=20 failure sources (causes, disorders) and n=20 tests (manifestations, symptoms).

Wu [20] proposed a decomposition method based on common and disjoint causal (failure source) relationships among the given symptoms (tests). This method decomposes the original problem into smaller and independent subproblems, and therefore, increases the performance and efficiency of multiple fault diagnosis. However, this approach is not applicable for systems with large numbers of nondecomposable causes and symptoms.

In this paper, we present a novel approach, using Lagrangian relaxation, to solve multiple fault diagnosis problem. By defining new variables and constraints, the multiple fault diagnosis (MFD) problem reduces to a combinatorial optimization problem with a set of equality constraints. The constraints are relaxed via Lagrange multipliers. The relaxation procedure generates an upper bound for the objective function. The procedure of minimizing the upper bound via a subgradient optimization produces a sequence of solutions that are modified, in a computationally effective way, to produce a sequence of feasible solutions to the MFD problem. If the objective function value for the best feasible solution and the upper bound are the same, the feasible solution is the optimal solution. Otherwise, the difference between the upper bound and the feasible solution, termed the approximate duality gap, provides a measure of suboptimality of the MFD solution. Alternatively, the optimal solution can be found via a tree search (or branch-and-bound) procedure. The computational complexity of the near-optimal algorithm is a linear function of the number of failure sources, m and the number of failed tests, $|T_f|$.

Next, we present an approach to determine a ranked set of multiple fault diagnosis solutions (i.e., the best, second best, ..., L-th best diagnosis). In this approach, following Murty [12] and Cox et. al. [5], we: (1) partition the MFD problem, based on its best solution, into disjoint subproblems; (2) solve the subproblems and sort them by the values of their solutions, and (3) select the subsequent best solutions. One of the advantages of this approach, compared to the one in [14], is that since the subproblems are disjoint, the optimal solution of each subproblem is different from the others. Finally, we show that the MFD algorithm can be extended to solve multiple fault diagnosis problems with repetitive application of tests.

The paper is organized as follows. In Section II, we formulate the multiple fault diagnosis problem in a bipartite system. In Section III, we present a near-optimal algorithm based on Lagrangian relaxation and subgradient optimization method to diagnose multiple faults, and generate an upper bound for the likelihood of multiple fault candidates. The upper bound can be used in an optimal branch-and-bound algorithm. The multiple fault algorithms for a set of L-ranked multiple fault diagnoses are presented in Section IV. In Section V, we consider the multiple fault diagnosis problem with repetitive tests. Several examples are presented in Section VI. Finally, in Section VII, we summarize the results and discuss future research issues.

II. Problem Formulation

The MFD problem in bipartite systems with imperfect tests consists of a bipartite digraph $DG = \{S, T, E\}$, where

- $S = \{s_1, ..., s_m\}$ is a finite set of *independent* failure sources (failure nodes) associated with the system;
- $T = \{t_1, t_2, ..., t_n\}$ is a finite set of n available binary outcome tests (test node), where the integrity of system failure sources/components/modules can be ascertained;
- $E = \{e_{ij}\}$ is the set of digraph edges (links) specifying the functional information flow between the set of failure sources and the set of tests in the system.

The input requirements of the failure nodes and edges of the digraph are as follows:

1. Failure node: ***A priori*** **probability vector of failure nodes** $P = [p(s_1), ..., p(s_m)]$, **where** $p(s_i) > 0$ **is the a priori probability of failure source** s_i.
2. Link (edge): A set of probability pairs $P_{ij} = \overline{(Pd_{ij}, Pf_{ij})}$ representing the detection-false-alarm probabilities of the set of tests, where Pd_{ij} and Pf_{ij} are the detection and false alarm probabilities of test t_j and failure source s_i, respectively (see Figure 1). Figure 2 shows a bipartite digraph model.

The problem is to find the most likely candidates $X \subseteq S$ that are consistent with the results of applied tests. This is formulated as:

$$\max_{X \subseteq S} \quad \text{Prob}(X|T_p, T_f) \tag{1}$$

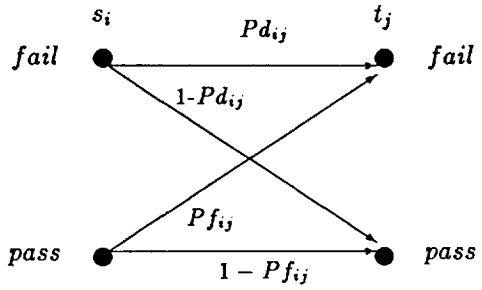

Figure 1: Detection-False-Alarm Probability of Failure Source s_i and Test t_j

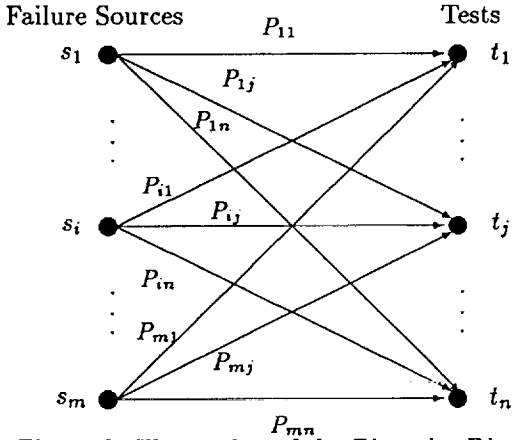

Figure 2: Illustration of the Bipartite Digraph Model

where $T_p \subseteq T$ and $T_f \subseteq T$ denote the set of passed and failed tests, respectively. Using Bayes' rule and eliminating the constant factor $\text{Prob}(T_p, T_f)$, we obtain the following equivalent maximization problem:

$$\max_{X \subseteq S} \quad \text{Prob}(T_f, T_p|X)\text{Prob}(X) \tag{2}$$

For notational simplicity, we define binary vector $\underline{x}$ of size m, where $x_i = 1$ if failure source $s_i \in X$; $x_i = 0$, otherwise. **Note that, given a multiple fault candidate X, the tests are independent. Thus, the above probabilities** can be evaluated as follows:

$$\text{Prob}(T_p|X) = \prod_{t_k \in T_p} \text{Prob}(O(t_k) = p|X) \tag{3}$$

$$\text{Prob}(T_f|X) = \prod_{t_j \in T_f} \text{Prob}(O(t_j) = f|X) \tag{4}$$

$$\begin{aligned} &\text{Prob}(O(t_j) = p|X) \\ &= \prod_{i=1}^{m}(1 - Pd_{ij})^{x_i}(1 - Pf_{ij})^{(1-x_i)} \\ &= [\prod_{i=1}^{m}(1 - Pf_{ij})][\prod_{i=1}^{m}(\frac{1 - Pd_{ij}}{1 - Pf_{ij}})^{x_i}] \end{aligned} \tag{5}$$

$$\text{Prob}(O(t_j) = f|X) = 1 - \text{Prob}(O(t_j) = p|X) \tag{6}$$

$$\begin{aligned} \text{Prob}(X) &= \prod_{i=1}^{m} p(s_i)^{x_i}(1 - p(s_i))^{(1-x_i)} \\ &= [\prod_{i=1}^{m}(1 - p(s_i))][\prod_{i=1}^{m}(\frac{p(s_i)}{1 - p(s_i)})^{x_i}] \end{aligned} \tag{7}$$

where $O(t_j) \in \{p(=\text{pass}), f(=\text{fail})\}$ is the outcome of test t_j, and $Pd_{ij} = 0$ and $Pf_{ij} = 0$ for $e_{ij} \notin E$.

III. Problem Solution

One approach for generating the optimal multiple fault diagnosis is to consider all possible combinations of failure sources, i.e., the power set 2^S, and select the multiple fault candidate with the highest likelihood function in (1). However, the computational complexity of this approach is exponential in the number of failure sources m. In the following, we present an algorithm, based on Lagrangian relaxation and subgradient optimization method, to generate a near-optimal solution for this problem.

By substituting (3) and (4) into (2) and taking the natural logarithm of the resulting objective function, the problem is equivalent to:

$$\begin{aligned} \max_{X \subseteq S} \quad & \textstyle\sum_{t_j \in T_f} \ln(\text{Prob}(O(t_j) = f|X)) + \\ & \textstyle\sum_{t_k \in T_p} \ln(\text{Prob}(O(t_k) = p|X)) + \\ & \ln(\text{Prob}(X)) \end{aligned} \tag{8}$$

By substituting (5), (6) and (7) into (8), the problem reduces to:

$$\begin{aligned} \max_{X \subseteq S} \quad & \textstyle\sum_{t_j \in T_f} \ln(1 - [\prod_{i=1}^{m}(\frac{\overline{Pd}_{ij}}{\overline{Pf}_{ij}})^{x_i}][\prod_{i=1}^{m}(\overline{Pf}_{ij})]) + \\ & \textstyle\sum_{t_k \in T_p}\{\sum_{i=1}^{m} x_i \ln(\frac{\overline{Pd}_{ik}}{\overline{Pf}_{ik}}) + \sum_{i=1}^{m} \ln(\overline{Pf}_{ik})\} + \\ & \textstyle\sum_{i=1}^{m}\{x_i \ln(p_i) + \ln(1 - p(s_i))\} \end{aligned} \tag{9}$$

where $p_i = \frac{p(s_i)}{(1-p(s_i))}$, $\overline{Pf}_{ij} = 1 - Pf_{ij}$ and $\overline{Pd}_{ij} = 1 - Pd_{ij}$ for $i = 1, ..., m$ and $j = 1, ..., m$. By (i) eliminating constant factors $\sum_{i=1}^{m} \ln(1 - p(s_i))$ and $\sum_{t_k \in T_p} \sum_{i=1}^{m} \ln(\overline{Pf}_{ik})$, and (ii) defining new variables $y_j = [\prod_{i=1}^{m}(\frac{\overline{Pd}_{ij}}{\overline{Pf}_{ij}})^{x_i}][\prod_{i=1}^{m}(\overline{Pf}_{ij})]$ for $t_j \in T_f$, and taking the natural logarithm of it, the problem reduces to the following optimization problem:

$$\begin{aligned} \max_{\underline{x}, \underline{y}} \quad & J(\underline{x}, \underline{y}) = \textstyle\sum_{t_j \in T_f} \ln(1 - y_j) + \\ & \textstyle\sum_{i=1}^{m} x_i\{\sum_{t_k \in T_p} \ln(\frac{\overline{Pd}_{ik}}{\overline{Pf}_{ik}}) + \ln(p_i)\} \end{aligned} \tag{10}$$

$$\text{subject to}: \ln(y_j) = \sum_{i=1}^{m} x_i \ln(\frac{\overline{Pd_{ij}}}{\overline{Pf_{ij}}}) + \sum_{i=1}^{m} \ln(\overline{Pf_{ij}}) \quad (11)$$

$$0 \le y_j \le 1 \text{ for } t_j \in T_f \quad (12)$$

$$x_i = 0 \text{ or } 1 \quad \text{for } i = 1, ..., m \quad (13)$$

where $\underline{y} = [y_1, ..., y_{|T_f|}]$, and $|.|$ denotes set cardinality. For simplicity, we define new variables $h_j = \sum_{i=1}^{m} \ln(\overline{Pf_{ij}})$ for $t_j \in T_f$. Note that, if we define $\overline{Pc_{ij}} = (\frac{\overline{Pd_{ij}}}{\overline{Pf_{ij}}})$ for $i = 1, ..., m$ and $j = 1, ..., n$, then $\overline{Pc_{ij}}$, h_j and p_i are sufficient statistics for solving this problem. The following lemmas present two important properties of the MFD problem.

Lemma 1: If Pf_{ik}=0 and $Pd_{ik} = 1$ for any passed test t_k, then the optimal solution does not contain failure source s_i, i.e., x_i=0 (or equivalently $s_i \notin X$).

Proof: If $s_i \in X$ (or $x_i = 1$), then the second part of the objective function in (10), and, consequently, the overall objective function will be unbounded, i.e., it would be -∞.

Using Lemma 1, the size of the MFD problem can be reduced by removing all failure sources $\{s_i | Pf_{ik}=0, Pd_{ik} = 1$ and $t_k \in T_p\}$ from the problem.

Lemma 2: If the false alarm probabilities of a failed test t_j are zero, i.e., Pf_{ij}=0 for $i = 1, ..., m$, then the optimal solution contains at least one x_i=1, such that $Pd_{ij} > 0$. That is, the optimal solution must cover the failed tests.

Proof: We prove this lemma by contradiction. Pf_{ij}=0 for $i = 1, ..., m$ results in h_j=0. If for all $Pd_{ij} > 0$, x_i=0, then we have $\ln(y_j)$=0 and, hence, $y_j = 1$. Thus, $\ln(1 - y_j)$, the first part of the objective function in (10), and, consequently, the overall objective function will be unbounded.

Using Lemma 2, we define the following constraints:

$$A\underline{x} \ge \underline{e} \qquad \text{for } t_j \in T_f \text{ and } h_j = 0 \quad (14)$$

where $A = \{a_{lj}\}$ is a binary matrix of size $|H| \times m$; $H = \{t_j \in T_f | h_j = 0 \text{ for } j = 1, ..., n\}$; each row l of matrix A corresponds to a failed test t_j with $h_j = 0$; a_{li}=1, if $Pd_{ij} > 0$ for $i = 1, ..., m$; otherwise, a_{li}=0, and $\underline{e}$ is a vector of 1's.

Adding the set of constraints (14) to the problem in (10)-(13) results in a smaller search space and tighter upper and lower bounds (best feasible solution found), and, therefore, a better estimate of the optimal solution.

Lemma 3: When all tests are perfect, that is, $Pd_{ij} = 1$ and $Pf_{ij} = 0$ for $i = 1, ..., m$, $j = 1, ..., n$ and $e_{ij} \in E$, using Lemmas 1 and 2, the problem reduces to the following set-covering problem: $\max_{\underline{x}} \sum_{s_i \in S^-} p_i x_i$ subject to (13) and (14), where S^- is the reduced set of failure sources, i.e., S after eliminating the failure sources satisfying Lemma 1.

Proof: This lemma can easily be proved by Lemmas 1 and 2. $Pd_{ij} = 1$ and $Pf_{ij} = 0$ for the failed tests results in $h_j = y_j = 0$. Therefore, the first part as well as the second part (using Lemma 1) of the objective function in (10) can be eliminated, and the problem reduces to the traditional set covering problem. The set covering problem can be solved optimally by any optimal set-covering algorithm [2, 7], or near-optimally via a Lagrangian relaxation and subgradient optimization method [1].

By relaxing the constraints in (11) via Lagrange multipliers $\{\lambda_j\}$, we obtain the Lagrangian function:

$$\max_{\underline{x},\underline{y}} Q(\underline{\lambda}, \underline{x}, \underline{y}) = \sum_{t_j \in T_f} \{\ln(1 - y_j) + \lambda_j \ln(y_j)\} + \sum_{i=1}^{m} x_i \{ \sum_{t_k \in T_p} \ln(\frac{\overline{Pd_{ik}}}{\overline{Pf_{ik}}}) + \ln(p_i) - \sum_{t_j \in T_f} \lambda_j \ln(\frac{\overline{Pd_{ij}}}{\overline{Pf_{ij}}}) \} - \sum_{t_j \in T_f} \lambda_j h_j \quad (15)$$

$$= \sum_{t_j \in T_f} f_j(\lambda_j, y_j) + \sum_{i=1}^{m} c_i(\underline{\lambda}) x_i - \sum_{t_j \in T_f} \lambda_j h_j$$

subject to (12), (13) and (14), where $f_j(\lambda_j, y_j)$ and $c_i(\underline{\lambda})$ denote the first and second equations in the brackets in (15), respectively. *The important point here is that the maximization of Lagrangian function in (15) with respect to $\underline{x}$ and $\underline{y}$ can be carried out independently for each fixed $\underline{\lambda}$.* Maximization of $Q(\underline{\lambda}, \underline{x}, \underline{y})$ with respect to $\underline{y}$ is equivalent to:

$$\max_{0 \le y_j \le 1} f_j(\lambda_j, y_j) = \ln(1 - y_j) + \lambda_j \ln(y_j) \text{ for } t_j \in T_f \quad (16)$$

The maximum of this function is $y_j^*(\lambda_j) = \frac{\lambda_j}{1+\lambda_j} u(\lambda_j)$. At the value of $y_j^*(\lambda_j)$, the first and second derivatives of the function are zero and negative, respectively, indicating that $f_j(\lambda_j, y_j)$ is a maximum (where $u(.)$ is the unit step function).

The maximization of the Lagrangian function $Q(\underline{\lambda}, \underline{x}, \underline{y})$ with respect to $\underline{x}$ is equivalent to:

$$\max_{\underline{x}} \quad W(\underline{\lambda}, \underline{x}) = \sum_{i=1}^{m} c_i(\underline{\lambda}) x_i \quad (17)$$

subject to (13) and (14), which is a traditional set-covering problem. This problem has been extensively studied by the operations research and management science communities [2, 7]. There exist a number of optimal algorithms, based on feasible solution exclusion constraints, Gomory f-cuts and tree-search procedures for this problem [2, 7]. Let $\underline{x}^*(\underline{\lambda})$ be the optimal solution of this set-covering problem. Thus, $Q(\underline{\lambda}, \underline{x}^*(\underline{\lambda}), \underline{y}^*(\underline{\lambda}))$ is an upper bound for the optimal objective value in (10). This result is summarized in the following Lemma:

Lemma 4: Let J^* be the optimal value of the objective function in (10). Then $Q(\underline{\lambda}, \underline{x}^*(\underline{\lambda}), \underline{y}^*(\underline{\lambda})) \geq J^*$ for any $\underline{\lambda}$.

Proof: Let $\underline{x}^o$ and $\underline{y}^o$ be the optimal solution of the problem in (10). Thus, $Q(\underline{\lambda}, \underline{x}^o, \underline{y}^o) \leq Q(\underline{\lambda}, \underline{x}^*(\underline{\lambda}), \underline{y}^*(\underline{\lambda}))$. This is because, $\underline{x}^*(\underline{\lambda})$ and $\underline{y}^*(\underline{\lambda})$ are optimal with respect to the relaxed problem in (15). Since the optimal solutions $\underline{x}^o$ and $\underline{y}^o$ satisfy (11), we have $Q(\underline{\lambda}, \underline{x}^o, \underline{y}^o) = J^*$, and therefore, $J^* \leq Q(\underline{\lambda}, \underline{x}^*(\underline{\lambda}), \underline{y}^*(\underline{\lambda}))$.

After evaluating the optimum values $\underline{x}^*(\underline{\lambda})$ and $\underline{y}^*(\underline{\lambda})$ for a fixed $\underline{\lambda}$, the problem reduces to one of minimizing the upper bound $Q(\underline{\lambda})$= $Q(\underline{\lambda}, x^*(\underline{\lambda}), y^*(\underline{\lambda}))$. Since $Q(\underline{\lambda})$ is a piecewise function of $\underline{\lambda}$, this problem cannot be solved using differentiable optimization algorithms. As an alternative, we use a subgradient optimization algorithm [13] to produce a sequence of upper bounds for $Q(\underline{\lambda})$.

If we denote by Q^*, the optimal Lagrangian function value, i.e., Q^*= $Q(\underline{\lambda}^*)$=$\min_{\underline{\lambda}} Q(\underline{\lambda})$, the difference $(Q^* - J^*)$ is termed the exact duality gap. Since the problem in (10)-(14) is NP-hard [4], we may never know the global optimal solution J^*. Instead, we construct several feasible solutions to this problem from the Lagrangian function solution, and select the best feasible solution from the set. Let $J(\underline{\lambda}^*, \underline{x}^J, \underline{y}^J)$ be the best feasible value, then we have, $J(\underline{\lambda}^*, \underline{x}^J, \underline{y}^J) \leq J^* \leq Q^*$. A nice feature of the Lagrangian relaxation method is that the approximate duality gap:

$$Q^* - J(\underline{\lambda}^*, \underline{x}^J, \underline{y}^J) = (Q^* - J^*) + (J^* - J(\underline{\lambda}^*, \underline{x}^J, \underline{y}^J)) \geq 0 \quad (18)$$

provides an overestimate (by the value of the exact duality gap, $(Q^* - J^*)$) of the error between the global optimal solution and the best feasible solution found. Thus, in some cases, even though the best optimal solution found is the optimum solution of the problem, the approximate duality gap may be nonzero, see Example 1 in Section VI. Based on extensive computational experiments, the relative approximate duality gap, δJ, defined by:

$$\delta J = |\frac{Q^* - J(\underline{\lambda}^*, \underline{x}^J, \underline{y}^J)}{J(\underline{\lambda}^*, \underline{x}^J, \underline{y}^J)}| \quad (19)$$

is small for the multiple fault diagnosis algorithms (typically less than 5%). The pseudocode of the multiple fault diagnosis algorithm is presented in the next section.

A. Multiple Fault Diagnosis Algorithm

Let $(\underline{x}^J, \underline{y}^J)$, Q_{min}, Q_{ub} and Q_{lb} be the best feasible solution found, minimum upper bound, current upper bound and maximum lower bound (function value based on the best feasible solution found, i.e., $J(\underline{x}^J, \underline{y}^J)$) for $Q(\underline{\lambda}, \underline{x}, \underline{y})$, respectively. The pseudocode of multiple fault diagnosis algorithm is shown in Figure 3.

Initialization: Initialize: (1) $\lambda_j = 1$ for $j = 1, ..., |T_f|$, (2) $Q_{min} = \infty$, (3) $Q_{lb} = -\infty$, and (4) set iteration count $t = 1$. The reason for initializing $\lambda_j = 1$ is that it results in $y_j^* = 0.5$.

Step 1: Find optimum values $\underline{x}^*(\underline{\lambda})$ by solving the set-covering problem in (17).

Step 2: Find optimum values $\underline{y}^*(\underline{\lambda})$ where $y_j^*(\lambda_j) = \frac{\lambda_j}{1+\lambda_j} u(\lambda_j)$ for $j = 1, ..., |T_f|$.

Step 3: Evaluate $\underline{y}(\underline{x}^*(\underline{\lambda}))$ using equation (11).

Step 4: Update $\underline{x}^J$, $\underline{y}^J$, Q_{min}, Q_{ub} and Q_{lb} as follows:
- If $J(\underline{x}^*(\underline{\lambda}), \underline{y}(\underline{x}^*(\underline{\lambda}))) \geq Q_{lb}$, then $\underline{x}^J = \underline{x}^*(\underline{\lambda})$, $\underline{y}^J = y(\underline{x}^*(\underline{\lambda}))$, and $Q_{lb} = J(\underline{x}^*(\underline{\lambda}), \underline{y}(\underline{x}^*(\underline{\lambda})))$,
- $Q_{ub} = Q(\underline{\lambda}, \underline{x}^*(\underline{\lambda}), \underline{y}^*(\underline{\lambda}))$,
- $Q_{min} = \min(Q_{min}, Q_{ub})$.

Step 5: Calculate the subgradient $d_j = \ln(\frac{\lambda_j}{1+\lambda_j}) - \{\sum_{i=1}^{m} x_i^*(\underline{\lambda}) \ln(\frac{\overline{Pd_{ij}}}{\overline{Pf_{ij}}}) + h_j\}$ for $j = 1, ..., |T_f|$.

Step 6: Stop if $\sum_{j=1}^{|T_f|} d_j^2 = 0$ since in this case we cannot define a suitable step size.

Step 7: Define a step size β by $\beta = -f \frac{(\alpha Q_{ub} - Q_{lb})}{(\sum_{j=1}^{|T_f|} d_j^2)}$ where initially $f = 2$. If Q_{min} has not decreased in the last 10 iterations of the subgradient procedure with the current value of f, then f is halved. This approach to deciding the value of f is based on the procedure of Fisher [6]. The parameter α with typical value $1 \leq \alpha \leq 1.1$ is to ensure that β does not become too small as the gap between Q_{ub} and Q_{lb} decreases [1].

Step 8: Stop if $f \leq 0.05$ or $t \geq 100$ (or any other suitable stopping criteria).

Step 9: Update the Lagrange multipliers λ_j as follows: $\lambda_j = \max(0, \lambda_j + \beta d_j)$ for $t_j \in T_f$, $t \leftarrow t + 1$, and go to step 1.

Figure 3: Pseudocode of MFD Algorithm

B. Improving the Computational Complexity

The computational complexity of MFD algorithm for all steps except the first step is $O(m|T_f|)$. It is well known that the set-covering problem is NP-hard [11], and therefore, the first step of the multiple fault diagnosis algorithm limits the size of the problem that we can solve.

One of the important points here is that a near-optimal solution as well as an upper bound solution for the set-covering problem can be found via Lagrangian relax-

ation method [1] in a manner similar to the MFD algorithm. Let $\underline{x}^n(\underline{\lambda})$ and $\underline{x}^u(\underline{\lambda})$ denote the near-optimal (best feasible solution found) and upper bound solution for the set-covering problem, respectively. Note that any feasible solution for the set-covering problem is a feasible solution for the multiple fault diagnosis problem. However, for a given $\underline{\lambda}$, the best feasible solution for set-covering may not be the best feasible solution for the multiple fault diagnosis problem. Therefore, we have: $J(\underline{x}^n(\underline{\lambda}), \underline{y}(\underline{x}^n(\underline{\lambda}))) \leq J^* \leq Q(\underline{\lambda}, \underline{x}^*(\underline{\lambda}), \underline{y}^*(\underline{\lambda}))$ $\leq Q(\underline{\lambda}, \underline{x}^u(\underline{\lambda}), \underline{y}^*(\underline{\lambda}))$. Thus, using $\underline{x}^u(\underline{\lambda})$ and $\underline{x}^n(\underline{\lambda})$, we can generate a sequence of upper and lower bounds to the multiple fault diagnosis problem. In this case, the multiple fault diagnosis algorithm should be modified as follows: replace the optimal solution $\underline{x}^*(\underline{\lambda})$ in the algorithm with the near-optimal solution $\underline{x}^n(\underline{\lambda})$, except in Q_{ub} where $\underline{x}^*(\underline{\lambda})$ should be replaced by the upper bound solution $\underline{x}^u(\underline{\lambda})$. By this modification, the computational complexity of this approach reduces to $O(m|T_f|)$, and therefore, can be applied to large-scale systems. Note that, because of storage complexity of storing Pd_{ij} and Pf_{ij} for all failure sources and tests, the available memory of a given computer may limit the largest size of the problem that we can solve.

In large-scale systems, it is practical to assume that the detection and false alarm probabilities of each test t_j is the same for all failure sources connected to it, i.e., Pd_{ij}=Pd_j and Pf_{ij}=Pf_j, if $e_{ij} \in E$, otherwise, Pd_{ij}=0 and Pf_{ij}=0. In this case, we define a binary reachability matrix $R = \{r_{ij}\}$ such that $r_{ij} = 1$ if $e_{ij} \in E$, otherwise, $r_{ij} = 0$. The detection and false alarm probabilities of each test t_j for each failure source s_i can be evaluated as follows: $Pd_{ij} = r_{ij}Pd_j$ and $Pf_{ij} = r_{ij}Pf_j$. Note that, in this case, the binary matrix $R = r_{ij}$ can be stored in a bit-compact format, and consequently, the storage complexity of the problem reduces by a factor of approximately $2K$, where K is the number of bits for representing a floating variable in a given computer. For example, the storage complexity of the MFD problem for a system with 10,000 failure sources and tests when K=32 bits (or equivalently 4 bytes) are 800 Mbytes for storing Pd_{ij} and Pf_{ij}, and 12.5 Mbytes for storing the binary matrix $R = \{r_{ij}\}$. However, by storing Pd_j, pf_j and $R = \{r_{ij}\}$, the total memory required reduces to 12.6 Mbytes.

Despite the complexity analysis results for the combinatorial nature of multiple fault problem, the optimal solution for this problem can be found via a branch-and-bound. In the branch-and-bound algorithm: (1) a binary tree is employed for the representation of the 0-1 combinations, (2) the feasible region is partitioned into subdomains systematically, and (3) valid upper and lower bounds are generated at different levels of the binary tree. The main objective in a general branch-and-bound algorithm is to perform an enumeration of the alternatives without examining all 0-1 combinations of failure sources. Details of branch-and-bound algorithms can be found in any integer programming textbooks, e.g., [8, 13, 17, 18].

IV. Ranked Set of Most Likely Candidates

In this section, we consider the problem of determining a ranked set of solutions to the multiple fault diagnosis problem. That is, the problem is to find L sets of most likely candidates. We present the following sequential approach to solve this problem:

Initialization: Find the first most likely candidate X^1 for the multiple fault diagnosis problem.

Algorithm:

DO for $l = 2, ..., L$, or until no feasible solution exists,
Eliminate the set of candidates $\{ X^1,...,X^{l-1} \}$ from the problem and generate the l-th most likely candidate.

END

The first part of the algorithm, i.e., initialization, can be solved by the algorithm of previous section. In this section, we present an approach to solve the second part of the sequential algorithm. In this approach, we solve a series of modified copies of the initial multiple fault diagnosis.

A. Ranked Algorithm: Modified Copies of MFD Problem

In this approach, at each iteration, we solve a series of multiple fault diagnosis problems assuming that the states of some of the failure sources are known prior to diagnosis, i.e., some failure sources are known *good*, and some of them are known *bad* (definitely faulty). A similar approach has been considered by Murty [12] for determining a ranked set of solutions to assignment problems, and was recently enhanced by Cox et. al. [5] within the context of multi-target tracking. For simplicity, we represent the multiple fault diagnosis problem by four-tuple Γ=(MFD, G, B, X), where

1. MFD is the problem in (10)-(14),
2. $G \subseteq S$ represents the set of known good failure sources, i.e., for all $s_i \in G$, $x_i = 0$ (or $s_i \notin X$),
3. $B \subseteq S$ represents the set of definitely faulty failure sources, i.e., for all $s_i \in B$, $x_i = 1$ (or $s_i \in X$),

4. X is the optimal solution to the MFD problem subject to G and B.

Note that the number of unknown failure sources in Γ=(MFD, G, B, X) is m - $|G|$ - $|B|$. Initially, G and B are empty, i.e., $\Gamma^1 = (MFD, \emptyset, \emptyset, X^1)$. Subsequent solutions to Γ^1 are found by solving a succession of multiple fault diagnosis problems that are created from Γ^1 by a process called *partitioning*. A problem, Γ, with the best solution X and size $m - |G| - |B|$, is partitioned into a set of subproblems, Γ_1, ..., $\Gamma_{m-|G|-|B|+1}$, such that:

- The union of the set of possible solutions to Γ_1 through $\Gamma_{m-|G|-|B|+1}$ is exactly the set of possible solutions to Γ,
- The sets of possible solutions to Γ_1 through $\Gamma_{m-|G|-|B|+1}$ are disjoint, and
- $\Gamma_{m-|G|-|B|+1}$ has only one solution X.

Let us assume that Γ^r is a dummy subproblem that is used to generate the subproblems Γ_1 through $\Gamma_{m-|G|-|B|+1}$ from Γ. The following procedure shows: (1) how to update the subproblem $\Gamma^r = (MFD, G^r, B^r, X^r)$, and (2) how to make subproblem $\Gamma_l = (MFD, G_l, B_l, X_l)$ form Γ^r for $l = 1, ..., m - |G| - |B|$, sequentially, and finally, (3) $\Gamma_{m-|G|-|B|+1} = \Gamma^r$. Initially, Γ^r=Γ. Then, for $l = 1, ..., m-|G|-|B|$, Γ^r is partitioned as follows:

- Select any $s_i \in S - (G^r \cup B^r)$,
- If $s_i \in X$, then $G_l \leftarrow G^r \cup \{s_i\}$ and $B^r \leftarrow B^r \cup \{s_i\}$, else $B_l \leftarrow B^r \cup \{s_i\}$ and $G^r \leftarrow G^r \cup \{s_i\}$.

Note that, at each iteration, the problem Γ^r is partitioned into two disjoint subproblems. This is because we force the subproblems to be different in the status of only one failure source s_i in the system, i.e., we add s_i to the set of definitely faulty failure sources in one subproblem, and to the set of known good failure sources in another subproblem. In addition, X cannot be a solution to Γ_l for $l = 1, ..., m - |G| - |B|$. Further more, Γ^r is the only subproblem which contains X and only X as its solution. This is because B^r=X and G^r=$S - X$.

As an illustration, let us consider a simple system with 3 failure sources $\{s_1, s_2, s_3\}$. In addition, let us assume that the optimal solution for the MFD problem in this case is X=$\{s_1\}$, i.e., $\Gamma^1 = (MFD, \emptyset, \emptyset, X^1 = \{s_1\})$. Therefore, the MFD problem can be partioned into the following subproblems; $\Gamma_1 = (MFD, G = \{s_1\}, B = \emptyset, X_1)$; $\Gamma_2 = (MFD, G = \emptyset, B = \{s_1, s_2\}, X_2)$, $\Gamma_3 = (MFD, G = \{s_2\}, B = \{s_1, s_3\}, X_3)$, and $\Gamma_4 = (MFD, G = \{s_2, s_3\}, B = \{s_1\}, X_4)$.

Therefore, we partition Γ^1 according to its best solution X^1, and place the resulting subproblems together with their best solutions, except the last one, i.e., $\Gamma_{m-|G|-|B|+1}$, on a priority queue of four-tuple (MFD, G, B, X). We then find a problem in the queue that has the best solution. The solution of this problem is the second-best solution to the multiple fault diagnosis problem. Now, we remove this problem from the queue and replace it by its partitioning. The best solution found in the queue now is the third-best solution to the multiple fault diagnosis problem, and so on. The pseudocode for the L-ranked algorithm is shown in Figure 4.

Initialization: Find the first solution X^1 to MFD problem, and initialize a priority queue of four-tuple problems to contain only Γ^1=(MFD, $\emptyset$, $\emptyset$, X^1). The top problem on this queue will always be the problem with the highest likelihood solution.

Step 1: Clear the list of solutions to be returned.
Step 2: DO until priority queue of problems is empty.
  Step 2.1: Take the top problem
    Γ=(MFD, G, B, X) off the queue.
  Step 2.2: Add X to the list of solutions.
  Step 2.3: If the cardinality of solution set is L, Stop.
  Step 2.4: Let Γ^r =Γ,
  Step 2.5: DO for $l = 1, ..., m - |G| - |B|$,
    Step 2.5.1: Partition Γ^r into Γ^r and Γ' as follows:
    Step 2.5.2: Select any $s_i \in S - (G^r \cup B^r)$,
    Step 2.5.3: If $s_i \in X$, then
      $G' \leftarrow G^r \cup \{s_i\}$ and $B^r \leftarrow B^r \cup \{s_i\}$,
      else $B' \leftarrow B^r \cup \{s_i\}$ and $G^r \leftarrow G^r \cup \{s_i\}$.
    Step 2.5.4: Find the best solution X' to Γ'. If X' exists, add (MFD, G', B', X') to the queue.
    END
  END

Figure 4: Pseudocode for L-Rank MFD Algorithm

Since each subproblem is NP-hard, we use the near-optimal MFD algorithm of previous section to solve the ranked set problem near-optimally, i.e., X is a near-optimal solution for the problem Γ=(MFD, G, B, X). Thus, it is possible that l-th solution, i.e., X^l, has higher likelihood than the k-th solution, i.e., X^k, where $k > l$. Note that, we perform one partitioning for each of the L-best solution, in the worst case, each partitioning creates $O(m)$ new problems. This creates up to $O(Lm)$ multiple fault problems and insertions on the priority queue. Each problem takes at most $O(m|T_f|)$ time to solve near-optimally, and each insertion takes at most $O(\log(Lm))$ time. Therefore, the worst-case execution time of this approach is $O(Lm(m|T_f| + \log(Lm)))$, or approximately, $O(Lm^2|T_f|)$.

V. Multiple Fault Diagnosis with Repetitive Tests

A reasonable and common situation in unreliable testing is to apply a test several times to improve the confidence about a given hypothesis (a set of multiple fault candidates). For example, in order to reduce the probability of error, i.e., false alarm and missed detection of some faults (disorders or diseases), a system (a patient) may be tested multiple times, and because of imperfect nature of tests, the test results may be different. In this section, we assume that each test t_j has been applied n_j times in which it passed and failed μ_j and η_j times, respectively, i.e., $n_j = \mu_j + \eta_j$. Note that applying a test at different times is equivalent to applying independent tests with the same structure. In this case, let us assume that T_f and T_p denote the set of failed and passed tests (without any redundancy), respectively, and $T_f \cap T_p$ may not be empty. Thus, the problem is:

$$\max_{\underline{x},\underline{y}} \quad J(\underline{x},\underline{y}) = \sum_{t_j \in T_f} \eta_j \ln(1-y_j) + \sum_{i=1}^{m} x_i \{\sum_{t_k \in T_p} \mu_k \ln(\frac{\overline{Pd}_{ik}}{\overline{Pf}_{ik}}) + \ln(p_i)\} \quad (20)$$

subject to (11)-(14). This problem is similar to the problem in (10). Thus, the algorithms in previous sections can be readily applied to solve this problem. In this case: (1) in the first step of the MFD algorithm, $c_i(\underline{\lambda})$ is a function of μ_k for $k = 1, ..., |T_p|$, i.e., the number of time that test t_k passed, and (2) in the second step of the MFD algorithm, the optimum of the objective function with respect to $\underline{y}$ is replaced by $y_j^*(\lambda_j) = \frac{\lambda_j}{\eta_j + \lambda_j} u(\lambda_j)$ for $j = 1, ..., |T_f|$.

VI. Examples

Example 1: In this example, we consider: (1) a simple diagnostic problem with $m = 20$ failure sources (disorders) and $n = 20$ tests (manifests) which was used as an example in [3]; (Example 1.a - 1.d), and (2) a diagnostic problem with $m = 15$ failure sources and $n = 10$ tests from [9]; (Example 1.e). The false alarm probabilities for these systems are all zero, i.e., $Pf_{ij} = 0$ for $i = 1, ..., m$ and $j = 1, ..., n$ and T_p= T - T_f. Figures 5 and 6 show: (1) the set of failed tests T_f, (2) diagnostic results, (3) likelihood, (4) processing time and total number of runs to converge to the diagnostic results, (5) total processing time and total number of runs, and (6) approximate duality gap. The diagnostic results are based on the near-optimal multiple fault diagnosis algorithm in Figure 3. The processing times for these examples are obtained by running the MFD algorithm on a SPARC 10. Binglin et. al. [3] presented a genetic algorithm which required

Ex.	$\{j \mid t_j \in T_f\}$	$\{s_i \mid s_i \in X\}$	$Prob(X \mid T_f, T_p)$
1.a	{1, 2, 4, 5, 7, 8, 13, 18}	{1, 9, 10, 14, 17}	$3.66e^{-09}$
1.b	{7, 8, 9, 11, 14, 15}	{4, 5, 17, 20}	$1.32e^{-10}$
1.c	{1, 3, 4, 6, 7, 11, 13, 15, 16}	{1, 5, 9, 14, 16, 17}	$6.82e^{-13}$
1.d	{1, 2, 3, 7, 8, 12, 13, 17}	{4, 5, 8, 14, 19}	$2.49e^{-09}$
1.e	{1, 2, 4, 5, 7, 8, 9, 10}	{3, 4, 9, 12, 13}	$7.77e^{-02}$

Figure 5: MFD Algorithm Results for Examples 1.a-1.e

Ex.	Convergence		Total		Approximate Duality Gap
	# Runs	Time (sec)	# Runs	Time (sec)	
1.a	6	0.170	58	0.310	4.66%
1.b	2	0.009	65	0.240	4.76%
1.c	3	0.050	68	0.340	4.69%
1.d	1	0.004	64	1.83	4.69%
1.e	2	0.007	58	0.15	4.52%

Figure 6: MFD Algorithm Results for Examples 1.a-1.e

10 minutes to find the set of diagnoses in Example 1.a with an IBM PS/2 Model 40 SX-20 MHZ microcomputer. These results show the superior performance of our algorithm compared to the algorithm in [3]. Miller et. al. [9] have not reported the processing time for Example 1.e. However, the largest problem that they considered contained 20 failure sources and 15 tests.

Example 2: In this example, we consider systems with: (1) m=n=100, m=n=500 and m=n=1000, (2) the probability of each failure source is set to a random number between (0.001, 0.5), (3) each test, on average, covers 5, 10 and 20 failure sources, (4) detection probabilities of a test associated with its covered failure sources are set to random numbers between (0,1), (5) the false alarm probabilities are assumed to be zero, and (6) the number of failed tests are 5, 10 and 20. Figures 7, 8 and 9 show the simulation results for these systems. Each row of these Figures represents the average of simulation results for 5 randomly generated systems. Note that, in most of the cases, the average approximate duality gaps are around 5%. However, in some of the cases, for example, the last row of Figure 7, the approximate duality gap is very large, i.e., 22.15%. In order to improve the solution (or, equivalently, approximate duality gap), we can apply the L-ranked algorithm. The average approximate duality gap based on 2-ranked algorithm for the last set of systems in Figure 7 reduces to 1.49%.

Example 3: In this example, we consider three systems with 10 failure sources and 10 tests as in [15]. The false alarm probabilities are assumed to be zero. The simulation results for 2^{10} possible combinations of test results are shown in Figure 10. The second column shows the number of correct cases out of 1024 possible combinations of test results. The third column shows the weighted probability of correct cases. The columns corresponding to N_d and N_f denote the unweighted probabilities of detection, i.e., the unweighted probability of

Average Test Coverage	$\|T_f\|$	Convergence # Runs	Convergence Time (sec)	Total # Runs	Total Time (sec)	Approximate Duality Gap
5	5	2	0.10	62	2.70	4.25%
5	10	16	0.38	63	3.11	3.75%
5	20	4	0.83	70	13.78	4.79%
10	5	3	0.25	63	3.35	6.75%
10	10	12	2.91	60	12.00	5.26%
10	20	9	3.01	83	31.00	9.59%
20	5	1	0.11	55	6.97	6.12%
20	10	2	0.85	58	28.81	10.96%
20	20	16	23.39	59	90.72	22.15%

Figure 7: Simulation Results for m=n=100

Average Test Coverage	$\|T_f\|$	Convergence # Runs	Convergence Time (sec)	Total # Runs	Total Time (sec)	Approximate Duality Gap
5	5	1	0.54	62	23.56	4.15%
5	10	3	1.62	72	44.58	3.96%
5	20	16	7.15	66	32.77	4.23%
10	5	1	0.48	58	26.99	4.82%
10	10	1	1.20	69	46.68	4.60%
10	20	15	19.25	67	85.37	6.60%
20	5	1	0.56	51	30.41	3.75%
20	10	6	6.55	64	60.38	3.00%
20	20	11	26.19	64	184.06	16.03%

Figure 8: Simulation Results for m=n=500

common faulty failure sources in the optimal and near-optimal solutions, and false alarm, i.e., the unweighted probability of faulty failure sources in the near-optimal solution and not in the optimal solution. Figure 11 shows the simulation results based on the 2-ranked algorithm. The average weighted (unweighted) accuracy based on the MFD algorithm and 2-ranked algorithm are 97.71% (94.99%) and 99.96%(99.77%), respectively.

Example 4: In this example, we apply the MFD and L-ranked algorithms to the medical example in [14, 19]. The system under consideration is for neuropsychiatric diagnosis. The system consists of 26 disorders (failure sources) from psychiatry and neurology which affect mental status. A list of 56 symptoms (tests) and signs was assembled for each disorder. There are 384 links in the system, each of which connects a disorder to a manifestation. Similar to [14], five groups of test cases are used to test the MFD and L-ranked algorithm. Manifestations are chosen randomly from the total set of 56 possible manifestations based on a uniform distribution. Each group of test cases consists of ten different sets of manifestations. Each case in the first test group has one present manifestation (failed test); each case in the other groups have 3, 5, 7 and 9 manifestations. If any randomly generated test result is inconsistent with the causal network, the case is discarded and a new one is generated. The inconsistent test results may occur because the causal network used in the experiment has some perfect tests, i.e., $Pd_{ij} = 1$ and $Pf_{ij} = 0$. Thus, after applying the first Lemma, and reducing the size of the problem, the second Lemma may not be satisfied, i.e., there exists a failed test that is not covered by any failure source. Simulation results show that among all 50 cases MFD algorithm and 2-ranked algorithm generate 98% and 100% optimal solutions, respectively. Peng and Reggia applied their competition-based connectionist methods to this causal network. Their algorithm generated 74% of globally optimal solutions, and 90% of one of the three globally optimal solutions.

Average Test Coverage	$\|T_f\|$	Convergence # Runs	Convergence Time (sec)	Total # Runs	Total Time (sec)	Approximate Duality Gap
5	5	2	3.39	67	102.09	5.89%
5	10	2	3.53	73	122.88	5.12%
5	20	2	5.02	67	136.56	4.59%
10	5	1	2.27	54	87.95	4.09%
10	10	2	3.83	55	99.53	4.27%
10	20	5	11.90	66	169.60	4.99%
20	5	1	3.26	53	103.29	4.26%
20	10	11	30.29	57	137.15	4.94%
20	20	26	139.14	76	374.54	9.42%

Figure 9: Simulation Results for m=n=1000

Example	Correct cases # Times (out of 1024)	Correct cases Weighted	N_d	N_f
3.a	992 (96.68%)	99.91%	98.65%	0.39%
3.b	971 (94.82%)	98.61%	97.60%	0.31%
3.c	955 (93.26%)	94.61%	97.66%	0.37%

Figure 10: MFD Alg. Results for Examples 3.a-3.c

Example	Correct cases # Times (out of 1024)	Correct cases Weighted	N_d	N_f
3.a	1024 (100%)	100%	100%	0.00%
3.b	1019 (99.51%)	99.92%	99.77%	0.06%
3.c	1022 (99.80%)	99.95%	99.92%	0.03%

Figure 11: 2-ranked Alg. Results for Examples 3.a-3.c

VII. Conclusion

In this paper, we considered the problem of constructing optimal and near-optimal multiple fault diagnosis in bipartite systems with unreliable (imperfect) tests. We presented a multiple fault diagnosis algorithm based on Lagrangian relaxation and subgradient optimization method, which provides near optimal solutions for the multiple fault diagnosis, and upper bounds for an optimal branch-and-bound algorithm. Computational results indicate that our algorithm can be used in systems with as many as 1000 faults. In addition, we presented an algorithm to generate the set of L-ranked multiple fault candidates. In this algorithm, we find the most likely candidate using the near optimal multiple fault diagnosis algorithm. Then, we partition the problem, based on the first solution, to a set of disjoint subproblems. The

solutions to these subproblems with the highest likelihood represents the second most likely candidates. This procedure is continued until L-ranked multiple fault diagnoses are found, or no more feasible solutions exist. We showed that the computational complexity of this approach is $O(Lm^2|T_f|)$, and therefore, applicable for systems with as many as 1000 faults and tests. Finally, we extended the multiple fault diagnosis problem to redundant or repetitive tests. In this case, the problem is very similar to the original multiple fault diagnosis problem, and therefore, the MFD algorithm can be extended to this problem as well.

In this paper, we assumed that the test results are known prior to diagnosis. That is, we considered the problem of multiple fault *diagnosis* with unreliable tests. The problem of *sequential* multiple fault diagnosis *strategy (testing)* with unreliable tests is an important problem in field maintenance. Furthermore, the order of partitioning in the L-ranked algorithm may improve the accuracy of the near-optimal solutions. We expect to investigate these challenging issues in our future efforts.

Acknowledgment

We would like to thank Yun Peng, Jim Reggia and Jonathan Wald for allowing us to use the data for the medical application example.

References

[1] J.E. Beasley. A Lagrangian heuristic for set-covering problems. *Naval Research Logistics*, 37:151–163, 1990.

[2] J.E. Beasley. Enhancing an algorithm for set covering problems. *European Journal of Operational Research*, 58:293–300, 1992.

[3] Z. Binglin, Y. Tinghu, and H. Ren. A genetic algorithm for diagnosis problem solving,. *Conference Proceedings. International Conference on Systems, Man and Cybernetics*, 2:404–408, 1993.

[4] G.F. Cooper. The computational complexity of probabilistic inference using bayesian belief networks. *Artificial Intelligence*, 42:393–405, 1990.

[5] I.J. Cox and M.L. Miller. On finding ranked assignments with application to multi-target tracking and motion correspondence. *IEEE Transactions on AES*, 32(1):486–489, 1995.

[6] M.L. Fisher. The Lagrangian relaxation method for solving integer programming problems. *Management Science*, 27(1):1–18, 1981.

[7] M.L. Fisher and P. Kedia. Optimal solution of set covering/partitioning problems using dual heuristics. *Management Science*, 36:674–688, 1990.

[8] C.A. Floudas. *Nonlinear and Mixed-Integer Optimization, Fundamentals and applications.* Oxford University Press, 1995.

[9] J.A. Miller, W.D. potter, R.V. Gandham, and C.N. Lapena. An evaluation of local improvement operators for genetic algorithms. *IEEE Transaction on Systems, Man, and Cybernetics*, 23(5):1340–1351, September/October 1993.

[10] R.A. Miller, M.A. McNeil, S.M. Challinor, F.E. Masarie, and J.D. Myers. The internist-1/quick medical reference project. *West. J. Med. 145*, pages 816–822, 1986.

[11] R.E. Miller and J.W. Thatcher. *Complexity of Computer Computations.* Plenum Press, 1972.

[12] K.G. Murty. An algorithm for ranking all the assignments in order of increasing cost. *Operations Research*, 16:682–687, 1968.

[13] G.L. Nemhauser and Wolsey L.A. *Integer and Combinatorial Optimization.* John Wiley and Sons, 1988.

[14] Y. Peng and J.A. Reggia. *Abductive Inference Models for Diagnostic Problem-Solving.* Springer-Verlag, 1990.

[15] Y. Peng and J.A. Reggia. A connectionist model for diagnostic problem solving. *IEEE Transaction on Systems, Man, and Cybernetics*, 19(2):285–298, March/April 1989.

[16] Y. Peng and J.A. Reggia. A probabilistic causal model for diagnostic problem solving, part II: Diagnostic strategy *IEEE Transaction on Systems, Man, and Cybernetics*, 17(3):395–406, May/June 1987.

[17] A. Schrijver. *Theory of Linear and Integer Programming.* John Wiley and Sons, 1986.

[18] M.M. Syslo, N. Deo, and J.S. Kowalik. *Discrete Optimization Algorithms.* Prentice-Hall, Inc, 1983.

[19] J. Wald, M. Farach, M. Tagamets, and J. Reggia. Generating plausible diagnostic hypotheses with self processing causal networks. *Journal of Experimental and Theoretical AI*, 35:91–112, 1991.

[20] T.D. Wu. A problem decomposition method for efficient diagnosis and interpretation of multiple disorderss. *Computer Methods and Programs in Biomedicine*, 35:239–250, 1991.

Algorithms for Multiple Fault Diagnosis with Unreliable Tests

Mojdeh Shakeri[†], Vijaya Raghavan[†], Krishna Pattipati[††], and Ann Patterson-Hine[†††]

Qualtech Systems Inc.[†]
Box-407, Mansfield Center, CT 06250
e-mail: mojdeh@sol.uconn.edu

Department of Electrical and Systems Engineering[††]
University of Connecticut, Storrs, CT 06269-3157
e-mail: krishna@sol.uconn.edu

NASA-Ames Research Center, Mail Stop 269-4[†††]
Moffett Field, CA 94035-1000
e-mail: Ann_Patterson-Hine@styx.arc.nasa.gov

Abstract

In this paper, we consider the problem of constructing optimal and near-optimal multiple fault diagnosis (MFD) in bipartite systems with unreliable (imperfect) tests. It is known that exact computation of conditional probabilities for multiple fault diagnosis is NP-hard. The novel feature of our diagnostic algorithms is the use of Lagrangian relaxation and subgradient optimization methods to provide: (1) near optimal solutions for the MFD problem, and (2) upper bounds for an optimal branch-and-bound algorithm. The proposed method is illustrated using several examples. Computational results indicate that: (1) our algorithm has superior computational performance to the existing algorithms (approximately three orders of magnitude improvement over the algorithm in [3]), (2) the near optimal algorithm generates the most likely candidates with a very high accuracy, and (3) our algorithm can find the most likely candidates in systems with as many as 1000 faults.

I. Introduction

With the increased recognition of importance of design for testability, there is an increasing trend towards the use of smart sensors for on-board system health management. The results of on-board tests are available to the ground test systems and operators as a *block* of symptoms. Due to improper set up, operator error, electromagnetic interference, environmental conditions, or aliasing inherent in the signature analysis of on-board tests, the nature of tests may be unreliable (imperfect). Imperfect tests introduce an additional element of uncertainty into the diagnostic process: the pass outcome of a test does not guarantee the integrity of components under test (because the test may have missed a fault), or a failed test outcome does not mean that one or more of the implicated components are faulty (because the test outcome may have been a false alarm). Consequently, the diagnostic procedures must hedge against this uncertainty in test outcomes.

In this paper, we consider the problem of constructing optimal and near-optimal multiple fault diagnosis in bipartite digraphs with unreliable tests. This problem is a central and long-standing concern in system fault diagnosis, and medical decision making [10]. When the false alarm probabilities of all tests are zero, the problem simplifies to the parsimonious covering theory (or probabilistic causal model) discussed in [16]. Peng and Reggia [15] proposed a competition based connectionist method to subdue the problem of combinatorial explosion in computing the posterior probabilities of all possible combinations of failure sources in probabilistic causal models. However, this method does not guarantee a global optimum and suffers from large computation times even for problems with small numbers of failure sources, m=26.

Genetic algorithms are offered as an alternative to the connectionist methods [3, 9]. Genetic algorithms are based on an analogy with Darwin's biological evolutionary theory in which a group of solutions evolves via natural selection. It emulates the rules of biological evolutionary process, such as reproduction, crossover, mutation, and natural selection, etc. At each iteration, a population of

individuals is established, where each individual corresponds to a point in the search space. The objective function is evaluated for each individual to rate its fitness. Then, a next generation is formed based on the survival of the fittest. Therefore, the evolution of individuals from generation to generation tends to result in fitter individuals (i.e., solutions) in the search space. These algorithms converge extremely slowly, and have been applied to small problems with m=20 failure sources (causes, disorders) and n=20 tests (manifestations, symptoms).

Wu [20] proposed a decomposition method based on common and disjoint causal (failure source) relationships among the given symptoms (tests). This method decomposes the original problem into smaller and independent subproblems, and therefore, increases the performance and efficiency of multiple fault diagnosis. However, this approach is not applicable for systems with large numbers of nondecomposable causes and symptoms.

In this paper, we present a novel approach, using Lagrangian relaxation, to solve multiple fault diagnosis problem. By defining new variables and constraints, the multiple fault diagnosis (MFD) problem reduces to a combinatorial optimization problem with a set of equality constraints. The constraints are relaxed via Lagrange multipliers. The relaxation procedure generates an upper bound for the objective function. The procedure of minimizing the upper bound via a subgradient optimization produces a sequence of solutions that are modified, in a computationally effective way, to produce a sequence of feasible solutions to the MFD problem. If the objective function value for the best feasible solution and the upper bound are the same, the feasible solution is the optimal solution. Otherwise, the difference between the upper bound and the feasible solution, termed the approximate duality gap, provides a measure of suboptimality of the MFD solution. Alternatively, the optimal solution can be found via a tree search (or branch-and-bound) procedure. The computational complexity of the near-optimal algorithm is a linear function of the number of failure sources, m and the number of failed tests, $|T_f|$.

Next, we present an approach to determine a ranked set of multiple fault diagnosis solutions

(i.e., the best, second best, ..., L-th best diagnosis). In this approach, following Murty [12] and Cox et. al. [5], we: (1) partition the MFD problem, based on its best solution, into disjoint subproblems; (2) solve the subproblems and sort them by the values of their solutions, and (3) select the subsequent best solutions. One of the advantages of this approach, compared to the one in [14], is that since the subproblems are disjoint, the optimal solution of each subproblem is different from the others. Finally, we show that the MFD algorithm can be extended to solve multiple fault diagnosis problems with repetitive application of tests.

The paper is organized as follows. In Section II, we formulate the multiple fault diagnosis problem in a bipartite system. In Section III, we present a near-optimal algorithm based on Lagrangian relaxation and subgradient optimization method to diagnose multiple faults, and generate an upper bound for the likelihood of multiple fault candidates. The upper bound can be used in an optimal branch-and-bound algorithm. The multiple fault algorithms for a set of L-ranked multiple fault diagnoses are presented in Section IV. In Section V, we consider the multiple fault diagnosis problem with repetitive tests. Several examples are presented in Section VI. Finally, in Section VII, we summarize the results and discuss future research issues.

II. Problem Formulation

The MFD problem in bipartite systems with imperfect tests consists of a bipartite digraph $DG = \{S, T, E\}$, where

- $S = \{s_1, ..., s_m\}$ is a finite set of *independent* failure sources (failure nodes) associated with the system;
- $T = \{t_1, t_2, ..., t_n\}$ is a finite set of n available binary outcome tests (test node), where the integrity of system failure sources/components/modules can be ascertained;

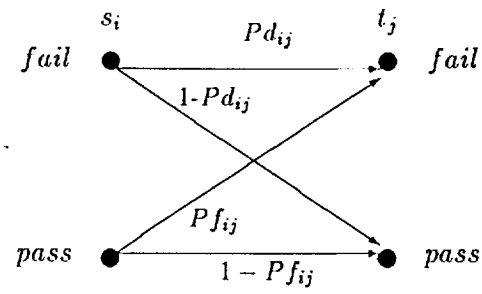

Figure 1: Detection-False-Alarm Probability of Failure Source s_i and Test t_j

- $E = \{e_{ij}\}$ is the set of digraph edges (links) specifying the functional information flow between the set of failure sources and the set of tests in the system.

The input requirements of the failure nodes and edges of the digraph are as follows:

1. Failure node: *A priori* probability vector of failure nodes $P = [p(s_1), ..., p(s_m)]$, where $p(s_i) > 0$ is the a priori probability of failure source s_i.

2. Link (edge): A set of probability pairs $P_{ij} = (Pd_{ij}, Pf_{ij})$ representing the detection-false-alarm probabilities of the set of tests, where Pd_{ij} and Pf_{ij} are the detection and false alarm probabilities of test t_j and failure source s_i, respectively (see Figure 1). Figure 2 shows a bipartite digraph model.

The problem is to find the most likely candidates $X \subseteq S$ that are consistent with the results of applied tests. This is formulated as:

$$\max_{X \subseteq S} \text{Prob}(X|T_p, T_f) \tag{1}$$

where $T_p \subseteq T$ and $T_f \subseteq T$ denote the set of passed and failed tests, respectively. Using Bayes' rule and eliminating the constant factor $\text{Prob}(T_p, T_f)$, we obtain the following equivalent maximization problem:

$$\max_{X \subseteq S} \text{Prob}(T_f, T_p|X)\text{Prob}(X) \tag{2}$$

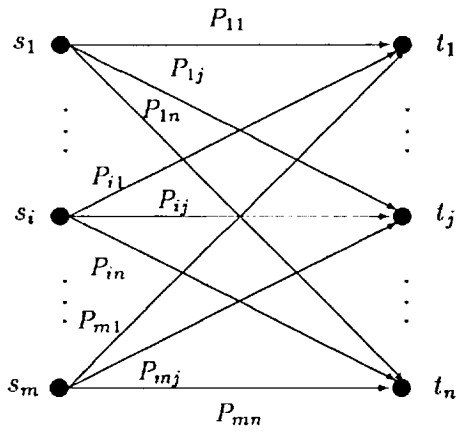

Figure 2: Illustration of the Bipartite Digraph Model

For notational simplicity, we define binary vector $\underline{x}$ of size m, where $x_i = 1$ if failure source $s_i \in X$; $x_i = 0$, otherwise. Note that, given a multiple fault candidate X, the tests are independent. Thus, the above probabilities can be evaluated as follows:

$$\text{Prob}(T_p|X) = \prod_{t_k \in T_p} \text{Prob}(O(t_k) = p|X) \tag{3}$$

$$\text{Prob}(T_f|X) = \prod_{t_j \in T_f} \text{Prob}(O(t_j) = f|X) \tag{4}$$

$$\begin{aligned}\text{Prob}(O(t_j) = p|X) &= \prod_{i=1}^{m}(1 - Pd_{ij})^{x_i}(1 - Pf_{ij})^{(1-x_i)} \\ &= [\prod_{i=1}^{m}(1 - Pf_{ij})][\prod_{i=1}^{m}(\frac{1 - Pd_{ij}}{1 - Pf_{ij}})^{x_i}]\end{aligned} \tag{5}$$

$$\text{Prob}(O(t_j) = f|X) = 1 - \text{Prob}(O(t_j) = p|X) \tag{6}$$

$$\begin{aligned}\text{Prob}(X) &= \prod_{i=1}^{m} p(s_i)^{x_i}(1 - p(s_i))^{(1-x_i)} \\ &= [\prod_{i=1}^{m}(1 - p(s_i))][\prod_{i=1}^{m}(\frac{p(s_i)}{1 - p(s_i)})^{x_i}]\end{aligned} \tag{7}$$

where $O(t_j) \in \{p(=\text{pass}), f(=\text{fail})\}$ is the outcome of test t_j, and $Pd_{ij} = 0$ and $Pf_{ij} = 0$ for $e_{ij} \notin E$.

III. Problem Solution

One approach for generating the optimal multiple fault diagnosis is to consider all possible combinations of failure sources, i.e., the power set 2^S, and select the multiple fault candidate with the highest likelihood function in (1). However, the computational complexity of this approach is exponential in the number of failure sources m. In the following, we present an algorithm, based on Lagrangian relaxation and subgradient optimization method, to generate a near-optimal solution for this problem.

By substituting (3) and (4) into (2) and taking the natural logarithm of the resulting objective function, the problem is equivalent to:

$$\max_{X \subseteq S} \sum_{t_j \in T_f} \ln(\text{Prob}(O(t_j) = f|X)) + \sum_{t_k \in T_p} \ln(\text{Prob}(O(t_k) = p|X)) + \ln(\text{Prob}(X)) \tag{8}$$

By substituting (5), (6) and (7) into (8), the problem reduces to:

$$\max_{X \subseteq S} \sum_{t_j \in T_f} \ln(1 - [\prod_{i=1}^{m} (\frac{\overline{Pd}_{ij}}{\overline{Pf}_{ij}})^{x_i}][\prod_{i=1}^{m} (\overline{Pf}_{ij})]) + \sum_{t_k \in T_p} \{\sum_{i=1}^{m} x_i \ln(\frac{\overline{Pd}_{ik}}{\overline{Pf}_{ik}}) + \sum_{i=1}^{m} \ln(\overline{Pf}_{ik})\} + \sum_{i=1}^{m} \{x_i \ln(p_i) + \ln(1 - p(s_i))\} \tag{9}$$

where $p_i = \frac{p(s_i)}{(1-p(s_i))}$, $\overline{Pf}_{ij} = 1 - Pf_{ij}$ and $\overline{Pd}_{ij} = 1 - Pd_{ij}$ for $i = 1, ..., m$ and $j = 1, ..., m$. By (i) eliminating constant factors $\sum_{i=1}^{m} \ln(1 - p(s_i))$ and $\sum_{t_k \in T_p} \sum_{i=1}^{m} \ln(\overline{Pf}_{ik})$, and (ii) defining new variables $y_j = [\prod_{i=1}^{m} (\frac{\overline{Pd}_{ij}}{\overline{Pf}_{ij}})^{x_i}][\prod_{i=1}^{m} (\overline{Pf}_{ij})]$ for $t_j \in T_f$, and taking the natural logarithm of it, the problem reduces to the following optimization problem:

$$\max_{\underline{x}, \underline{y}} J(\underline{x}, \underline{y}) = \sum_{t_j \in T_f} \ln(1 - y_j) + \sum_{i=1}^{m} x_i \{\sum_{t_k \in T_p} \ln(\frac{\overline{Pd}_{ik}}{\overline{Pf}_{ik}}) + \ln(p_i)\} \tag{10}$$

$$\text{subject to:} \qquad \ln(y_j) = \sum_{i=1}^{m} x_i \ln(\frac{\overline{Pd}_{ij}}{\overline{Pf}_{ij}}) + \sum_{i=1}^{m} \ln(\overline{Pf}_{ij}) \tag{11}$$

$$0 \leq y_j \leq 1 \quad \text{for } t_j \in T_f \tag{12}$$

$$x_i = 0 \text{ or } 1 \quad \text{for } i = 1, ..., m \tag{13}$$

where $\underline{y} = [y_1, ..., y_{|T_f|}]$, and $|.|$ denotes set cardinality. For simplicity, we define $h_j = \sum_{i=1}^{m} \ln(\overline{Pf}_{ij})$ for $t_j \in T_f$. The following lemmas present two important properties of the MFD problem.

Lemma 1: If $Pd_{ik} = 1$ for any passed test t_k, then the optimal solution does not contain failure source s_i, i.e., x_i=0 (or equivalently $s_i \notin X$).

Proof: If $s_i \in X$ (or $x_i = 1$), then the second part of the objective function in (10), and, consequently, the overall objective function will be unbounded, i.e., it would be -∞.

Using Lemma 1, the size of the MFD problem can be reduced by removing all failure sources $\{s_i | Pf_{ik}$=0, $Pd_{ik} = 1$ and $t_k \in T_p\}$ from the problem.

Lemma 2: If the false alarm probabilities of a failed test t_j are zero, i.e., Pf_{ij}=0 for $i = 1, ..., m$, then the optimal solution contains at least one x_i=1, for which $Pd_{ij} > 0$. That is, the optimal solution must cover the failed tests.

Proof: We prove this lemma by contradiction. Pf_{ij}=0 for $i = 1, ..., m$ results in h_j=0. If for all $Pd_{ij} > 0$, x_i=0, then we have $\ln(y_j)$=0 and, hence, $y_j = 1$. Thus, $\ln(1 - y_j)$, the first part of the objective function in (10), and, consequently, the overall objective function will be unbounded. Using Lemma 2, we define the following constraints:

$$A\underline{x} \geq \underline{e} \qquad \text{for } t_j \in T_f \text{ and } h_j = 0 \tag{14}$$

where $A = \{a_{lj}\}$ is a binary matrix of size $|H| \times m$; $H = \{t_j \in T_f | h_j = 0 \text{ for } j = 1, ..., n\}$; each row l of matrix A corresponds to a failed test t_j with $h_j = 0$; a_{li}=1, if $Pd_{ij} > 0$ for $i = 1, ..., m$; otherwise, a_{li}=0, and $\underline{e}$ is a vector of 1's.

Adding the set of constraints (14) to the problem in (10)-(13) results in a smaller search space

and tighter upper and lower bounds (which result in faster convergence), and, therefore, a better estimate of the optimal solution.

Lemma 3: When all tests are perfect, that is, $Pd_{ij} = 1$ and $Pf_{ij} = 0$ for $i = 1, ..., m, j = 1, ..., n$ and $e_{ij} \in E$, using Lemmas 1 and 2, the problem reduces to the following set-covering problem: $\max_{\underline{x}} \sum_{s_i \in S^-} \ln(p_i) x_i$ subject to (13) and (14), where S^- is the reduced set of failure sources, i.e., S after eliminating the failure sources satisfying Lemma 1.

Proof: This lemma can easily be proved by Lemmas 1 and 2. $Pd_{ij} = 1$ and $Pf_{ij} = 0$ for the failed tests results in $h_j = y_j = 0$. Therefore, the first part as well as the second part (using Lemma 1) of the objective function in (10) can be eliminated, and the problem reduces to the traditional set covering problem. The set covering problem can be solved optimally by any optimal set-covering algorithm [2, 7], or near-optimally via a Lagrangian relaxation and subgradient optimization method [1].

By relaxing the constraints in (11) via Lagrange multipliers $\{ \lambda_j \}$, we obtain the Lagrangian function:

$$\begin{aligned} \max_{\underline{x},\underline{y}} Q(\underline{\lambda}, \underline{x}, \underline{y}) = & \sum_{t_j \in T_f} \{\ln(1 - y_j) + \lambda_j \ln(y_j)\} + \\ & \sum_{i=1}^{m} x_i \{ \sum_{t_k \in T_p} \ln(\frac{\overline{Pd}_{ik}}{\overline{Pf}_{ik}}) + \ln(p_i) - \sum_{t_j \in T_f} \lambda_j \ln(\frac{\overline{Pd}_{ij}}{\overline{Pf}_{ij}}) \} - \\ & \sum_{t_j \in T_f} \lambda_j h_j \\ = & \sum_{t_j \in T_f} f_j(\lambda_j, y_j) + \sum_{i=1}^{m} c_i(\underline{\lambda}) x_i - \sum_{t_j \in T_f} \lambda_j h_j \end{aligned} \tag{15}$$

subject to (12), (13) and (14), where $f_j(\lambda_j, y_j)$ and $c_i(\underline{\lambda})$ denote the first and second expressions in the brackets in (15), respectively. ***The important point here is that the maximization of Lagrangian function in (15) with respect to $\underline{x}$ and $\underline{y}$ can be carried out independently for each fixed $\underline{\lambda}$.*** Maximization of $Q(\underline{\lambda}, \underline{x}, \underline{y})$ with respect to $\underline{y}$ is equivalent to:

$$\max_{0 \le y_j \le 1} f_j(\lambda_j, y_j) = \ln(1 - y_j) + \lambda_j \ln(y_j) \text{ for } t_j \in T_f \tag{16}$$

The maximum of this function is $y_j^*(\lambda_j) = \frac{\lambda_j}{1+\lambda_j} u(\lambda_j)$. At the value of $y_j^*(\lambda_j)$, the first and second derivatives of the function are zero and negative, respectively, indicating that $f_j(\lambda_j, y_j)$ is a maximum (where $u(.)$ is the unit step function).

The maximization of the Lagrangian function $Q(\underline{\lambda}, \underline{x}, \underline{y})$ with respect to $\underline{x}$ is equivalent to:

$$\max_{\underline{x}} W(\underline{\lambda}, \underline{x}) = \sum_{i=1}^{m} c_i(\underline{\lambda}) x_i \tag{17}$$

subject to (13) and (14), which is a traditional set-covering problem. This problem has been extensively studied by the operations research and management science communities [2, 7]. There exist a number of optimal algorithms, based on feasible solution exclusion constraints, Gomory f-cuts and tree-search procedures for this problem [2, 7]. Let $\underline{x}^*(\underline{\lambda})$ be the optimal solution of this set-covering problem. Thus, $Q(\underline{\lambda}, \underline{x}^*(\underline{\lambda}), \underline{y}^*(\underline{\lambda}))$ is an upper bound for the optimal objective value in (10). This result is summarized in the following Lemma:

Lemma 4: Let J^* be the optimal value of the objective function in (10). Then $Q(\underline{\lambda}, \underline{x}^*(\underline{\lambda}), \underline{y}^*(\underline{\lambda})) \geq J^*$ for any $\underline{\lambda}$.

Proof: Let $\underline{x}^o$ and $\underline{y}^o$ be the optimal solution of the problem in (10). Thus, $Q(\underline{\lambda}, \underline{x}^o, \underline{y}^o) \leq Q(\underline{\lambda}, \underline{x}^*(\underline{\lambda}), \underline{y}^*(\underline{\lambda}))$. This is because, $\underline{x}^*(\underline{\lambda})$ and $\underline{y}^*(\underline{\lambda})$ are optimal with respect to the relaxed problem in (15). Since the optimal solutions $\underline{x}^o$ and $\underline{y}^o$ satisfy (11), we have $Q(\underline{\lambda}, \underline{x}^o, \underline{y}^o) = J^*$, and therefore, $J^* \leq Q(\underline{\lambda}, \underline{x}^*(\underline{\lambda}), \underline{y}^*(\underline{\lambda}))$.

After evaluating the optimum values $\underline{x}^*(\underline{\lambda})$ and $\underline{y}^*(\underline{\lambda})$ for a fixed $\underline{\lambda}$, the problem reduces to one of minimizing the upper bound $Q(\underline{\lambda}) = Q(\underline{\lambda}, x^*(\underline{\lambda}), y^*(\underline{\lambda}))$. Since $Q(\underline{\lambda})$ is a piecewise differentiable function of $\underline{\lambda}$, this problem cannot be solved using differentiable optimization algorithms. As an alternative, we use a subgradient optimization algorithm [13] to produce a sequence of upper bounds for $Q(\underline{\lambda})$.

If we denote by Q^*, the optimal Lagrangian function value, i.e., $Q^* = Q(\underline{\lambda}^*) = \min_{\underline{\lambda}} Q(\underline{\lambda})$, the difference $(Q^* - J^*)$ is termed the exact duality gap. Since the problem in (10)-(14) is NP-hard [4],

we may never know the global optimal solution J^*. Instead, we construct several feasible solutions to this problem from the Lagrangian function solution, and select the best feasible solution from the set. Let $J(\underline{\lambda}^*, \underline{x}^f, \underline{y}^f)$ be the best feasible value, then we have, $J(\underline{\lambda}^*, \underline{x}^f, \underline{y}^f) \leq J^* \leq Q^*$. A nice feature of the Lagrangian relaxation method is that the approximate duality gap:

$$Q^* - J(\underline{\lambda}^*, \underline{x}^f, \underline{y}^f) = (Q^* - J^*) + (J^* - J(\underline{\lambda}^*, \underline{x}^f, \underline{y}^f)) \geq 0 \tag{18}$$

provides an overestimate (by the value of the exact duality gap, $(Q^* - J^*)$) of the error between the global optimal solution and the best feasible solution found. Thus, in some cases, even though the best optimal solution found is the optimum solution of the problem, the approximate duality gap may be nonzero, see Example 1 in Section VI. Based on extensive computational experiments, the relative approximate duality gap, δJ, defined by:

$$\delta J = \left| \frac{Q^* - J(\underline{\lambda}^*, \underline{x}^f, \underline{y}^f)}{J(\underline{\lambda}^*, \underline{x}^f, \underline{y}^f)} \right| \tag{19}$$

is small for the multiple fault diagnosis algorithms (typically less than 5%). The pseudocode of the multiple fault diagnosis algorithm is presented in the next section.

A. Multiple Fault Diagnosis Algorithm

Let $(\underline{x}^f, \underline{y}^f)$, Q_{min}, Q_{ub} and Q_{lb} be the best feasible solution found, minimum upper bound, current upper bound and maximum lower bound (function value based on the best feasible solution found, i.e., $J(\underline{x}^f, \underline{y}^f)$) for $Q(\underline{\lambda}, \underline{x}, \underline{y})$, respectively. The pseudocode of multiple fault diagnosis algorithm is shown in Figure 3.

Initialization: Initialize: (1) $\lambda_j = 1$ for $j = 1, ..., |T_f|$, (2) $Q_{min} = \infty$, (3) $Q_{lb} = -\infty$, and (4) set iteration count $t = 1$. The reason for initializing $\lambda_j = 1$ is that it results in $y_j^* = 0.5$.

Step 1: Find optimum values $\underline{x}^*(\underline{\lambda})$ by solving the set-covering problem in (17).

Step 2: Find optimum values $\underline{y}^*(\underline{\lambda})$ where $y_j^*(\lambda_j) = \frac{\lambda_j}{1+\lambda_j} u(\lambda_j)$ for $j = 1, ..., |T_f|$.

Step 3: Evaluate $\underline{y}(\underline{x}^*(\underline{\lambda}))$ using equation (11).

Step 4: Update $\underline{x}^f$, $\underline{y}^f$, Q_{min}, Q_{ub} and Q_{lb} as follows:

- If $J(\underline{x}^*(\underline{\lambda}), \underline{y}(\underline{x}^*(\underline{\lambda}))) \geq Q_{lb}$, then $\underline{x}^f = \underline{x}^*(\underline{\lambda})$, $\underline{y}^f = y(\underline{x}^*(\underline{\lambda}))$, and $Q_{lb} = J(\underline{x}^*(\underline{\lambda}), \underline{y}(\underline{x}^*(\underline{\lambda})))$,
- $Q_{ub} = Q(\underline{\lambda}, \underline{x}^*(\underline{\lambda}), \underline{y}^*(\underline{\lambda}))$,
- $Q_{min} = \min(Q_{min}, Q_{ub})$.

Step 5: Calculate the subgradient $d_j = \ln(\frac{\lambda_j}{1+\lambda_j}) - \{\sum_{i=1}^{m} x_i^*(\underline{\lambda}) \ln(\frac{\overline{Pd_{ij}}}{\overline{Pf_{ij}}}) + h_j\}$ for $j = 1, ..., |T_f|$.

Step 6: Stop if $\sum_{j=1}^{|T_f|} d_j^2 = 0$ since in this case we cannot define a suitable step size.

Step 7: Define a step size β by $\beta = -f \frac{(\alpha Q_{ub} - Q_{lb})}{(\sum_{j=1}^{|T_f|} d_j^2)}$ where initially $f = 2$. If Q_{min} has not decreased in the last 10 iterations of the subgradient procedure with the current value of f, then f is halved. This approach to deciding the value of f is based on the procedure of Fisher [6]. The parameter α with typical value $1 \leq \alpha \leq 1.1$ is to ensure that β does not become too small as the gap between Q_{ub} and Q_{lb} decreases [1].

Step 8: Stop if $f \leq 0.05$ or $t \geq 100$ (or any other suitable stopping criteria).

Step 9: Update the Lagrange multipliers λ_j as follows: $\lambda_j = \max(0, \lambda_j + \beta d_j)$ for $t_j \in T_f$, $t \leftarrow t + 1$, and go to step 1.

Figure 3: Pseudocode of *MFD* Algorithm

B. Computational Issues

The computational complexity of MFD algorithm for all steps except the first step is $O(m|T_f|)$. It is well known that the set-covering problem is NP-hard [11], and therefore, the first step of the multiple fault diagnosis algorithm limits the size of the problem that we can solve.

One of the important points here is that a near-optimal solution as well as an upper bound solution for the set-covering problem can be found via Lagrangian relaxation method [1] in a manner similar to the MFD algorithm. Let $\underline{x}^n(\underline{\lambda})$ and $\underline{x}^u(\underline{\lambda})$ denote the near-optimal (best feasible solution found) and upper bound solution for the set-covering problem, respectively. Note that any feasible solution for the set-covering problem is a feasible solution for the multiple fault diagnosis problem. However, for a given $\underline{\lambda}$, the best feasible solution for set-covering may not be the best feasible solution for the multiple fault diagnosis problem. Therefore, we have: $J(\underline{x}^n(\underline{\lambda}), \underline{y}(\underline{x}^n(\underline{\lambda}))) \leq J^* \leq Q(\underline{\lambda}, \underline{x}^*(\underline{\lambda}), \underline{y}^*(\underline{\lambda})) \leq Q(\underline{\lambda}, \underline{x}^u(\underline{\lambda}), \underline{y}^*(\underline{\lambda}))$. Thus, using $\underline{x}^u(\underline{\lambda})$ and $\underline{x}^n(\underline{\lambda})$, we can generate a sequence of upper and lower bounds to the multiple fault diagnosis problem. In this case, the multiple fault diagnosis algorithm should be modified as follows: replace the optimal solution $\underline{x}^*(\underline{\lambda})$ in the algorithm with the near-optimal solution $\underline{x}^n(\underline{\lambda})$, except in Q_{ub} where $\underline{x}^*(\underline{\lambda})$ should be replaced by the upper bound solution $\underline{x}^u(\underline{\lambda})$. By this modification, the computational complexity of this approach reduces to $O(m|T_f|)$, and therefore, can be applied to large-scale systems. Note that, because of storage complexity of storing Pd_{ij} and Pf_{ij} for all failure sources and tests, the available memory of a given computer may limit the largest size of the problem that we can solve.

In large-scale systems, it is practical to assume that the detection and false alarm probabilities of each test t_j is the same for all failure sources connected to it, i.e., Pd_{ij}=Pd_j and Pf_{ij}=Pf_j, if $e_{ij} \in E$, otherwise, Pd_{ij}=0 and Pf_{ij}=0. In this case, we define a binary reachability matrix $R = \{r_{ij}\}$ such that $r_{ij} = 1$ if $e_{ij} \in E$, otherwise, $r_{ij} = 0$. The detection and false alarm probabilities of each test t_j for each failure source s_i can be evaluated as follows: $Pd_{ij} = r_{ij}Pd_j$

and $Pf_{ij} = r_{ij}Pf_j$. Note that, in this case, the binary matrix $R = r_{ij}$ can be stored in a bit-compacted format, and consequently, the storage complexity of the problem reduces by a factor of approximately $2K$, where K is the number of bits for representing a floating variable in a given computer. For example, the storage complexity of the MFD problem for a system with $10,000$ failure sources and tests when K=32 bits (or equivalently 4 bytes) are 800 Mbytes for storing Pd_{ij} and Pf_{ij}, and 12.5 Mbytes for storing the binary matrix $R = \{r_{ij}\}$. However, by storing Pd_j, pf_j and $R = \{r_{ij}\}$, the total memory required reduces to 12.6 Mbytes.

Despite the complexity analysis results for the combinatorial nature of multiple fault problem, the optimal solution for this problem can be found via a branch-and-bound. In the branch-and-bound algorithm: (1) a binary tree is employed for the representation of the 0-1 combinations, (2) the feasible region is partitioned into subdomains systematically, and (3) valid upper and lower bounds are generated at different levels of the binary tree. The main objective in a general branch-and-bound algorithm is to perform an enumeration of the alternatives without examining all 0-1 combinations of failure sources. Details of branch-and-bound algorithms can be found in any integer programming textbooks, e.g., [8, 13, 17, 18].

IV. Ranked Set of Most Likely Candidates

In this section, we consider the problem of determining a ranked set of solutions to the multiple fault diagnosis problem. That is, the problem is to find L sets of most likely candidates. We present the following sequential approach to solve this problem:

Initialization: Find the first most likely candidate X^1 for the multiple fault diagnosis problem.

Algorithm: DO for $l = 2, ..., L$, or until no feasible solution exists,

Eliminate the set of candidates $\{ X^1,...,X^{l-1} \}$ from the problem and generate the l-th most likely candidate.

The first part of the algorithm, i.e., initialization, can be solved by the algorithm of previous section. In this section, we present an approach to solve the second part of the sequential algorithm. In this approach, we solve a series of modified copies of the initial multiple fault diagnosis.

A. Ranked Algorithm: Modified Copies of MFD Problem

In this approach, at each iteration, we solve a series of multiple fault diagnosis problems assuming that the states of some of the failure sources are known prior to diagnosis, i.e., some failure sources are known *good*, and some of them are known *bad* (definitely faulty). A similar approach has been considered by Murty [12] for determining a ranked set of solutions to assignment problems, and was recently enhanced by Cox et. al. [5] within the context of multi-target tracking. For simplicity, we represent the multiple fault diagnosis problem by four-tuple Γ=(MFD, G, B, X), where

1. MFD is the problem in (10)-(14),
2. $G \subseteq S$ represents the set of known good failure sources, i.e., for all $s_i \in G$, $x_i = 0$ (or $s_i \notin X$),
3. $B \subseteq S$ represents the set of definitely faulty failure sources, i.e., for all $s_i \in B$, $x_i = 1$ (or $s_i \in X$),
4. X is the optimal solution to the MFD problem subject to G and B.

Note that the number of unknown failure sources in Γ=(MFD, G, B, X) is m - $|G|$ - $|B|$. Initially, G and B are empty, i.e., $\Gamma^1 = (MFD, \emptyset, \emptyset, X^1)$. Subsequent solutions to Γ^1 are found by solving a succession of multiple fault diagnosis problems that are created from Γ^1 by a process called ***partitioning***. A problem, Γ, with the best solution X and size $m - |G| - |B|$, is partitioned into a set of subproblems, Γ_1, ..., $\Gamma_{m-|G|-|B|+1}$, such that:

- The union of the set of possible solutions to Γ_1 through $\Gamma_{m-|G|-|B|+1}$ is exactly the set of possible solutions to Γ,

- The sets of possible solutions to Γ_1 through $\Gamma_{m-|G|-|B|+1}$ are disjoint, and
- $\Gamma_{m-|G|-|B|+1}$ has only one solution X.

Let us assume that Γ^r is a dummy subproblem that is used to generate the subproblems Γ_1 through $\Gamma_{m-|G|-|B|+1}$ from Γ. The following procedure shows: (1) how to update the subproblem $\Gamma^r = (MFD, G^r, B^r, X^r)$, and (2) how to make subproblem $\Gamma_l = (MFD, G_l, B_l, X_l)$ form Γ^r for $l = 1, ..., m - |G| - |B|$, sequentially, and finally, (3) $\Gamma_{m-|G|-|B|+1} = \Gamma^r$. Initially, Γ^r=Γ. Then, for $l = 1, ..., m - |G| - |B|$, Γ^r is partitioned as follows:

- Select any $s_i \in S - (G^r \cup B^r)$,
- If $s_i \in X$, then $G_l \leftarrow G^r \cup \{s_i\}$ and $B^r \leftarrow B^r \cup \{s_i\}$, else $B_l \leftarrow B^r \cup \{s_i\}$ and $G^r \leftarrow G^r \cup \{s_i\}$.

Note that, at each iteration, the problem Γ^r is partitioned into two disjoint subproblems. This is because we force the subproblems to be different in the status of only one failure source s_i in the system, i.e., we add s_i to the set of definitely faulty failure sources in one subproblem, and to the set of known good failure sources in another subproblem. In addition, X cannot be a solution to Γ_l for $l = 1, ..., m - |G| - |B|$. Further more, Γ^r is the only subproblem which contains X and only X as its solution. This is because B^r=X and G^r=$S - X$.

As an illustration, let us consider a simple system with 3 failure sources $\{s_1, s_2, s_3\}$. In addition, let us assume that the optimal solution for the MFD problem in this case is X=$\{s_1\}$, i.e., $\Gamma^1 = (MFD, \emptyset, \emptyset, X^1 = \{s_1\})$. Therefore, the MFD problem can be partioned into the following subproblems; $\Gamma_1 = (MFD, G = \{s_1\}, B = \emptyset, X_1)$; $\Gamma_2 = (MFD, G = \emptyset, B = \{s_1, s_2\}, X_2)$, $\Gamma_3 = (MFD, G = \{s_2\}, B = \{s_1, s_3\}, X_3)$, and $\Gamma_4 = (MFD, G = \{s_2, s_3\}, B = \{s_1\}, X_4)$.

Therefore, we partition Γ^1 according to its best solution X^1, and place the resulting subproblems together with their best solutions, except the last one, i.e., $\Gamma_{m-|G|-|B|+1}$, on a priority queue of four-tuple (MFD, G, B, X). We then find a problem in the queue that has the best solution. The

solution of this problem is the second-best solution to the multiple fault diagnosis problem. Now, we remove this problem from the queue and replace it by its partitioning. The best solution found in the queue now is the third-best solution to the multiple fault diagnosis problem, and so on. The pseudocode for the L-ranked algorithm is shown in Figure 4.

Initialization: Find the first solution X^1 to MFD problem, and initialize a priority queue of four-tuple problems to contain only $\Gamma^1=(MFD, \emptyset, \emptyset, X^1)$. The top problem on this queue will always be the problem with the highest likelihood solution.

Step 1: Clear the list of solutions to be returned.

Step 2: DO until priority queue of problems is empty.

Step 2.1: Take the top problem $\Gamma=(MFD, G, B, X)$ off the queue.

Step 2.2: Add X to the list of solutions.

Step 2.3: If the cardinality of solution set is L, Stop.

Step 2.4: Let $\Gamma^r = \Gamma$,

Step 2.5: DO for $l = 1, ..., m - |G| - |B|$,

Step 2.5.1: Partition Γ^r into Γ^r and Γ' as follows:

Step 2.5.2: Select any $s_i \in S - (G^r \cup B^r)$,

Step 2.5.3: If $s_i \in X$, then $G' \leftarrow G^r \cup \{s_i\}$ and $B^r \leftarrow B^r \cup \{s_i\}$,

else $B' \leftarrow B^r \cup \{s_i\}$ and $G^r \leftarrow G^r \cup \{s_i\}$.

Step 2.5.4: Find the best solution X' to Γ'. If X' exists, add (MFD, G', B', X') to the queue.

END

END

Figure 4: Pseudocode for L-Rank MFD Algorithm

Since each subproblem is NP-hard, we use the near-optimal MFD algorithm of previous section to solve the ranked set problem near-optimally, i.e., X is a near-optimal solution for the problem Γ=(MFD, G, B, X). Thus, it is possible that l-th solution, i.e., X^l, has higher likelihood than the k-th solution, i.e., X^k, where $k > l$. Note that, we perform one partitioning for each of the L-best solution, in the worst case, each partitioning creates $O(m)$ new problems. This creates up to $O(Lm)$ multiple fault problems and insertions on the priority queue. Each problem takes at most $O(m|T_f|)$ time to solve near-optimally, and each insertion takes at most $O(\log(Lm))$ time. Therefore, the worst-case execution time of this approach is $O(Lm(m|T_f| + \log(Lm)))$, or approximately, $O(Lm^2|T_f|)$.

V. Multiple Fault Diagnosis with Repetitive Tests

A reasonable and common situation in unreliable testing is to apply a test several times to improve the confidence about a given hypothesis (a set of multiple fault candidates). For example, in order to reduce the probability of error, i.e., false alarm and missed detection of some faults (disorders or diseases), a system (a patient) may be tested multiple times, and because of imperfect nature of tests, the test results may be different. In this section, we assume that each test t_j has been applied n_j times in which it passed and failed μ_j and η_j times, respectively, i.e., $n_j = \mu_j + \eta_j$. **Note that applying a test at different times is equivalent to applying independent tests with** the same structure. In this case, let us assume that T_f and T_p denote the set of failed and passed tests (without any redundancy), respectively, and $T_f \cap T_p$ may not be empty. Thus, the problem is:

$$\max_{\underline{x},\underline{y}} \; J(\underline{x},\underline{y}) = \sum_{t_j \in T_f} \eta_j \ln(1 - y_j) + \sum_{i=1}^{m} x_i \{ \sum_{t_k \in T_p} \mu_k \ln(\frac{\overline{Pd}_{ik}}{\overline{Pf}_{ik}}) + \ln(p_i) \} \tag{20}$$

subject to (11)-(14). This problem is similar to the problem in (10). Thus, the algorithms in

previous sections can be readily applied to solve this problem. In this case: (1) in the first step of the MFD algorithm, $c_i(\underline{\lambda})$ is a function of μ_k for $k = 1, ..., |T_p|$, i.e., the number of time that test t_k passed, and (2) in the second step of the MFD algorithm, the optimum of the objective function with respect to $\underline{y}$ is replaced by $y_j^*(\lambda_j) = \frac{\lambda_j}{\eta_j + \lambda_j} u(\lambda_j)$ for $j = 1, ..., |T_f|$.

VI. Examples

Example 1: In this example, we consider: (1) a simple diagnostic problem with $m = 20$ failure sources (disorders) and $n = 20$ tests (manifests) which was used as an example in [3]; (Example 1.a - 1.d), and (2) a diagnostic problem with $m = 15$ failure sources and $n = 10$ tests from [9]; (Example 1.e). The false alarm probabilities for these systems are all zero, i.e., $Pf_{ij} = 0$ for $i = 1, ..., m$ and $j = 1, ..., n$ and T_p= T - T_f. Figures 5 and 6 show the failure source and detection probabilities for Example (1.a) through (1.d), and Example (1.e), respectively. Figures 7 and 8 show: (1) the set of failed tests T_f, (2) diagnostic results, (3) likelihood, (4) processing time and total number of runs to converge to the diagnostic results, (5) total processing time and total number of runs, and (6) approximate duality gap. The diagnostic results are based on the near-optimal multiple fault diagnosis algorithm in Figure 3. The processing times for these examples are obtained by running the MFD algorithm on a SPARC 10. Binglin et. al. [3] presented a genetic algorithm which required 10 minutes to find the set of diagnoses in Example 1.a with an IBM PS/2 Model 40 SX-20 MHZ microcomputer. These results show the superior performance of our algorithm compared to the algorithm in [3]. Miller et. al. [9] have not reported the processing time for Example 1.e. However, the largest problem that they considered contained 20 failure sources and 15 tests.

Example 2: In this example, we consider systems with: (1) m=n=100, m=n=500 and m=n=1000, (2) the probability of each failure source is set to a random number between (0.001, 0.5), (3) each test, on average, covers 5, 10 and 20 failure sources, (4) detection probabilities of a test associated with its covered failure sources are set to random numbers between (0,1), (5) the false alarm prob-

$p(s_1)= 0.170$	$p(s_2)= 0.070$	$p(s_3)= 0.030$	$p(s_4)= 0.120$	$p(s_5)= 0.135$
$p(s_6)= 0.180$	$p(s_7)= 0.075$	$p(s_8)= 0.030$	$p(s_9)= 0.140$	$p(s_{10})= 0.050$
$p(s_{11})= 0.026$	$p(s_{12})= 0.014$	$p(s_{13})= 0.054$	$p(s_{14})= 0.060$	$p(s_{15})= 0.003$
$p(s_{16})= 0.023$	$p(s_{17})= 0.048$	$p(s_{18})= 0.079$	$p(s_{19})= 0.098$	$p(s_{20})= 0.027$
$Pd_{1,2}= 0.06$	$Pd_{1,4}= 0.68$	$Pd_{1,6}= 0.10$	$Pd_{1,7}= 0.51$	$Pd_{2,1}= 0.53$
$Pd_{2,3}= 0.81$	$Pd_{2,4}= 0.09$	$Pd_{2,5}= 0.85$	$Pd_{2,8}= 0.13$	$Pd_{2,9}= 0.34$
$Pd_{2,10}= 0.85$	$Pd_{3,2}= 0.54$	$Pd_{3,5}= 0.45$	$Pd_{3,6}= 0.90$	$Pd_{3,7}= 0.59$
$Pd_{3,10}= 0.29$	$Pd_{4,2}= 0.74$	$Pd_{4,5}= 0.52$	$Pd_{4,7}= 0.65$	$Pd_{4,9}= 0.32$
$Pd_{5,3}= 0.72$	$Pd_{5,8}= 0.49$	$Pd_{6,3}= 0.09$	$Pd_{6,5}= 0.66$	$Pd_{6,10}= 0.44$
$Pd_{7,3}= 0.22$	$Pd_{7,4}= 0.46$	$Pd_{7,5}= 0.21$	$Pd_{7,6}= 0.76$	$Pd_{7,10}= 0.43$
$Pd_{8,1}= 0.29$	$Pd_{8,2}= 0.34$	$Pd_{8,8}= 0.25$	$Pd_{9,1}= 0.39$	$Pd_{9,4}= 0.20$
$Pd_{9,5}= 0.90$	$Pd_{9,6}= 0.48$	$Pd_{9,7}= 0.38$	$Pd_{10,2}= 0.74$	$Pd_{10,8}= 0.27$
$Pd_{11,12}= 0.31$	$Pd_{11,14}= 0.85$	$Pd_{11,17}= 0.30$	$Pd_{12,17}= 0.50$	$Pd_{13,12}= 0.29$
$Pd_{13,14}= 0.64$	$Pd_{13,15}= 0.15$	$Pd_{13,16}= 0.11$	$Pd_{13,18}= 0.72$	$Pd_{13,19}= 0.62$
$Pd_{14,13}= 0.88$	$Pd_{14,18}= 0.27$	$Pd_{15,11}= 0.72$	$Pd_{15,12}= 0.92$	$Pd_{15,14}= 0.07$
$Pd_{15,15}= 0.47$	$Pd_{15,19}= 0.73$	$Pd_{15,20}= 0.96$	$Pd_{16,16}= 0.32$	$Pd_{16,17}= 0.26$
$Pd_{17,11}= 0.05$	$Pd_{17,12}= 0.80$	$Pd_{17,15}= 0.73$	$Pd_{17,18}= 0.58$	$Pd_{18,12}= 0.04$
$Pd_{18,14}= 0.26$	$Pd_{18,16}= 0.23$	$Pd_{18,17}= 0.69$	$Pd_{18,20}= 0.51$	$Pd_{19,12}= 0.12$
$Pd_{19,17}= 0.95$	$Pd_{19,20}= 0.67$	$Pd_{20,14}= 0.43$	$Pd_{20,15}= 0.18$	$Pd_{20,16}= 0.11$

Figure 5: Probabilities for Example 1.a-1.d

$p(s_1)= 0.12$	$p(s_2)= 0.14$	$p(s_3)= 0.39$	$p(s_4)= 0.64$	$p(s_5)= 0.01$
$p(s_6)= 0.21$	$p(s_7)= 0.26$	$p(s_8)= 0.19$	$p(s_9)= 0.59$	$p(s_{10})= 0.23$
$p(s_{11})= 0.06$	$p(s_{12})= 0.47$	$p(s_{13})= 0.56$	$p(s_{14})= 0.41$	$p(s_{15})=0.06$
$Pd_{1,1}= 0.58$	$Pd_{1,4}= 0.43$	$Pd_{1,5}= 0.46$	$Pd_{1,6}= 0.91$	$Pd_{1,7}= 0.9$
$Pd_{2,4}= 0.67$	$Pd_{2,5}= 0.1$	$Pd_{2,7}= 0.94$	$Pd_{2,10}= 0.63$	$Pd_{3,3}= 0.44$
$Pd_{3,4}= 0.79$	$Pd_{3,7}= 0.07$	$Pd_{3,8}= 0.14$	$Pd_{3,9}= 0.13$	$Pd_{3,10}= 0.07$
$Pd_{4,5}= 0.58$	$Pd_{4,7}= 0.28$	$Pd_{4,8}= 0.17$	$Pd_{4,9}= 0.12$	$Pd_{4,10}= 0.75$
$Pd_{5,4}= 0.26$	$Pd_{5,8}= 0.24$	$Pd_{5,9}= 0.17$	$Pd_{5,10}= 0.12$	$Pd_{6,1}= 0.25$
$Pd_{6,2}= 0.15$	$Pd_{6,4}= 0.72$	$Pd_{6,6}= 1.00$	$Pd_{6,9}= 0.04$	$Pd_{7,2}= 0.81$
$Pd_{7,4}= 0.07$	$Pd_{7,5}= 0.46$	$Pd_{7,6}= 0.28$	$Pd_{7,8}= 0.3$	$Pd_{8,1}= 0.96$
$Pd_{8,3}= 0.11$	$Pd_{8,10}= 0.45$	$Pd_{9,2}= 0.32$	$Pd_{9,5}= 0.57$	$Pd_{9,7}= 0.97$
$Pd_{9,8}= 0.26$	$Pd_{9,9}= 0.97$	$Pd_{10,3}= 0.97$	$Pd_{10,5}= 0.4$	$Pd_{10,10}= 0.88$
$Pd_{11,1}= 0.85$	$Pd_{11,2}= 0.36$	$Pd_{11,3}= 0.64$	$Pd_{11,4}= 0.84$	$Pd_{11,10}= 0.21$
$Pd_{12,2}= 0.77$	$Pd_{12,5}= 0.51$	$Pd_{12,7}= 0.91$	$Pd_{12,8}= 0.05$	$Pd_{13,1}= 0.74$
$Pd_{13,2}= 0.3$	$Pd_{13,3}= 0.63$	$Pd_{13,7}= 0.48$	$Pd_{13,9}= 0.43$	$Pd_{13,10}= 0.45$
$Pd_{14,2}= 0.61$	$Pd_{14,3}= 0.85$	$Pd_{14,4}= 0.42$	$Pd_{14,5}= 0.97$	$Pd_{14,7}= 0.23$
$Pd_{14,9}= 0.08$	$Pd_{14,10}= 0.86$	$Pd_{15,1}= 0.38$	$Pd_{15,4}= 0.64$	$Pd_{15,6}= 0.12$
$Pd_{15,7}= 0.72$	$Pd_{15,10}= 0.19$			

Figure 6: Probabilities for Example 1.e

Ex.	$\{j \mid t_j \in T_f\}$	$\{s_i \mid s_i \in X\}$	Prob$(X \mid T_f, T_p)$
1.a	$\{1, 2, 4, 5, 7, 8, 13, 15\}$	$\{1, 9, 10, 14, 17\}$	$3.66e^{-09}$
1.b	$\{7, 8, 9, 11, 14, 15\}$	$\{4, 5, 17, 20\}$	$1.32e^{-10}$
1.c	$\{1, 3, 4, 6, 7, 11, 13, 15, 16\}$	$\{1, 5, 9, 14, 16, 17\}$	$6.82e^{-13}$
1.d	$\{1, 2, 3, 7, 8, 12, 13, 17\}$	$\{4, 5, 8, 14, 19\}$	$2.49e^{-09}$
1.e	$\{1, 2, 4, 5, 7, 8, 9, 10\}$	$\{3, 4, 9, 12, 13\}$	$7.77e^{-02}$

Figure 7: *MFD* Algorithm Results for Examples 1.a-1.e

Ex.	Convergence		Total		Approximate
	# Runs	Time (sec)	# Runs	Time (sec)	Duality Gap
1.a	8	0.170	58	0.310	4.68%
1.b	2	0.009	65	0.240	4.76%
1.c	2	0.050	68	0.340	4.69%
1.d	1	0.004	64	1.83	4.69%
1.e	2	0.007	58	0.15	4.52%

Figure 8: *MFD* Algorithm Results for Examples 1.a-1.e

abilities are assumed to be zero, and (6) the number of failed tests are 5, 10 and 20. Figures 9, 10 and 11 show the simulation results for these systems. Each row of these Figures represents the average of simulation results for 5 randomly generated systems. Note that, in most of the cases, the average approximate duality gaps are around 5%. However, in some of the cases, for example, the last row of Figure 9, the approximate duality gap is very large, i.e., 22.15%. In order to improve the solution (or, equivalently, approximate duality gap), we can apply the L-ranked algorithm. The average approximate duality gap based on 2-ranked algorithm for the last set of systems in Figure 9 reduces to 1.49%.

Average Test Coverage	$\|T_f\|$	Convergence		Total		Approximate Duality Gap
		# Runs	Time (sec)	# Runs	Time (sec)	
5	5	2	0.10	62	2.70	4.25%
5	10	16	0.38	63	3.11	3.75%
5	20	4	0.83	70	13.78	4.79%
10	5	3	0.25	63	3.35	6.75%
10	10	12	2.91	60	12.00	5.26%
10	20	9	3.01	83	31.00	9.59%
20	5	1	0.11	55	6.97	6.12%
20	10	2	0.85	58	28.81	10.96%
20	20	16	23.39	59	90.72	22.15%

Figure 9: Simulation Results for m=n=100

Example 3: In this example, we consider three systems with 10 failure sources and 10 tests as in [15]. The false alarm probabilities are assumed to be zero. The simulation results for 2^{10} possible combinations of test results are shown in Figure 15. The second column shows the number of correct

Average Test Coverage	$\|T_f\|$	Convergence		Total		Approximate Duality Gap
		# Runs	Time (sec)	# Runs	Time (sec)	
5	5	1	0.54	62	23.56	4.15%
5	10	3	1.62	72	44.58	3.96%
5	20	16	7.15	66	32.77	4.23%
10	5	1	0.48	58	26.99	4.82%
10	10	1	1.20	69	46.68	4.60%
10	20	15	19.25	67	85.37	6.60%
20	5	1	0.56	51	30.41	3.75%
20	10	6	6.55	64	60.38	3.00%
20	20	11	26.19	64	184.06	16.03%

Figure 10: Simulation Results for m=n=500

cases out of 1024 possible combinations of test results. The third column shows the weighted probability of correct cases. The columns corresponding to N_d and N_f denote the unweighted probabilities of detection, i.e., the unweighted probability of common faulty failure sources in the optimal and near-optimal solutions, and false alarm, i.e., the unweighted probability of faulty failure sources in the near-optimal solution and not in the optimal solution. Figure 16 shows the simulation results based on the 2-ranked algorithm. The average weighted (unweighted) accuracy based on the MFD algorithm and 2-ranked algorithm are 97.71% (94.99%) and 99.96%(99.77%), respectively.

Example 4: In this example, we consider the medical example in [14, 19]. The system under consideration is for neuropsychiatric diagnosis. The system consists of 26 disorders (failure sources) from psychiatry and neurology which affect mental status. A list of 56 symptoms (tests) and signs was assembled for each disorder. There are 384 links in the system, each of which connects a

Average Test Coverage	$\|T_f\|$	Convergence		Total		Approximate Duality Gap
		# Runs	Time (sec)	# Runs	Time (sec)	
5	5	2	3.39	67	102.09	5.89%
5	10	2	3.53	73	122.88	5.12%
5	20	2	5.02	67	138.56	4.59%
10	5	1	2.27	54	87.95	4.09%
10	10	2	3.83	55	99.53	4.27%
10	20	5	11.90	66	169.60	4.99%
20	5	1	3.26	53	103.29	4.28%
20	10	11	30.29	57	137.15	4.94%
20	20	28	139.14	76	374.54	9.42%

Figure 11: Simulation Results for m=n=1000

disorder to a manifestation. Similar to [14], five groups of test cases are used to test the MFD and L-ranked algorithms. Manifestations are chosen randomly from the total set of 56 possible manifestations based on a uniform distribution. Each group of test cases consists of ten different sets of manifestations. Each case in the first test group has one present manifestation (failed test); each case in the other groups have 3, 5, 7 and 9 manifestations. If any randomly generated test result is inconsistent with the causal network, the case is discarded and a new one is generated. The inconsistent test results may occur because the causal network used in the experiment has some perfect tests, i.e., $Pd_{ij} = 1$ and $Pf_{ij} = 0$. Thus, after applying the first Lemma, and reducing the size of the problem, the second Lemma may not be satisfied, i.e., there exists a failed test that is not covered by any failure source. Simulation results show that among all 50 cases MFD and 2-ranked algorithms generate 98% and 100% optimal solutions. Peng and Reggia applied their

$p(s_1)= 0.026$	$p(s_2)= 0.014$	$p(s_3)= 0.054$	$p(s_4)= 0.060$	$p(s_5)= 0.003$
$p(s_6)= 0.023$	$p(s_7)= 0.048$	$p(s_8)= 0.079$	$p(s_9)= 0.098$	$p(s_{10})= 0.027$
$Pd_{1,2}= 0.31$	$Pd_{1,4}= 0.85$	$Pd_{1,7}= 0.30$	$Pd_{2,7}= 0.50$	$Pd_{3,2}= 0.29$
$Pd_{3,4}= 0.64$	$Pd_{3,5}= 0.15$	$Pd_{3,6}= 0.11$	$Pd_{3,8}= 0.72$	$Pd_{3,9}= 0.62$
$Pd_{4,3}= 0.88$	$Pd_{4,8}= 0.27$	$Pd_{5,1}= 0.72$	$Pd_{5,2}= 0.32$	$Pd_{5,4}= 0.07$
$Pd_{5,5}= 0.47$	$Pd_{5,9}= 0.73$	$Pd_{5,10}= 0.96$	$Pd_{6,6}= 0.32$	$Pd_{6,7}= 0.26$
$Pd_{7,1}= 0.05$	$Pd_{7,2}= 0.80$	$Pd_{7,5}= 0.73$	$Pd_{7,8}= 0.58$	$Pd_{8,2}= 0.04$
$Pd_{8,4}= 0.26$	$Pd_{8,6}= 0.23$	$Pd_{8,7}= 0.69$	$Pd_{8,10}= 0.51$	$Pd_{9,2}= 0.12$
$Pd_{9,7}= 0.95$	$Pd_{9,10}= 0.67$	$Pd_{10,4}= 0.43$	$Pd_{10,5}= 0.18$	$Pd_{10,6}= 0.11$

Figure 12: Failure Source and Detection Probabilities for Example 3.a

$p(s_1)= 0.170$	$p(s_2)= 0.070$	$p(s_3)= 0.030$	$p(s_4)= 0.120$	$p(s_5)= 0.135$
$p(s_6)= 0.180$	$p(s_7)= 0.075$	$p(s_8)= 0.030$	$p(s_9)= 0.140$	$p(s_{10})= 0.050$
$Pd_{1,2}= 0.06$	$Pd_{1,4}= 0.68$	$Pd_{1,6}= 0.10$	$Pd_{1,7}= 0.51$	$Pd_{2,1}= 0.53$
$Pd_{2,3}= 0.81$	$Pd_{2,4}= 0.09$	$Pd_{2,5}= 0.85$	$Pd_{2,8}= 0.13$	$Pd_{2,9}= 0.34$
$Pd_{2,10}= 0.85$	$Pd_{3,2}= 0.54$	$Pd_{3,5}= 0.45$	$Pd_{3,6}= 0.90$	$Pd_{3,7}= 0.59$
$Pd_{3,10}= 0.29$	$Pd_{4,2}= 0.74$	$Pd_{4,5}= 0.52$	$Pd_{4,7}= 0.65$	$Pd_{4,9}= 0.32$
$Pd_{5,3}= 0.72$	$Pd_{5,8}= 0.49$	$Pd_{6,3}= 0.09$	$Pd_{6,5}= 0.66$	$Pd_{6,10}= 0.44$
$Pd_{7,3}= 0.22$	$Pd_{7,4}= 0.46$	$Pd_{7,5}= 0.21$	$Pd_{7,6}= 0.76$	$Pd_{7,10}= 0.43$
$Pd_{8,1}= 0.29$	$Pd_{8,2}= 0.34$	$Pd_{8,8}= 0.25$	$Pd_{9,1}= 0.39$	$Pd_{9,4}= 0.20$
$Pd_{9,5}= 0.90$	$Pd_{9,6}= 0.48$	$Pd_{9,7}= 0.38$	$Pd_{10,2}= 0.74$	$Pd_{10,8}= 0.27$

Figure 13: Failure Source and Detection Probabilities for Example 3.b

$p(s_1)= 0.34$	$p(s_2)= 0.14$	$p(s_3)= 0.06$	$p(s_4)= 0.24$	$p(s_5)= 0.27$
$p(s_6)= 0.36$	$p(s_7)= 0.30$	$p(s_8)= 0.06$	$p(s_9)= 0.28$	$p(s_{10})= 0.10$
$Pd_{1,2}= 0.06$	$Pd_{1,4}= 0.68$	$Pd_{1,6}= 0.10$	$Pd_{1,7}= 0.51$	$Pd_{2,1}= 0.53$
$Pd_{2,3}= 0.81$	$Pd_{2,4}= 0.09$	$Pd_{2,5}= 0.85$	$Pd_{2,8}= 0.13$	$Pd_{2,9}= 0.34$
$Pd_{2,10}= 0.85$	$Pd_{3,2}= 0.54$	$Pd_{3,5}= 0.45$	$Pd_{3,6}= 0.90$	$Pd_{3,7}= 0.59$
$Pd_{3,10}= 0.29$	$Pd_{4,2}= 0.74$	$Pd_{4,5}= 0.52$	$Pd_{4,7}= 0.65$	$Pd_{4,9}= 0.32$
$Pd_{5,3}= 0.72$	$Pd_{5,8}= 0.49$	$Pd_{6,3}= 0.09$	$Pd_{6,5}= 0.66$	$Pd_{6,10}= 0.44$
$Pd_{7,3}= 0.22$	$Pd_{7,4}= 0.46$	$Pd_{7,5}= 0.21$	$Pd_{7,6}= 0.76$	$Pd_{7,10}= 0.43$
$Pd_{8,1}= 0.29$	$Pd_{8,2}= 0.34$	$Pd_{8,8}= 0.25$	$Pd_{9,1}= 0.39$	$Pd_{9,4}= 0.20$
$Pd_{9,5}= 0.90$	$Pd_{9,6}= 0.48$	$Pd_{9,7}= 0.38$	$Pd_{10,2}= 0.74$	$Pd_{10,8}= 0.27$

Figure 14: Failure Source and Detection Probabilities for Example 3.c

Example	Correct cases		N_d	N_f
	# Times (out of 1024)	Weighted		
3.a	992 (96.88%)	99.91%	98.63%	0.39%
3.b	971 (94.82%)	98.61%	97.60%	0.31%
3.c	955 (93.26%)	94.61%	97.66%	0.37%

Figure 15: MFD Alg. Results for Examples 3.a-3.c

Example	Correct cases		N_d	N_f
	# Times (out of 1024)	Weighted		
3.a	1024 (100%)	100%	100%	0.00%
3.b	1019 (99.51%)	99.92%	99.77%	0.06%
3.c	1022 (99.80%)	99.95%	99.92%	0.03%

Figure 16: 2-ranked Alg. Results for Examples 3.a-3.c

competition-based connectionist methods to this causal network. Their algorithm generated 74% of globally optimal solutions, and 90% of one of the three globally optimal solutions.

VII. Conclusion

In this paper, we considered the problem of constructing optimal and near-optimal multiple fault diagnosis in bipartite systems with unreliable (imperfect) tests. We presented a multiple fault diagnosis algorithm based on Lagrangian relaxation and subgradient optimization method, which provides near optimal solutions for the multiple fault diagnosis, and upper bounds for an optimal branch-and-bound algorithm. Computational results indicate that our algorithm can be used in systems with as many as 1000 faults. In addition, we presented an algorithm to generate the set of L-ranked multiple fault candidates. In this algorithm, we find the most likely candidate using the near optimal multiple fault diagnosis algorithm. Then, we partition the problem, based on the first solution, to a set of disjoint subproblems. The solutions to these subproblems with the highest likelihood represents the second most likely candidates. This procedure is continued until L-ranked multiple fault diagnoses are found, or no more feasible solutions exist. We showed that the computational complexity of this approach is $O(Lm^2|T_f|)$, and therefore, applicable for systems with as many as 1000 faults and tests. Finally, we extended the multiple fault diagnosis problem to redundant or repetitive tests. In this case, the problem is very similar to the original multiple fault diagnosis problem, and therefore, the MFD algorithm can be extended to this problem as well.

In this paper, we assumed that the test results are known prior to diagnosis. That is, we considered the problem of multiple fault *diagnosis* with unreliable tests. The problem of *sequential* multiple fault diagnosis *strategy (testing)* with unreliable tests is an important problem in field maintenance. Furthermore, the order of partitioning in the L-ranked algorithm may improve the accuracy of the near-optimal solutions. We expect to investigate these challenging issues in our future efforts.

Acknowledgment

We would like to thank Yun Peng, Jim Reggia and Jonathan Wald for allowing us to use the data for the medical application example.

References

[1] J.E. Beasley. A Lagrangian heuristic for set-covering problems. *Naval Research Logistics*, 37:151–163, 1990.

[2] J.E. Beasley. Enhancing an algorithm for set covering problems. *European Journal of Operational Research*, 58:293–300, 1992.

[3] Z. Binglin, Y. Tinghu, and H. Ren. A genetic algorithm for diagnosis problem solving,. *Conference Proceedings. International Conference on Systems, Man and Cybernetics*, 2:404–408, 1993.

[4] G.F. Cooper. The computational complexity of probabilistic inference using bayesian belief networks. *Artificial Intelligence*, 42:393–405, 1990.

[5] I.J. Cox and M.L. Miller. On finding ranked assignments with application to multi-target tracking and motion correspondence. *IEEE Transactions on Aerospace and Electronic Systems*, 32(1):486–489, 1995.

[6] M.L. Fisher. The Lagrangian relaxation method for solving integer programming problems. *Management Science*, 27(1):1–18, 1981.

[7] M.L. Fisher and P. Kedia. Optimal solution of set covering/partitioning problems using dual heuristics. *Management Science*, 36:674–688, 1990.

[8] C.A. Floudas. *Nonlinear and Mixed-Integer Optimization, Fundamentals and applications.* Oxford University Press, 1995.

[9] J.A. Miller, W.D. potter, R.V. Gandham, and C.N. Lapena. An evaluation of local improvement operators for genetic algorithms. *IEEE Transaction on Systems, Man, and Cybernetics*, 23(5):1340–1351, September/October 1993.

[10] R.A. Miller, M.A. McNeil, S.M. Challinor, F.E. Masarie, and J.D. Myers. The internist-1/quick medical reference project. *West. J. Med. 145*, pages 816–822, 1986.

[11] R.E. Miller and J.W. Thatcher. *Complexity of Computer Computations*. Plenum Press, 1972.

[12] K.G. Murty. An algorithm for ranking all the assignments in order of increasing cost. *Operations Research*, 16:682–687, 1968.

[13] G.L. Nemhauser and Wolsey L.A. *Integer and Combinatorial Optimization*. John Wiley and Sons, 1988.

[14] Y. Peng and J.A. Reggia. *Abductive Inference Models for Diagnostic Problem-Solving*. Springer-Verlag, 1990.

[15] Y. Peng and J.A. Reggia. A connectionist model for diagnostic problem solving. *IEEE Transaction on Systems, Man, and Cybernetics*, 19(2):285–298, March/April 1989.

[16] Y. Peng and J.A. Reggia. A probabilistic causal model for diagnostic problem solving, part II: Diagnostic strategy. *IEEE Transaction on Systems, Man, and Cybernetics*, 17(3):395–406, May/June 1987.

[17] A. Schrijver. *Theory of Linear and Integer Programming*. John Wiley and Sons, 1986.

[18] M.M. Syslo, N. Deo, and J.S. Kowalik. *Discrete Optimization Algorithms*. Prentice-Hall, Inc, 1983.

[19] J. Wald, M. Farach, M. Tagamets, and J. Reggia. Generating plausible diagnostic hypotheses with self processing causal networks. *Journal of Experimental and Theoretical AI*, 35:91–112, 1991.

[20] T.D. Wu. A problem decomposition method for efficient diagnosis and interpretation of multiple disorderss. *Computer Methods and Programs in Biomedicine*, 35:239–250, 1991.