RELIABILITY DIAGNOSTIC STRATEGIES FOR SERIES SYSTEMS UNDER IMPERFECT TESTING

by

Susan R. Reller

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

in

Industrial Engineering and Operations Research

APPROVED:

J. A. Nachlas, Chairman	B. S. Blanchard
T. R. Rakes	S. C. Sarin

September, 1987

Blacksburg, Virginia

RELIABILITY DIAGNOSTIC STRATEGIES FOR SERIES SYSTEMS UNDER IMPERFECT TESTING

by

Susan R. Reller

Committee Chairman: Joel A. Nachlas
Industrial Engineering and Operations Research

(ABSTRACT)

An expected cost model was developed for failure detection in series systems under imperfect testing. Type I and type II error probabilities are included and single-pass sample paths are required. The model accounts for the expected costs of testing components, false positive termination, and no-defect-found outcomes.

Based on the model, a heuristic was developed to construct the cost minimizing testing sequence. The heuristic algorithm utilizes elementary arithmetic computations and has been successfully applied to a variety of problems. Furthermore, the algorithm appears to be globally convergent. Choice of a starting solution affects the rate of convergence, and guidelines for selecting the starting solution were discussed. Implementation of the heuristic was illustrated by numerical example.

Acknowledgements

With sincere thanks, I acknowledge Dr. Joel Nachlas, my thesis advisor, for his never-failing enthusiasm and optimism, the Office of Naval Research, for their generous financial support, and, my family, for their love and encouragement. Without these people, my studies would not have been as enjoyable.

Table of Contents

List of Tables

Chapter 1

Introduction

Systems often fail without symptoms suggesting the cause. Electronic component configurations are particularly susceptible to these blind failure modes. In a series structure, system failure corresponds to failure of one of the constituent components. The failed component must be identified and repaired to return the system to operation. This may be accomplished by testing components, one at a time, until the fault is located. If testing stops once the failed component is found, all components need not be tested, and test order becomes an important design variable.

The total diagnostic cost is functionally related to characteristics of the system components. Each component has a conditional causative failure probability, the probability that it has failed given that the system is down. Also, a cost is incurred with each component test. In basic models, one can use this data to define a testing sequence that minimizes the expected total cost of the diagnostic scheme. This problem has been approached several ways [2-7, 9-10].

Most of the pertinent literature considers only the perfect- testing case. Departing from the literature, this paper allows imperfect information. If the test returns a positive failure reading when the component has failed, or a negative reading when it has not, a correct diagnosis has been made. However, under the assumption of imperfect testing there is some probability of an incorrect diagnosis. A type I, false positive, error occurs when the test returns a positive failure reading for a component which has not failed. Similarly, a type II, false negative, error occurs when the test yields a negative reading for a component which has failed. Distinct false positive and false negative error probabilities are associated with each component test.

Because the incorrect diagnoses imply misallocation of test and repair resources, each has an associated penalty. Suppose that testing terminates on a false positive reading. Then, repair action is attempted on a component which has not failed, and the system remains inoperative since the true fault is still undetected. The related penalty reflects the unnecessary repair effort. Alternatively, suppose that one false negative and no false positive readings are recorded. Then all components will be tested but no fault found. This is called an NDF, no-defect- found, outcome. Subsequent actions lead to a different penalty. No penalty is associated with a single false negative reading per se. The false negative must either be followed by a false positive reading or result in an NDF outcome. In addition to the direct cost penalties, test error probabilities also affect the expected testing cost. False negative probabilities tend to inflate the expected cost while false positive probabilities tend to diminish it.

The direct and indirect costs of imperfect testing must all be included in an expected value model of the diagnostic testing program. Such a model is developed in this paper. Analysis of the model suggests efficient methods for constructing the cost minimizing testing sequence.

Chapter 2

Literature Review

Development of optimal sequences for diagnostic testing has been addressed frequently in the literature. As already noted, these papers generally assume that tests yield perfect information. Although the various models differ in system structure assumptions and underlying probability definitions, their solutions follow a common pattern.

Butterworth [4] presents two fault-testing models for k-of-n systems. The series structure, of interest in this paper, is the special case for which k equals n. One model applies when the system has failed and tests are required to determine the state of each component. Because this model allows simultaneous component failures, even in the series case, it is less relevant than the alternate model in which component tests are required to determine the state of the system. Here, tests are sequentially run until k functional or n-k+1 failed components are found. For the series system, the author shows that the optimal sequential policy tests components in order of decreasing P_i/C_i where P_i is the probability that component i has failed and C_i is the cost of testing component i. Conceptually, this policy tests first the component having the greatest probability, per unit cost, of terminating testing. Although Butterworth uses component tests to determine system status while other authors assume system failure, this difference is reflected only in definition of the component failure probabilities. The intuitive sequencing rule holds for all of the models discussed.

In analyzing his models, Butterworth identifies two solution types: sequential and nonsequential. If the optimal sequence is completely specified before testing begins, the procedure is sequential. Conversely, if the optimal sequence is adjusted at each stage of testing to reflect outcomes of earlier tests, the policy is nonsequential. In certain cases Butterworth shows that the optimal procedure must be nonsequential. However, Butler and Lieberman [3] argue that the optimal policy for a series system, having a single failed component, must be sequential. With perfect information, at any testing stage, all previously tested components are functional. Therefore, the information available at any stage may be prespecified and the overall testing sequence predetermined. Their argument is not entirely convincing, however, when tests yield imperfect information. Nonetheless, under the governing assumptions of the model presented, this paper seeks a sequential solution to the imperfect testing problem.

More importantly, Butler and Lieberman [3] address the general coherent complex system of which the k-of-n system is but one example. Recognizing that optimal sequences for complex systems are likely to be nonsequential and difficult to define, they propose a heuristic sequencing approach. The conditional failure probability given system failure is computed for each component, and tests are ordered by decreasing failure probabilities. Most of their work focuses on strategies for computing the failure probabilities, but their heuristic resembles Butterworth's formal solution.

Gluss [7] specifies optimal fault-detecting policies for complex systems of modules and subcomponents. In his first model, top-level tests are run to locate the failed module. Then, tests are made of subcomponents in the failed module. At the module level, there are a priori failure probabilities and testing costs. The optimal sequence tests in order of decreasing P/C--the common solution. A similar result holds at the subcomponent level. In Gluss' second model, overall tests of modules are not possible and a penalty is incurred whenever the search moves from one module to another. This model is less relevant to the work presented here.

Chu [5] constrains the problem by considering complex systems which fail with observable symptoms corresponding to sets of possible malfunctions. His objective, to identify the active malfunction, is somewhat broader than locating a single failed component. However, he assumes that the malfunctions are disjoint and mutually exclusive; therefore, the analogy is reasonable. Using a priori probability data, he derives probabilities for malfunctions conditioned on symptoms. His first model, the case of perfect testing, yields an optimal solution, based on the malfunction probabilities and testing costs, comparable to the common pattern.

Models also have been developed strictly for series systems. Though less general, these more directly apply to the problem of interest. Without assuming a priori knowledge, but based on the component lifetime distribution, Park [11] computes the probability that a given component has failed conditioned on system failure. He then shows that the optimal test sequence is arranged in order of decreasing probability-to-cost ratio. However, by his probability construction, he is forced to make the simplifying, and restricting, assumption that component hazard functions are constant or that the exact ages of components at the time of system failure are known.

Nachlas and Binney [10] remove Parks' restriction by deriving, from system and component reliability characteristics, the probability that any component in the series system has caused system failure. Their causative failure probabilities apply quite generally. Specifically, the derived probabilities can be used even when the underlying component hazard functions are not constant or when the ages of components at the time of system failure are not precisely known. The Nachlas and Binney conditional causative failure probabilities are adopted throughout the work developed here. Using the failure probabilities and testing costs, the authors establish an optimal sequence which is identical to the common rule.

The imperfect testing case has received less attention in the literature. Kumar and Kapoor [8] discuss failure classification for a two-unit series system and include the costs associated with mis-classification. However, failure classification is based on known component lifetime distributions and does not involve testing of components. Therefore, the problem they consider is not optimal sequencing of component tests, but rather, failure classification to minimize the costs of mis-classification.

Extending his initial model, Chu [5] incorporates imperfect malfunction detection. For each malfunction type, he introduces the probability of detecting the malfunction when it is present. This is the complement of the overlook probability encountered in the classic search problem. His model compensates for the possibility of incorrect diagnosis by a scheme of repeated testing. From search theory, if the malfunction is to be found with certainty, an infinite number of tests are required. In practical implementation, however, the number of test cycles must be constrained. Therefore, Chu considers only sequences in which diagnostic tests are repeated a finite number of times.

By Chu's interpretation, the diagnostic sequencing problem is an application of the standard search problem with overlook probabilities. Although the pure search solution, when overlook probabilities are non-trivial, requires an infinite sequence of searches, Matula [9] shows conditions under which the optimal sequence, though infinite, is periodic. That is, after some number of tests, the sequence repeats itself from the beginning. So, were the diagnostic sequencing problem to be cast as a standard search problem under appropriate conditions, a manageable solution would be suggested by Matula's work.

Although the classical search problem has received wide attention, it misses essential elements of the diagnostic sequencing problem. Imperfect testing is reflected in the overlook probabilities, but this accounts for only the type II, false negative, component of test error. The type I error probability must also be included. Furthermore, repeated imperfect testing, whether periodic or not, may not be effective when applied to physical systems. Consequently, classic search models are generally inadequate for the purpose intended here.

In an extension of earlier work, Firstman and Gluss [6] modify Gluss' Model I [7] to include both type I and type II test errors. Under this scenario, a system of modules and subcomponents is subjected first to module-level screening and then to subcomponent screening. At both levels there are probabilities of false positive and false negative test errors. Testing proceeds until the failed subcomponent is correctly identified. If a false positive reading results at the module level, an unnecessary subcomponent search through that module is conducted after which module-level testing resumes until the fault is correctly located. If a false negative reading occurs at the module level, then all other modules are unnecessarily tested after which the failed module is retested. By assumption, on the retest a correct diagnosis is made. Penalties associated with test errors at the subcomponent level are similar to these. The model's key characteristic is the cyclic nature of the diagnostic scheme in the event of a test error: errors are discovered in the course of testing by application of perfect checkout tests, and testing does not terminate until the true fault is identified.

At each level, the optimal solution orders by decreasing failure probability-to-cost ratio where the cost function measures both the cost of testing and the cost of incurring a false positive error. The false negative penalty does not factor into the sequencing criterion because, being the added cost of a complete test cycle, it is the same regardless of which item has actually failed.

Although Firstman and Gluss present a cost-minimizing solution for imperfect testing, their assumptions are fundamentally different from those governing the model proposed here. Aside from the question of perfect retests, the primary discrepancy concerns the diagnostic test sample paths. Because the authors use cyclic testing with perfect rechecks, they disallow false positive test termination and NDF outcomes. In practice, with electronic components, test errors, positive or negative, often recur when inefficient tests are replicated. Therefore, repeated testing may be technologically impractical. When tests are subject to error and repeated testing economically or technically infeasible, either of Firstman and Gluss' disallowed outcomes is possible. To cover those outcomes, the diagnostic scheme assumed here allows only one pass through the system components. Testing stops when a positive reading (correct or not) has been recorded or when all components have been tested and no defect found (an NDF outcome). Given these sample path assumptions, the model developed measures the expected total cost of the diagnostic program.

Chapter 3

Model Development

The expected total cost of the diagnostic program is the sum of the expected cost of testing components, the expected cost of NDF outcomes, and the expected cost of false positive termination. When these quantities are expressed algebraically, the resulting mathematical model can be analyzed as a tool for constructing the optimal testing sequence. The following notation is used to develop the expected value model.

P_i = Pr{component i failed / system failed}

These probabilities, defined by Nachlas and Binney [10], constitute a proper density function.

a_i = Pr{positive reading on component i / i not failed}

B_i = Pr{negative reading on component i / i failed}

C_i = Cost of testing component i

D_1 = Cost penalty for NDF outcome

D_2 = Cost penalty for false positive termination

n = Number of components in system

The three contributors to expected total cost may be constructed separately. The expected cost of testing equals the sum of the expected component testing costs.

$$E[\text{testing cost}] = \sum_{i=1}^{n} C_{[i]} \Pr\{[i] \text{ tested}\}. \tag{1}$$

The brackets refer to test order indices. For example, [3] = 5 denotes component 5 tested third. Pr{[i] tested} is the probability that the ith test is made before a positive reading is recorded. To ease notation, the following definitions are used:

$$T_{[i]} = \Pr\{[i] \text{ tested}\}$$
$$T_{[i]}' = \Pr\{[i] \text{ not tested}\}.$$

Obviously, $T_{[1]} = 1$. Conditioned on testing [1], [2] is tested if and only if a negative reading, correct or not, is recorded on the first test. Consequently,

$$T_{[2]} = P_{[1]}B_{[1]} + (1-P_{[1]})(1-a_{[1]}).$$

Note that

$$T_{[2]}' = P_{[1]}(1-B_{[1]}) + (1-P_{[1]})a_{[1]}.$$

Therefore,

$$T_{[2]} + T_{[2]}' = T_{[1]} = 1.$$

Conditioned on testing [2], [3] is tested if and only if a negative reading results on the second test:

$$T_{[3]} = P_{[1]}B_{[1]}(1-a_{[2]}) + P_{[2]}(1-a_{[1]})B_{[2]} + (1-P_{[1]}-P_{[2]})(1-a_{[1]})(1-a_{[2]}).$$

By simple interpretation, this probability expression enumerates the possible states of the components tested. For example, the second term, with factor $P_{[2]}$, refers to the case in which [2] is the failed component; the third term, with factor $(1-P_{[1]}-P_{[2]})$, refers to the case in which neither [1] nor [2] is failed.

By similar reasoning, $T_{[3]}'$ is the probability, conditioned on testing [2], of a positive reading on the second test.

$$T_{[3]}' = P_{[1]}B_{[1]}a_{[2]} + P_{[2]}(1-a_{[1]})(1-B_{[2]}) + (1-P_{[1]}-P_{[2]})(1-a_{[1]})a_{[2]}.$$

With some algebraic manipulation it can be shown that

$$T_{[3]} + T_{[3]}' = T_{[2]}.$$

This reflects the conditional nature of the test probabilities.

Higher order test probabilities can also be constructed using the state enumeration pattern. However, a recursively simpler expression may be derived. The derivation is shown by example.

$$\begin{aligned} T_{[3]} &= P_{[1]}B_{[1]}(1-a_{[2]}) + P_{[2]}(1-a_{[1]})B_{[2]} \\ &\qquad + (1-P_{[1]}-P_{[2]})(1-a_{[1]})(1-a_{[2]}) \\ &= P_{[1]}B_{[1]}(1-a_{[2]}) + (1-P_{[1]})(1-a_{[1]})(1-a_{[2]}) \\ &\qquad + P_{[2]}(1-a_{[1]})B_{[2]} - P_{[2]}(1-a_{[1]})(1-a_{[2]}) \\ &= T_{[2]}(1-a_{[2]}) - P_{[2]}(1-a_{[1]})(1-a_{[2]}-B_{[2]}). \end{aligned}$$

An additional calculation will confirm the relation.

$$\begin{aligned} T_{[4]} &= P_{[1]}B_{[1]}(1-a_{[2]})(1-a_{[3]}) + P_{[2]}(1-a_{[1]})B_{[2]}(1-a_{[3]}) \\ &\qquad + P_{[3]}(1-a_{[1]})(1-a_{[2]})B_{[3]} \\ &\qquad + (1-P_{[1]}-P_{[2]}-P_{[3]})(1-a_{[1]})(1-a_{[2]})(1-a_{[3]}) \\ &= T_{[3]}(1-a_{[3]}) - P_{[3]}(1-a_{[1]})(1-a_{[2]})(1-a_{[3]}-B_{[3]}). \end{aligned}$$

The recursive relation holds generally for i = 1, 2, 3, ...

$$T_{[i+1]} = T_{[i]}(1-a_{[i]}) - P_{[i]}(1-a_{[i]}-B_{[i]}) \prod_{j=1}^{i-2}(1-a_{[j]}). \qquad (2)$$

(When i=1, the embedded product is defined to be one.) Recall that $T_{[1]} = 1$, conditioned on system failure. Together (1) and (2) imply that the expected testing cost is

$$C_{[1]} + \sum_{i=2}^{n} C_{[i]}[T_{[i-1]}(1-a_{[i-1]}) - P_{[i-1]}(1-a_{[i-1]}-B_{[i-1]})\prod_{j=1}^{i-2}(1-a_{[j]})]. \qquad (3)$$

The expected cost of an NDF outcome is the product of the penalty, D_1, and the associated probability. This probability is easily calculated:

$$\Pr\{\text{NDF outcome}\} = \sum_{i=1}^{n} P_i B_i \prod_{j \neq i}(1-a_j). \qquad (4)$$

Since each term in the sum encompasses all n components, order indices are not needed. That is, the probability of an NDF outcome is not affected by the order in which components are tested. Therefore, the expected NDF cost is

$$D_1 \sum_{i=1}^{n} P_i B_i \prod_{j \neq i}(1-a_j). \qquad (5)$$

The expected cost of false positive termination is the product of the penalty, D_2, and the probability of false positive termination. This is the sum over all components of the probability of a false positive reading on a component conditioned on testing that component. The first term in the sum is straightforward.

$$P\{\text{false positive on } [1] \,/\, [1] \text{ tested}\} = (1-P_{[1]})a_{[1]}.$$

Again, all indices are order indices. This term is a component of the probability of not testing [2]. Similarly, the second term in the sum is a component of the probability of not testing [3].

$$\begin{aligned}
&\Pr\{\text{false positive on } [2] \,/\, [2] \text{ tested}\}\\
&\quad = P_{[1]}B_{[1]}a_{[2]} + (1-P_{[1]}-P_{[2]})(1-a_{[1]})a_{[2]}.\\
&\quad = [P_{[1]}B_{[1]} + (1-P_{[1]})(1-a_{[1]})]a_{[2]} - P_{[2]}(1-a_{[1]})a_{[2]}\\
&\quad = [T_{[2]} - P_{[2]}(1-a_{[1]})]a_{[2]}.
\end{aligned}$$

An additional calculation will confirm the relation suggested.

$$\begin{aligned}
&\Pr\{\text{false positive on } [3] \,/\, [3] \text{ tested}\}\\
&\quad = P_{[1]}B_{[1]}(1-a_{[2]})a_{[3]} + P_{[2]}(1-a_{[1]})B_{[2]}a_{[3]}\\
&\qquad\qquad + (1-P_{[1]}-P_{[2]}-P_{[3]})(1-a_{[1]})(1-a_{[2]})a_{[3]}\\
&\quad = [T_{[3]} - P_{[3]}(1-a_{[1]})(1-a_{[2]})]a_{[3]}.
\end{aligned}$$

The relation holds generally for i = 1, 2, 3, ...

$$\begin{aligned}
&\Pr\{\text{false positive on } [i]/[i] \text{ tested}\}\\
&\quad = [T_{[i]} - P_{[i]}\prod_{j=1}^{i-1}(1-a_{[j]})]a_{[i]}. \qquad (6)
\end{aligned}$$

(When i = 1 the embedded product is defined to be 1.)

Therefore, over all components, the probability of false positive termination is

$$\sum_{i=1}^{n}\{T_{[i]} - P_{[i]}\prod_{j=1}^{i-1}(1-a_{[j]}))\}a_{[i]}. \tag{7}$$

Consequently, omitting the index brackets, the expected cost of false positive termination is

$$D_2\sum_{i=1}^{n}\{T_{[i]} - P_{[i]}\prod_{j=1}^{i-1}(1-a_{[j]}))\}a_{[i]}. \tag{8}$$

The expected total cost of the diagnostic scheme is the sum of equations (3), (5), and (8):

$$\begin{aligned} C_{[1]} &+ \sum_{i=2}^{n} C_{[i]}[T_{[i-1]}(1-a_{[i-1]}) \\ &- P_{[i-1]}(1-a_{[i-1]}-B_{[i-1]})\prod_{j=1}^{i-2}(1- a_{[j]})] + D_1\sum_{i=1}^{n} P_i B_i \prod_{j\neq i}(1-a_j) \\ &+ D_2 \sum_{i=1}^{n}\{T_{[i]} - P_{[i]}\prod_{j=1}^{i-1}(1-a_{[j]}))\}a_{[i]}. \end{aligned} \tag{9}$$

In expected value measure, equation (9) captures all of the costs arising over a sample path of the diagnostic program. The imperfect nature of testing is reflected in the error probabilities and impacts both the expected testing cost and the expected error penalties. Once probability and cost parameters are defined, the test sequence determines the overall expected cost.

Chapter Four

Analysis and Results

Analysis of the expected total cost model suggests a strategy for identifying the cost minimizing sequence. The approach taken in this section is based on the adjacent pairwise switching algorithm studied by Baker [1] and applied by Nachlas and Binney [10], Park [11], and Chu [5]. Following the algebraic analysis, a heuristic solution procedure is presented. Application of the heuristic is illustrated by numerical example.

Algebraic Analysis

The analysis proceeds by considering two sequences for an n-component system. One is chosen arbitrarily, and the other differs from the first by the interchange of the [i] and [i+1] elements. The associated expected total cost expressions differ only in the [i] and [i+1] terms. Furthermore, the expected NDF cost is sequence independent and, therefore, common to both. Consequently, the interesting costs pertain to tests of the [i] and [i+1] components and penalties for false positive termination on either of the tests.

In the first sequence, let [i] = m and [i+1] = n. Then the interesting cost equation is

$$Z = C_{[i]}T_{[i]} + C_{[i+1]}T_{[i+1]} + D_2\{T_{[i]} - P_{[i]}K\}a_{[i]}$$
$$+ D_2\{T_{[i+1]} - P_{[i+1]}(1-a_{[i]})K\}a_{[i+1]}. \qquad (10)$$

K is defined to be $\prod_{j=1}^{i-1}(1 - a_{[j]})$. Using (2), $T_{[i+1]}$ may be written as a function of $T_{[i]}$. Making this substitution, retaining $T_{[i]}$ in general form, and otherwise writing m for [i] and n for [i+1], the cost equation becomes

$$Z = C_mT_{[i]} + C_n\{T_{[i]}(1-a_m) - P_m(1-a_m-B_m)K\} + D_2\{T_{[i]}-P_mK\}a_m$$
$$+ D_2\{T_{[i]}(1-a_m) - P_m(1-a_m-B_m)K - P_n(1-a_m)K\}a_n. \qquad (11)$$

As suggested by (2), $T_{[i]}$ is a function of the components tested before [i]. Since the two sequences differ only at [i] and [i+1], T[i] is the same for both. Similarly, K, as defined above, is the same for the two sequences.

In the second sequence [i] = n and [i+1] = m. Expressing $T_{[i+1]}$ in terms of $T_{[i]}$ and substituting n for [i] and m for [i+1], the relevant cost equation becomes

$$Z' = C_nT_{[i]} + C_m\{T_{[i]}(1-a_n) - P_n(1-a_n-B_n)K\} + D_2\{T_{[i]}\ P_nK\}a_n$$
$$+ D_2\{T_{[i]}(1-a_n) - P_n(1-a_n-B_n)K - P_m(1-a_n)K\}a_m. \qquad (12)$$

If $Z < Z'$ then (11) < (12):

$$C_mT_{[i]} + C_nT_{[i]} - C_nT_{[i]}a_m - C_nP_m(1-a_m-B_m)K + D_2T_{[i]}a_m$$
$$- D_2P_mKa_m + D_2T_{[i]}a_n - D_2T_{[i]}a_ma_n - D_2P_m(1-a_m-B_m)Ka_n$$
$$- D_2P_nKa_n + D_2P_nKa_ma_n \qquad <$$
$$C_nT_{[i]} + C_mT_{[i]} - C_mT_{[i]}a_n - C_mP_n(1-a_n-B_n)K + D_2T_{[i]}a_n$$
$$- D_2P_nKa_n + D_2T_{[i]}a_m - D_2T_{[i]}a_na_m - D_2P_n(1-a_n-B_n)Ka_m$$
$$- D_2P_mKa_m + D_2P_mKa_na_m. \qquad (13)$$

Cancelling common terms yields a simpler inequality:

$$C_m\{T_{[i]}a_n + P_n(1-a_n-B_n)K\} + D_2P_n(1-B_n)Ka_m \qquad <$$
$$C_n\{T_{[i]}a_m + P_m(1-a_m-B_m)K\} + D_2P_m(1-B_m)Ka_n. \qquad (14)$$

Finally, dividing both sides by C_mC_n gives

$$(1/C_n)\{T_{[i]}a_n+P_n(1-a_n-B_n)K\} + (1/C_n)D_2(a_m/C_m)P_n(1-B_n)K <$$
$$(1/C_m)\{T_{[i]}a_m+P_m(1-a_m-B_m)K\} + (1/C_m)D_2(a_n/C_n)P_m(1-B_m)K. \quad (15)$$

If this inequality is satisfied, then $Z < Z'$ implying that $[i] = m$ and $[i+1] = n$ produces a lower expected total cost than the reverse. This relationship is the fundamental condition from which a low cost sequence may be constructed.

Insight into the nature of the inequality relationship is gained by considering separately the underlying costs. Again, NDF cost is unaffected by test order and is irrelevant to a discussion of test sequence. The expression in (15) relates to expected cost with imperfect testing.

If, however, tests are perfect, error probabilities are zero and (15) becomes

$$P_n/C_n < P_m/C_m. \tag{16}$$

This is the result achieved in earlier work. When (16) is satisfied, the sequence with [i] = m and [i+1] = n has lower expected cost than the reverse. By extension, under perfect testing, the cost minimizing sequence orders components by decreasing P/C ratio. Papers already cited prove this the optimal permutation. This strategy is hereafter called the "P/C Rule" or the "perfect test sequence."

When tests are imperfect, sequencing strategies are complicated by conflicting cost objectives. Suppose minimum false positive termination cost is sought. Then, in equations (11) and (12), only those terms having factor D_2 are pertinent. Carrying the algebra through, the final inequality becomes

$$P_n(1-B_n)/a_n < P_m(1-B_m)/a_m. \tag{17}$$

When (17) is satisfied the sequence with [i] = m and [i+1] = n produces lower expected false positive termination cost than the reverse. By extension, the sequence minimizing the expected cost of false positive termination orders component tests by decreasing P(1 - B)/a. However, this ranking rule, in general, will not minimize the expected cost of testing components. Therefore, it is not likely to minimize the expected total cost of the diagnostic scheme.

When the expected testing costs are examined, without regard to false positive termination costs, the remaining terms in the equations (11) and (12) become relevant. On simplification, the final inequality becomes

$$(1/C_n)\{T_{[i]}a_n + P_n(1-a_n-B_n)K\} < (1/C_m)\{T_{[i]}a_m + P_m(1-a_m-B_m)K\}. \quad (18)$$

As the error probabilities go to zero this relationship approaches that for the perfect testing problem. However, when the error probabilities are significant, a simple ranking rule for cost minimization does not exist because $T_{[i]}$ is recursively defined. $T_{[i]}$ cannot be calculated until the components tested up to [i-1] have been identified. In other words, the quantities in (18) cannot be evaluated until the sequence is defined.

However, (18) is useful in constructing the optimal permutation sequentially from [1] to [n]. When the relation is satisfied, [i] = m and [i+1] = n is preferred to the reverse. Consequently, at each i level, the component maximizing

$$(1/C)\{T_{[i]}a + P(1-a-B)K\} = (1/C)\{T_{[i]} - T_{[i+1]}\} \quad (19)$$

is selected. Starting at [i=1], $T_{[i]} = 1$. Given the T value, the quantity in (19) can be evaluated for each component and the maximum identified. The component with maximum value is tested first. Knowing the identity of [1], $T_{[2]}$ can be computed, and (19) re-evaluated for the remaining components. Of these, that maximizing (19) is tested second. This continues until the sequence is completely defined. When error probabilities are small, the sequence so defined is likely to be similar to the perfect test sequence. As error probabilities increase, the two solutions diverge. Because the permutation governed by (19) is defined without regard to false positive termination penalties, it generally will not be consistent with the sequence minimizing the expected cost of false positive termination.

Some tradeoff is implied between the expected cost of testing components and the expected cost of false positive termination. Optimizing with respect to either of the individual criteria is likely to suboptimize the expected total cost. The degree of error depends on the relative magnitudes of the underlying cost and probability parameters. Optimizing over expected total cost requires consideration of the entire inequality in (15). However, the complex relation defies separation of variables and has a recursive characteristic reflected in the test probabilities. Consequently, neither a simple ranking procedure nor a sequential identification procedure can be directly applied. Some other less direct technique must be employed.

Algorithm Development

An efficient heuristic has been developed and applied with good results. The heuristic is based on two observations. First, the P/C Rule provides an easily obtained first approximation to the optimal sequence. The smaller the error probabilities or error penalties, the better the perfect-test sequence. Second, when considering adjacent pairs, if the inequality in (15) is not satisfied then a lower expected cost can be achieved by switching the order of the two component tests. The heuristic begins by identifying the perfect test sequence. Then left-to-right passes through the sequence are made evaluating the quantities in (15) for all adjacent pairs. At a particular step, if the inequality is satisfied, the heuristic moves on to the next pair. For example, if (15) is satisfied at [i] = m and [i+1] = n, the heuristic moves to compare [i+1] = n and [i+2]. If the inequality is not satisfied, the two component tests are interchanged. For example, [i] would become n and [i+1] would become m. Following an interchange the process begins again from the far left. The heuristic stops once a complete pass from left-to-right can be made without switching any component pair.

Each time an interchange is made, in strict accord with (15), the resultant sequence represents an expected cost reduction over the preceding sequence. Since each move constitutes a strict descent in the objective function, it would seem that the cycling phenomenon is not a problem. Of course, in contrived problems numerical accuracy may become a significant factor promoting inefficiencies in the solution algorithm. However, this is more a matter of theoretical interest than of practical concern.

Given certain conditions, it is possible to demonstrate that the algorithm systematically moves toward the optimum. These conditions tend to apply quite generally to the class of problems studied here and so do not constrain application of the heuristic. The informal proof follows a pattern of inductive reasoning.

Suppose [1] = m, [2] = n yields lower expected cost than the reverse. Recall that T = 1 and K = 1 at [i=1]. Then, by (15)

$$(1/C_n)\{a_n + P_n(1-a_n-B_n)\} + (1/C_n)D_2(a_m/C_m)P_n(1-B_n) \quad < $$
$$(1/C_m)\{a_m + P_m(1-a_m-B_m)\} + (1/C_m)D_2(a_n/C_n)P_m(1-B_m). \qquad (20)$$

Now suppose that the same components are compared at [i=2] and that [2] = n, [3] = m is preferred to its reverse. The proof will show that this contradicts the assumption behind (20). Again by (15):

$$(1/C_m)\{T_{[2]}a_m + P_m(1-a_m-B_m)K\} + (1/C_m)D_2(a_n/C_n)P_m(1-B_m)K <$$
$$(1/C_n)\{T_{[2]}a_n + P_n(1-a_n-B_n)K\} + (1/C_n)D_2(a_m/C_m)P_n(1-B_n)K.$$

At [i=2], K = $(1-a_{[1]})$. Using (2), $T_{[2]}$ can be written as a function of $T_{[1]}$. Making this substitution,

$$(a_m/C_m)\{(1-a_{[1]}) - P_{[1]}(1-a_{[1]}-B_{[1]})\}$$
$$+(1/C_m)P_m(1-a_m-B_m)(1-a_{[1]})+(1/C_m)D_2(a_n/C_n)P_m(1-B_m)(1-a_{[1]}) <$$
$$(a_n/C_n)\{(1-a_{[1]}) - P_{[1]}(1-a_{[1]}-B_{[1]})\}$$
$$+(1/C_n)P_n(1-a_n-B_n)(1-a_{[1]})+(1/C_n)D_2(a_m/C_m)P_n(1-B_n)(1-a_{[1]}).$$

Rearranging terms,

$$(1/C_m)\{a_m + P_m(1-a_m-B_m) + D_2(a_n/C_n)P_m(1-B_m)\}(1-a_{[1]})$$
$$- (a_m/C_m)P_{[1]}(1-a_{[1]}-B_{[1]}) \quad <$$
$$(1/C_n)\{a_n + P_n(1-a_n-B_n) + D_2(a_m/C_m)P_n(1-B_n)\}(1-a_{[1]})$$
$$- (a_n/C_n)P_{[1]}(1-a_{[1]}-B_{[1]}).$$

Dividing by $(1-a_{[1]})$,

$$(1/C_m)\{a_m + P_m(1-a_m-B_m) + D_2(a_n/C_n)P_m(1-B_m)\}$$
$$- (a_m/C_m)P_{[1]}(1-a_{[1]}-B_{[1]})/(1-a_{[1]}) \quad <$$
$$(1/C_n)\{a_n + P_n(1-a_n-B_n) + D_2(a_m/C_m)P_n(1-B_n)\}$$
$$- (a_n/C_n)P_{[1]}(1-a_{[1]}-B_{[1]})/(1-a_{[1]}). \qquad (21)$$

If $a_{[1]}+B_{[1]} > 1$, the subtrahends in the inequality are negative. Therefore, unless $(a_m/C_m) << (a_n/C_n)$, (21) contradicts (20). However, if a and B are error probabilities of conventional magnitude, their sum will not exceed one. Consequently, this case need not be considered. Conversely, if $a_{[1]}+B_{[1]} < 1$, the subtrahends in (21) are positive. Unless $(a_m/C_m) >> (a_n/C_n)$, (21) contradicts (20). It is theoretically possible that $(a_m/C_m) >> (a_n/C_n)$. However, such a condition is not conceptually consistent with the nature of the problem under study. More reasonably, a_m and a_n would have the same order of magnitude, as would C_m and C_n.

Assuming the a/C ratios are of the same order, the contradiction holds, implying a preferred order for components m and n at [i=1] and [i=2]. That is, if order -n-m-... is preferred at [i=2], then order n-m-... must also be preferred at [i=1]. Otherwise, the algebraic contradiction arises. The implication is strengthened by generalization.

Continuing the inductive reasoning, suppose that [i] = m, [i+1] = n is preferred to [i] = n, [i+1] = m. Then, by (15):

$$(1/C_n)\{T_{[i]}a_n+P_n(1-a_n-B_n)K\}+(1/C_n)D_2(a_m/C_m)P_n(1-B_n)K < (1/C_m)\{T_{[i]}a_m+P_m(1-a_m-B_m)K\}+(1/C_m)D_2(a_n/C_n)P_m(1-B_m)K. \quad (22)$$

Now suppose that the same components are compared at the next sequence position and that [i+1] = n, [i+2] = m is preferred to the reverse. Again it can be shown that this leads to a contradiction. By (15):

$$(1/C_m)\{T_{[i+1]}a_m+P_m(1-a_m-B_m)K'\} + (1/C_m)D_2(a_n/C_n)P_m(1-B_m)K' <$$
$$(1/C_n)\{T_{[i+1]}a_n+P_n(1-a_n-B_n)K'\} + (1/C_n)D_2(a_m/C_m)P_n(1-B_n)K',$$

where $K' = \prod_{j=1}^{i}(1-a_{[j]}) = K(1-a_{[i]})$. Then,

$$\begin{aligned}&(a_m/C_m)T_{[i+1]} + (1/C_m)P_m(1-a_m-B_m)K(1-a_{[i]})\\ &\quad + (1/C_m)D_2(a_n/C_n)P_m(1-B_m)K(1-a_{[i]}) \quad <\\ &(a_n/C_n)T_{[i+1]} + (1/C_n)P_n(1-a_n-B_n)K(1-a_{[i]})\\ &\quad + (1/C_n)D_2(a_m/C_m)P_n(1-B_n)K(1-a_{[i]}).\end{aligned}$$

$T_{[i+1]}$ is the same for both permutations of m and n. Using (2), it can be expressed in terms of $T_{[i]}$:

$$\begin{aligned}&(a_m/C_m)\{T_{[i]}(1-a_{[i]}) - P_{[i]}(1-a_{[i]}-B_{[i]})K\}\\ &\quad + (1/C_m)\{P_m(1-a_m-B_m) + D_2(a_n/C_n)P_m(1-B_m)\}K(1-a_{[i]}) \quad <\\ &(a_n/C_n)\{T_{[i]}(1-a_{[i]}) - P_{[i]}(1-a_{[i]}-B_{[i]})K\}\\ &\quad + (1/C_n)\{P_n(1-a_n-B_n) + D_2(a_m/C_m)P_n(1-B_n)\}K(1-a_{[i]}).\end{aligned}$$

Rearranging terms,

$$(1/C_m)\{T_{[i]}a_m + P_m(1-a_m-B_m)K + D_2(a_n/C_n)P_m(1-B_m)K\}(1-a_{[i]}) - (a_m/C_m)P_{[i]}(1-a_{[i]}-B_{[i]})K \quad < \quad (1/C_n)\{T_{[i]}a_n + P_n(1-a_n-B_n)K + D_2(a_m/C_m)P_n(1-B_n)K\}(1-a_{[i]}) - (a_n/C_n)P_{[i]}(1-a_{[i]}-B_{[i]})K.$$

Dividing by $(1-a_{[i]})$,

$$(1/C_m)\{T_{[i]}a_m+P_m(1-a_m-B_m)K+D_2(a_n/C_n)P_m(1-B_m)K\} - (a_m/C_m)P_{[i]}(1-a_{[i]}-B_{[i]})K/(1-a_{[i]}) \quad < \quad (1/C_n)\{T_{[i]}a_n+P_n(1-a_n-B_n)K+D_2(a_m/C_m)P_n(1-B_n)K\} - (a_n/C_n)P_{[i]}(1-a_{[i]}-B_{[i]})K/(1-a_{[i]}). \qquad (23)$$

The reasoning here parallels that used before. If $a_{[i]}+B_{[i]} > 1$, the subtrahends are negative. Unless $(a_m/C_m) << (a_n/C_n)$, (23) contradicts (22). But, as before, this case is implausible and may be disregarded. In the more reasonable case, $a_{[i]}+B_{[i]} < 1$, and the subtrahends are positive. Unless $a_m/C_m >> a_n/C_n$, (23) contradicts (22).

As already stated, the error probabilities, a, are likely to be of the same order of magnitude. A similar supposition holds for the testing cost coefficients, C, and, for that matter, for the type II error probabilities, B. Under these conditions, the contradiction holds, and the argument implies a preferred order for each pair of components throughout the testing sequence. That is, at some i, if [i] = m, [i+1] = n is preferred to the reverse, then this order will be preferred at all other i. Once the heuristic establishes an order for a component pair, the two will not be interchanged by any subsequent comparison. Therefore, when the a/C ratios are of consistent magnitudes, the algorithm prevents cycling back to sequences already encountered. Rather, each time a pairwise interchange is justified by the heuristic, a new sequence results. Successive sequences represent reductions in the expected total cost function and, consequently, converge to the optimal solution.

In peculiar conditions, under which the heuristic may perform inefficiently, other techniques for finding an approximate solution may be employed. The simplest alternative would order component tests by either the P/C or the P(1-B)/a ranking rule already discussed. While neither will yield the optimal solution, each is easily applied, and, under particular circumstances, one or the other will produce an acceptable solution as compared to an arbitrary ordering. Nevertheless, the full heuristic algorithm had excellent results in an extensive set of problems. Furthermore, the full heuristic is easily implemented, requiring only elementary arithmetic calculations.

Numerical Example

Application of the test sequencing heuristic is best explained by example. Suppose a series system has eight components. Each component follows the Weibull life distribution given by

$$R(t) = \exp(-ut^{v}) \tag{24}$$

From Nachlas and Binney [10], the conditional failure probability is

$$P_i = \{\int_{t_1}^{t_2} f_i(t) \prod_{j \neq i} R_j(t)dt\} / \{R_s(t_1) - R_s(t_2)\}. \tag{25}$$

The interval $[t_1, t_2]$ is the time span during which system failure occurred. $R_s(t)$ is the system reliability function. For a series system, $R_s(t)$ is the product of the individual $R_i(t)$'s. $f_i(t)$ is the lifetime probability density function. Since the hazard function, $z_i(t)$ is defined as $f_i(t)/R_i(t)$, the numerator in (25) may be expressed as

$$\int_{t_1}^{t_2} z_i(t) R_s(t) dt. \tag{26}$$

For the Weibull failure model,

$$z_i(t) = u_i v_i t^{vi-1}. \tag{27}$$

Using equations (25), (26), and (27), given a particular time interval of failure and a set of parametric values, the conditional failure probabilities may be calculated. For the Weibull failure model the integral in (26) cannot be evaluated in closed form. However, numerical methods are easily applied. In this example, $t_1 = 1{,}000$ and $t_2 = 1{,}500$. The Weibull parameters and resultant failure probabilities are given in Table 1. Test error penalties are $D_1 = 25$ and $D_2 = 100$. The remaining cost and probability data are provided in Table 2.

In an eight component system there are 8! = 40,320 test permutations. The expected total cost of each may be calculated using the model in (9). However, identifying the cost minimizing sequence by complete enumeration is inefficient. For systems of larger order, the method of complete enumeration becomes practically impossible to implement. Consequently, construction of a cost efficient esting strategy requires the use of an algorithm such as that presented here.

To apply the algorithm, the suggested initial solution, the perfect test sequence, is 1-6-2-5-7-8-3-4. The corresponding expected total cost is $25.13. The heuristic begins by comparing components 1 and 6 at [i=1]. The inequality in (15) is satisfied for [1] = 1 and [2] = 6. Therefore, no interchange is justified, and the heuristic next compares components 6 and 2 at [i=2]. Again the interchange is not justified. The original sequence is preserved until the comparison of components 5 and 7 at

Table 1

Example 1: Weibull Parameters and Conditional Failure Probabilities.

Component	u	v	P
1	2E-06	.90	.2836
2	5E-06	.67	.1026
3	1E-06	.80	.0618
4	3E-06	.41	.0059
5	6E-06	.64	.0950
6	5E-06	.81	.3362
7	1E-06	.85	.0938
8	9E-06	.44	.0211
			1.0000

Table 2

Example 1: Cost and Probability Data.

Component	C	a	B
1	6	.040	.008
2	4	.071	.008
3	7	.042	.088
4	3	.057	.091
5	5	.097	.028
6	8	.026	.078
7	5	.006	.065
8	2	.022	.015

[i=4]. At this step the interchange is justified, and the sequence becomes 1-6-2-7-5-8-3-4. The corresponding expected total cost is $24.55. Following the change, the heuristic restarts from the left, beginning the chain of comparisons at the first perturbation. In this case, the comparison resumes between components 2 and 7 at [i=3]. Again the interchange is justified, and the sequence now becomes 1-6-7-2-5-8-3-4 with expected total cost $24.26. The algorithm restarts by comparing components 6 and 7 and [i=2]. No additional changes are warranted, and the final sequence is 1-6-7-2-5-8-3-4. The final cost represents a three and one-half percent savings over the initial cost. Savings magnitude depends on the magnitude of the underlying costs, and in other examples the cost reduction may be substantial. More importantly, when all sequences and corresponding costs are enumerated, the optimum is revealed to be identical to that generated by the heuristic.

Experience with the algorithm suggests that choice of initial sequence does not affect the eventual solution. That is, the algorithm appears to be globally convergent. However, the initial solution may affect the rate of convergence. Conventionally, error probabilities are small; typically, less than 0.10. Then, unless the false positive termination penalty is several orders larger than the testing cost coefficients, the expected testing cost dominates the expected false positive termination cost. In these cases, the P/C Rule yields a sequence close to that minimizing the expected testing cost and provides a good initial solution. However, when the expected cost of false positive termination is significant, because of unconventional error probabilities or large error penalty, the P/C Rule may not give the most efficient starting solution.

To illustrate this point, consider a second example, a modification of the first in which some error probabilities have been dramatically increased. Cost and probability data are provided in Table 3. The P/C Rule generates the sequence 1-6-2-5-7-8-3-4 with expected total cost \$32.65: expected testing cost = \$15.23, expected false positive termination cost = \$17.00, and expected NDF cost = \$0.41. When this is taken as the initial sequence the algorithm requires fourteen pairwise comparisons involving five transpositions to reach the optimal solution. It is possible to improve the efficiency of the heuristic by selecting a different starting solution. First note that the P/C sequence renders the expected testing cost and expected false positive termination cost nearly equal. However, neither is minimized. Furthermore, the P/C sequence is markedly different than the sequence minimizing expected testing cost: 8-4-2-1-6-5-7-3. On this ordering, the expected total cost is \$73.45: expected testing cost \$10.56, expected false positive termination cost \$62.48, and

Table 3

Example 2: Cost and Probability Parameters.

Component	u	v	P	a	B	C
1	2E-06	.90	.2836	.040	.008	6
2	5E-06	.67	.1026	.143	.084	4
3	1E-06	.80	.0618	.042	.088	7
4	3E-06	.41	.0059	.301	.136	3
5	6E-06	.64	.0950	.097	.028	5
6	5E-06	.81	.3362	.156	.065	8
7	1E-06	.85	.0938	.006	.065	5
8	9E-06	.44	.0211	.352	.214	2
			1.0000			

$t_1 = 1{,}000$ $t_2 = 1{,}500$

$D_1 = 25$ $D_2 = 100$

expected NDF cost $0.41. When the expected testing cost sequence initializes the algorithm, thirty-two comparisons with eighteen interchanges are required. Again the true optimum is reached, but the heuristic's efficiency clearly is not improved. Because the expected false positive termination cost tends to be large, due to inflated error probabilities, it is reasonable to suspect the sequence minimizing expected false positive termination cost to be a good starting solution. The designated sequence is 7-1-6-3-5-2-8-4 with expected total cost $31.37: expected testing cost $18.74, expected false positive termination cost $12.21, and expected NDF cost $0.41. When the heuristic begins here, nine pairwise comparisons and three interchanges are required to reach the optimal solution of 1-7-6-5-2-3-8-4 with expected total cost $30.23.

Although all three initial alternatives eventually reach the true optimum, the sequence minimizing expected false positive termination cost does so most quickly. This suggests that in implementing the algorithm flexibility in choice of an initial solution may lead to improved efficiency. When possible, the starting solution should reflect dominant characteristics of the underlying parameter set. Nonetheless, given any initial solution, the algorithm behaves well with minimal computational effort.

Chapter Five

Summary

When repairable coherent systems fail via blind failure modes, it is necessary to test individual components to locate and correct the fault. If the system has a series structure and instantaneous failure, the diagnostic program seeks the single failed component, and testing terminates once it has been found. Costs are incurred with the testing of each component. Furthermore, when tests are subject to error, costs are incurred with incorrect diagnoses. Because testing stops once a positive failure reading is recorded, all components need not be tested. Therefore, diagnostic tests should be ordered in a cost effective manner.

This paper mathematically models the expected total cost of the diagnostic strategy including the cost of testing components, the cost of false positive termination, and the cost of inconclusive, NDF, testing. Given the model, the objective is to extract the test sequence minimizing the expected total cost. When error probabilities are zero, the model collapses to an analog of those presented by other authors. The optimal solution is then obtained by application of a simple ranking rule. When error probabilities are non-trivial the solution must be obtained through less direct methods.

Despite the combinatorial nature of the solution space, an efficient heuristic algorithm has been constructed. The algorithm is patterned on an exhaustive set of adjacent pairwise comparisons and moves through an ordered enumeration of candidate solutions. The heuristic requires only simple arithmetic computations and has been applied to a variety of problems with extremely good results. Experimental trials suggest that this solution method globally converges to the optimum with choice of initial solution affecting the speed of convergence.

While only medium sized problems ($n < 10$) were studied, the algorithm has greatest value when applied to large systems. As the number of components increases, the expected cost of the diagnostic system escalates proportionally. Depending on the sequence employed, the expected total cost may vary over a significant range. However, it becomes nearly impossible to identify the cost minimizing sequence by complete enumeration. Substantial expected cost savings may be realized if testing is sequenced by the heuristic algorithm rather than arbitrarily.

The model presented extends earlier work by the inclusion of type I and II testing errors and one-pass sample paths. Extensions to this model are worthy of further study. Order dependent testing cost coefficients and component-specific false positive termination penalties are two possible modifications. Either would have a noticeable affect on optimal sequencing rules, but would also complicate the mathematical modeling approaches and workable solution techniques.

References

1. Baker, Kenneth R. Introduction to Sequencing and Scheduling, New York: John Wiley & Sons, Inc., 1974, pp. 42-44.

2. Barlow, Richard E. and Frank Proschan, Statistical Theory of Reliability and Life Testing: Probability Models, Silver Spring, MD: To Begin With, reprint 1981.

3. Butler, David A. and Gerald J. Lieberman, "Inspection Policies for Fault Location," Operations Research, 32, 3, (1984),566-574.

4. Butterworth, Richard, "Some Reliability Fault-Testing Models," Operations Research, 20, 2, (1972), 335-343.

5. Chu, Wesley W. "A Mathematical Model for Diagnosing System Failures," IEEE Transactions on Electronic Computers, VEC-16, 3, (1967), 327-331.

6. Firstman, Sidney I. and Brian Gluss. "Optimum Search Routines for Automatic Fault Location," Operations Research, 8, (1960), 512-523.

7. Gluss, Brian. "An Optimum Policy for Detecting a Fault in a Complex System," Operations Research, 7, 4, (1959), 468-477.

8. Kumar, Ashok and Vipin B. Kapoor, "Optimal Classification of Failures in a Two-Unit Series System," International Journal of Systems Science, 12, 1, (1981), 127-132.

9. Matula, David, "A Periodic Optimal Search," American Mathematical Monthly, 71, 15, (1964), 15-21.

10. Nachlas, Joel A. and Blair A. Binney, "Reliability Based Diagnostic Strategies for Series Systems," Proceedings 4th IMEKO International Symposium on Technical Diagnostics, (1986), 2.8-2.10.

11. Park, Kyung S., "Optimal Diagnostic Procedure for Failures in a Series System," International Journal of Systems Science, 13, 4, (1982), 409-412.

Appendix A

BASIC program numerically evaluating component failure probabilities.

```
1     REM Program numerically evaluates conditional failure
2     REM probabilities using Nachlas and Binney [10] def'n.
3     REM Weibull failure distribution assumed.  Numerical
4     REM integration by Simpson's Rule.
5     REM
6     REM Variable Definition
7     REM
8     REM    A, B:      Weibull parameters, where component
9     REM               reliability R=exp(-At^B)
10    REM    T1, T2:    endpoints of time span during which
11    REM               system failure occurred
12    REM    P:         component conditional failure prob.
13    REM    NUM:       number of system components
14    REM    RT1, RT2:  component reliability evaluated at t1
15    REM               and t2
16    REM    RST1,RST2:system reliability evaluated at t1, t2
17    REM    DELTA:     stepsize used in Simpson's Rule integr
18    REM    X, XHALF:  abscissa variables used in Simpson's
19    REM               Rule integration
20    REM    Y, YHALF:  ordinate variables used in Simpson's
21    REM               Rule integration
22    REM    N:         number of steps used in Simpson's Rule
23    REM    SUM1:      accumulates Nachlas-Binney numerator
24    REM               evaluated at interval endpoints
25    REM    SUM2:      accumulates Nachlas-Binney numerator
26    REM               evaluated at interval midpoints
27    REM
28    REM
75    DIM A(10), B(10), P(10), RT1(10), RT2(10), X(1000),
              Y(1000), XHALF(1000), YHALF(1000)
97    REM
98    REM Keyboard input of parameter variables
99    REM
100   CLS:INPUT "Enter number of system components ",NUM
110   PRINT "Enter Weibull distribution parameters"
115   FOR I = 1 TO NUM
120           PRINT "Component ",I
125           INPUT "Enter alpha value ",A(I)
130           INPUT "Enter beta  value ",B(I)
135           PRINT
140   NEXT I
145   INPUT "Enter t1 ",T1
150   INPUT "Enter t2 ",T2
155   INPUT "Enter number of steps ",N
197   REM
198   REM Calculate stepsize
199   REM
200   DELTA = (T2 - T1)/N
247   REM
248   REM Calculate component & system reliability at t1, t2
249   REM
```

```
250   FOR I = 1 TO NUM
255           RT1(I) = EXP(-A(I)*T1^B(I))
260           RT2(I) = EXP(-A(I)*T2^B(I))
265   NEXT I
270   PRINT TAB(10);"I";TAB(25);"R(T1)";TAB(35);"R(T2)"
275   FOR I = 1 TO NUM
280           PRINT TAB(10);I;TAB(25);RT1(I);TAB(35);RT2(I)
285   NEXT I
290   RST1 = 1
295   RST2 = 1
300   FOR I = 1 TO NUM
305           RST1 = RST1 * RT1(I)
310           RST2 = RST2 * RT2(I)
315   NEXT I
320   PRINT TAB(15);"Rs(t1)";TAB(25);RST1;TAB(35);RST2;
              TAB(50);"Rs(t2)"
347   REM
348   REM Calculate P for each component
349   REM
350   FOR I = 1 TO NUM
351   REM
352   REM Evaluate Nachlas-Binney numerator at time sp endpts
353   REM
355           X0 = T1
360           XN = T2
365           Y0 = A(I)*B(I)*X0^(B(I)-1)
370           FOR J = 1 TO NUM
375                   Y0 = Y0*EXP(-A(J)*X0^B(J))
380           NEXT J
390           YN = A(I)*B(I)*XN^(B(I)-1)
395           FOR J = 1 TO NUM
400                   YN = YN*EXP(-A(J)*XN^B(J))
405           NEXT J
407   REM
408   REM Evaluate Nachlas-Binney numerator at intrvl endpts
409   REM
410           X(1) = X0 + DELTA
415           FOR K = 2 TO N-1
420                   X(K) = X(K-1) + DELTA
425          NEXT K
430           SUM1 = 0
435           FOR K = 1 TO N-1
440                   Y(K) = A(I)*B(I)*X(K)^(B(I)-1)
445                   FOR J = 1 TO NUM
450                      Y(K) = Y(K)*EXP(-A(J)*X(K)^B(J))
455                   NEXT J
460                   SUM1 = SUM1 + Y(K)
465           NEXT K
467   REM
468   REM Evaluate N-B numerator at interval midpoints
469   REM
```

```
470          XHALF(1) = (X0 + X(1))/2
475          FOR K = 2 TO N
480                    XHALF(K) = XHALF(K-1) + DELTA
485          NEXT K
490          SUM2 = 0
495          FOR K = 1 TO N
500                  YHALF(K) = A(I)*B(I)*XHALF(K)^(B(I)-1)
505                  FOR J = 1 TO NUM
510              YHALF(K)=YHALF(K)*EXP(-A(J)*XHALF(K)^B(J))
515                  NEXT J
520                  SUM2 = SUM2 + YHALF(K)
525          NEXT K
526   REM
527   REM Calculate N-B numerator integral using Simpson's
528   REM Rule approximation
529   REM
530          INTEGR1 = Y0 + YN + 2*SUM1 + 4*SUM2
535          INTEGR2 = (T2 - T1)*INTEGR1 / (6*N)
537   REM
538   REM Calculate N-B conditional failure probability
539   REM
540          P(I) = INTEGR2 / (RST1 - RST2)
545          PRINT TAB(10);"I  ";I;"P(I)  ";TAB(25);P(I)
550   NEXT I
600   STOP
605   END
```

Appendix B

BASIC program computing expected total cost for all permutations of an 8 component diagnostic strategy.

```
1     REM Program computes expected total cost for all perm-
2     REM utations of an 8-component diagnostic strategy.
3     REM The cost minimizing sequence is identified.
4     REM
5     REM Variable Definition
6     REM
7     REM    A:      false positive error probability
8     REM    B:      false negative error probability
9     REM    C:      component testing cost
10    REM    P:      component conditional failure probability
11    REM    D1:     NDF penalty cost
12    REM    D2:     false positive termination penalty cost
13    REM    FPOS:   false positive termination probability
14    REM    NDF:    NDF outcome probability
15    REM    FPOSCOST:expected false pos termination cost
16    REM    NDFCOST:expected NDF outcome cost
17    REM    TESTCOST:expected testing cost
18    REM    TOTAL:  expected total cost
19    REM    T:      test probability
20    REM    MIN:    minimum total cost yet encountered.
21    REM            initialized to 100,000!.  updated with
22    REM            each pass through program
23    REM    MINROW:stores sequence corresponding to MIN
24    REM    L1-L8:  sequence variables.  L3 stores number of
25    REM            component tested third.
26    REM    K(J,I):identity of component tested ith in jth
27    REM            permutation
28    REM
29    REM
50    DIM A(8), B(8), C(8), P(8), T(8), K(720,8), MINROW(8)
97    REM
98    REM Keyboard input of parameter values
99    REM
100   PRINT "Number of system components is 8":NUM = 8
105   INPUT "Enter cost of NDF outcome",D1
110   INPUT "Enter cost of false positive termination",D2
115   PRINT:PRINT "Enter component probability and cost data"
120   FOR I = 1 TO NUM
125            PRINT "Component ",I
130            INPUT "failure probability:  ",P(I)
135            INPUT "false pos probability:  ",A(I)
140            INPUT "false neg probability:  ",B(I)
145            INPUT "test cost:  ",C(I)
150            PRINT
155   NEXT I
160   CLS
165   MIN = 100000!
166   REM
167   REM L1 and L2 input from data file.  Each assumes all
168   REM values from 1 to 8, and all distinct pairs are
169   REM considered.  Program terminates when L1=L2=9
```

```
170  REM
175  OPEN "B:DATA" FOR INPUT AS #1
178  INPUT #1, L1, L2
180  IF L1 = L2 GOTO 910
184  REM
185  REM Call subroutine to generate remainder of sequence.
186  REM L1 and L2 have been input from data file.  L3 to L8
187  REM are generated in subroutine.
188  REM
190  GOSUB 10000
195  REM
196  REM Loop drives program through all possible test
197  REM sequences for given L1, L2 (Note: 6! = 720)
200  REM
205  FOR J = 1 TO 720
210  REM
211  REM Calculate T's using recursive definition
212  REM
215           T(1) = 1
220           ONE = K(J,1)
225           T(2) = (1-A(ONE))-P(ONE)*(1-A(ONE)-B(ONE))
230           FOR I = 2 TO NUM-1
235                     MARK = K(J,I)
240                     PROD = 1
245                     FOR M = 1 TO I-1
250                               HOLD = K(J,M)
255                               PROD = PROD*(1-A(HOLD))
260                     NEXT M
265                 T(I+1)=PROD*P(MARK)*(1-A(MARK)-B(MARK))
270                 T(I+1)=T(I)*(1-A(MARK))-T(I+1)
275           NEXT I
397  REM
398  REM Calculate the probability of false pos termination
399  REM
400           ONE = K(J,1)
405           FPOS = (T(1)-P(ONE))*A(ONE)
410           FOR I = 2 TO NUM
415                     MARK = K(J,I)
420                     PROD = 1
425                     FOR M = 1 TO I-1
430                               HOLD = K(J,M)
435                               PROD = PROD*(1-A(HOLD))
440                     NEXT M
445                   FPOS=FPOS+(T(I)-P(MARK)*PROD)*A(MARK)
450           NEXT I
455  REM
456  REM Calculate expected cost of false pos termination
457  REM
460           FPOSCOST = D2*FPOS
497  REM
498  REM Calculate expected cost of testing components
```

```
499  REM
500           TESTCOST = 0
505           FOR I = 1 TO NUM
510                     HOLD = K(J,I)
515                     TESTCOST = TESTCOST + T(I)*C(HOLD)
520           NEXT I
597  REM
598  REM Calculate expected NDF cost
599  REM
600           NDF = 0
605           FOR I = 1 TO NUM
610                     PROD = 1
615                     FOR M = I TO NUM
620                               IF M=I GOTO 630
625                               PROD = PROD*(1-A(M))
630                     NEXT M
635                     NDF = NDF + P(I)*B(I)*PROD
640           NEXT I
645           NDFCOST = NDF*D1
697  REM
698  REM Calculate expected total cost
699  REM
700           TOTAL = FPOSCOST + TESTCOST + NDFCOST
701  REM
702  REM Compare TOTAL to MIN and update MIN and MINROW, if
703  REM necessary
704  REM
705           IF TOTAL>MIN GOTO 800
710                     MIN = TOTAL
715                     FOR I = 1 TO NUM
720                               MINROW(I) = K(J,I)
725                     NEXT I
797  REM
798  REM Use next permutation for this L1, L2 pair
799  REM
800  NEXT J
900  REM
901  REM All permutations have been run for this L1,L2 pair.
902  REM Return to input statement to do another run.
903  REM
905  GOTO 178
906  REM
907  REM When data file has been exhausted, control returns
908  REM here.  Print most recent MIN and MINROW
909  REM
910  PRINT:PRINT "Min total cost is;MIN;" generated by
              sequence";
915  FOR I = 1 TO NUM
920           PRINT MINROW(I);
925  NEXT I
930  PRINT
```

```
1000 STOP:END
9996 REM
9997 REM Subroutine to generate permutations.  L1 and L2
9998 REM read in from data file above
9999 REM
10000 J=0
10080 FOR L3=1 TO NUM
10090     IF L3 = L1 GOTO 12990
10100     IF L3 = L2 GOTO 12990
10110     FOR L4 = 1 TO NUM
10120         IF L4 = L1 GOTO 12980
10130         IF L4 = L2 GOTO 12980
10140         IF L4 = L3 GOTO 12980
10150         FOR L5 = 1 TO NUM
10160             IF L5 = L1 GOTO 12970
10170             IF L5 = L2 GOTO 12970
10180             IF L5 = L3 GOTO 12970
10190             IF L5 = L4 GOTO 12970
10200             FOR L6 = 1 TO NUM
10210                 IF L6 = L1 GOTO 12960
10220                 IF L6 = L2 GOTO 12960
10230                 IF L6 = L3 GOTO 12960
10240                 IF L6 = L4 GOTO 12960
10250                 IF L6 = L5 GOTO 12960
10260                 FOR L7 = 1 TO NUM
10270                     IF L7 = L1 GOTO 12950
10280                     IF L7 = L2 GOTO 12950
10290                     IF L7 = L3 GOTO 12950
10300                     IF L7 = L4 GOTO 12950
10310                     IF L7 = L5 GOTO 12950
10320                     IF L7 = L6 GOTO 12950
10330                     FOR L8 = 1 TO NUM
10340                         IF L8 = L1 GOTO 12940
10350                         IF L8 = L2 GOTO 12940
10360                         IF L8 = L3 GOTO 12940
10370                         IF L8 = L4 GOTO 12940
10380                         IF L8 = L5 GOTO 12940
10390                         IF L8 = L6 GOTO 12940
10400                         IF L8 = L7 GOTO 12940
10498 REM
10499 REM
10500                         J=J+1
10510                         K(J,1) = L1
10520                         K(J,2) = L2
10530                         K(J,3) = L3
10540                         K(J,4) = L4
10550                         K(J,5) = L5
10560                         K(J,6) = L6
10570                         K(J,7) = L7
10580                         K(J,8) = L8
10998 REM
```

```
10999 REM
12940                     NEXT L8
12950                 NEXT L7
12960             NEXT L6
12970         NEXT L5
12980     NEXT L4
12990 NEXT L3
13998 REM
13999 REM
14000 RETURN
15000 STOP:END
```