APPENDIX E: Comparison of fault detection algorithms for real-time diagnosis in large-scale systems

Presented at the SPIE Aerosense Conference, Orlando, FL, April 16-20, 2001.

528788

Fault Detection Algorithms for Real-Time Diagnosis in Large-Scale Systems

Thiagalingam Kirubarajan[a], Venkat Malepati[b], Somnath Deb[b] and Jie Ying[a]

[a] Dept. of Electrical and Computer Engineering, Univ. of Connecticut, Storrs, CT 06269-2157
[b] Qualtech Systems, Inc., 100 Great Meadow Road, Suite 501, Wethersfield , CT 06109

ABSTRACT

In this paper, we present a review of different real-time capable algorithms to detect and isolate component failures in large-scale systems in the presence of inaccurate test results. A sequence of imperfect test results (as a row vector of 1's and 0's) are available to the algorithms. In this case, the problem is to recover the uncorrupted test result vector and match it to one of the rows in the test dictionary, which in turn will isolate the faults. In order to recover the uncorrupted test result vector, one needs the accuracy of each test. That is, its detection and false alarm probabilities are required. In this problem, their true values are not known and, therefore, have to be estimated online. Other major aspects in this problem are the large-scale nature and the real-time capability requirement. Test dictionaries of sizes up to 1000 x 1000 are to be handled. That is, results from 1000 tests measuring the state of 1000 components are available. However, at any time, only 10-20% of the test results are available. Then, the objective becomes the real-time fault diagnosis using incomplete and inaccurate test results with online estimation of test accuracies. It should also be noted that the test accuracies can vary with time --- one needs a mechanism to update them after processing each test result vector. Using Qualtech's TEAMS-RT (system simulation and real-time diagnosis tool), we test the performances of 1) TEAMS-RT's built-in diagnosis algorithm, 2) Hamming distance based diagnosis, 3) Maximum Likelihood based diagnosis, and 4) HiddenMarkov Model based diagnosis.

Keywords: fault diagnosis, fault isolation, real-time diagnosis, large-scale systems, performance analysis

1. INTRODUCTION

The problem of fault diagnosis and isolation in large-scale systems requires computationally efficient algorithms that can process large amounts of data in order to achieve real-time capability. In addition, the algorithms should not sacrifice fault isolation accuracy for computational efficiency. Recent advances in sensor technology, communications and computational capabilities have made online system health management an essential component of complex system operations. By means of smart sensors onboard a system, low-level test decisions are made based on processing of sensed waveforms. A real-time monitoring and inferencing algorithm fuses these low-level decisions into an overall assessment of thesystem state. However, low-level decisions are prone to error due to improper threshold selection, lack of adequate physical models, electromagnetic interference, environmental conditions, etc. 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; on the other hand, a "FAIL" outcome of a test 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 addition, at any sampling time, the results of all test decisions in a system are not available due to varying sampling rates of sensors and signal processing limitations. Thus, the problem is one of determining the states (good/faulty) of components given a set of partial and unreliable tests over time, which is the subject of this paper.

A number of algorithms have been proposed for this large-scale fault isolation problem. One particularly efficient algorithm is TEAMS-RT, which is part of Qualtech's Intergerated Diagnostics Toolset [1]. Using fast enumeration techniques, TEAMS-RT classifies the systems components into Good, Bad, Suspect and Unknown states. As discussed below, TEAMS-RT can handle systems with thousands of components and test points in real-time. One missing feature in TEAMS-RT is the ability to handle test inaccuracies, that is, missed detections and false alarms in test outcomes. A missed detection occurs when a test gives out a PASS reading when the actual outcome should have been a FAIL due to a component failure. Similarly, a false alarm occurs when a test outcome is a FAIL when no underlying component has failed. Taking the probabilities of missed detections and false alarm in the diagnosis step can usually improve performance. That is, the reliabilities of the sensors can be used in the diagnosis process itself.

In order to rectify this, options were added to TEAMS-RT to handle unreliable tests [2]. This involves estimating the missed detection and false alarm probabilities of the sensors in real-time and using them in a Maximum Likelihood (ML) fashion for fault isolation [3]. If the reliabilities are not available, a Hamming distance type approach can be used to find a single or multiple faults [4]. This involves comparing the observed test outcome with the test dictionary and selecting the likeliest fault (or faults) that could have produced the observed sensor outputs.

Another approach for handling unreliable tests for single fault analysis is to use a HiddenMarkov Model (HMM) [5]. A HMM is capable of characterizing a doubly embedded stochastic process with an underlying stochastic process that, although unobservable (hidden), can be observed through another set of stochastic processes. In the fault diagnosis problem, the faulty states of the system are not observable directly, i.e., they correspond to the hidden part of the doubly embedded stochastic process. The hidden states of the system can be observed through another set of stochastic processes that produce the sequence of uncertain test outcomes. The key problem in diagnosis is to choose the most likely (hidden) state sequence, given the sequence of uncertain test outcomes. A HMM is a parametric model characterized by the state transition probabilities, the instantaneous probabilities of test outcomes given the system state and the initial state distribution. These parameters can be adaptively estimated by the well-knownBaum-Welch algorithm [6]. Therefore, an algorithm capable of considering these two problems of finding the most likely state sequence and of estimating model parameters within the same theoretical framework is required. HMM provide the required theoretical machinery. One major disadvantage of the HMM approach is its computational complexity that is not real-time feasible.

This paper reviews these different methods, which differ in computational complexity and fault isolation capability, and tests their performances on some representative large-scale systems. Test dictionaries of sizes up to 1000 x 1000 are to be handled. That is, results from 1000 tests measuring the state of 1000 components are available. However, at any time, only 10-20% of the test results are available. Then, the objective becomes the real-time fault diagnosis using incomplete and inaccurate test results with online estimation of test accuracies. It should also be noted that the test accuracies can vary with time --- one needs a mechanism to update them after processing each test result vector. The experiments are carried out usingQualtech's TEAMS-RT system simulation and real-time diagnosis tool.

This paper is organized as follows: Section 2 reviews basic TEAMS-RT inference engine. In Section 3, Hamming distance and Maximum Likelihood based diagnosis methods are presented. Section 4 summarizes the use of the HiddenMarkov model approach for single fault isolation. Each of these sections presents representative simulation results.

2. REAL-TIME DIAGNOSIS USING TEAMS-RT

Introducing TEAMS-RT

QSI's integrated tool set automates the tasks of Design for Testability (DFT), Reliability Analysis, FMECA, on-line monitoring and off-line diagnosis [1]. The software tool set consistsof :

- TEAMS: Testability assessment and improvement (DFT), reliability analysis, Failure Modes, Effects and Criticality Analyisis (FMECA) and pre-computed diagnostic test strategy generation in a variety of forms (e.g., SGML-based Interactive Electronic Technical Manual);
- TEAMS-RT: on-board diagnostics, health and usage monitoring systems (see Figure 1);
- TEAMATE: Portable Intelligent Maintenance Aids (PIMAs) with interactive electronic technical manuals and multi-media animation, dynamicTPSs for ATEs.
- TEAMS-KB: Scheduled and unscheduled maintenance and diagnostics data collection, statistical data analysis and data mining for trend and anomaly detection/isolation.

The TEAMS toolset implements a model-based reasoning approach, wherein information about failure sources, tests and monitoring points, redundancy and system modes are captured in colored directed graph models known asmultisignal models [7]. In simple terms, these models enable the inference engine to interpret test results by answering these

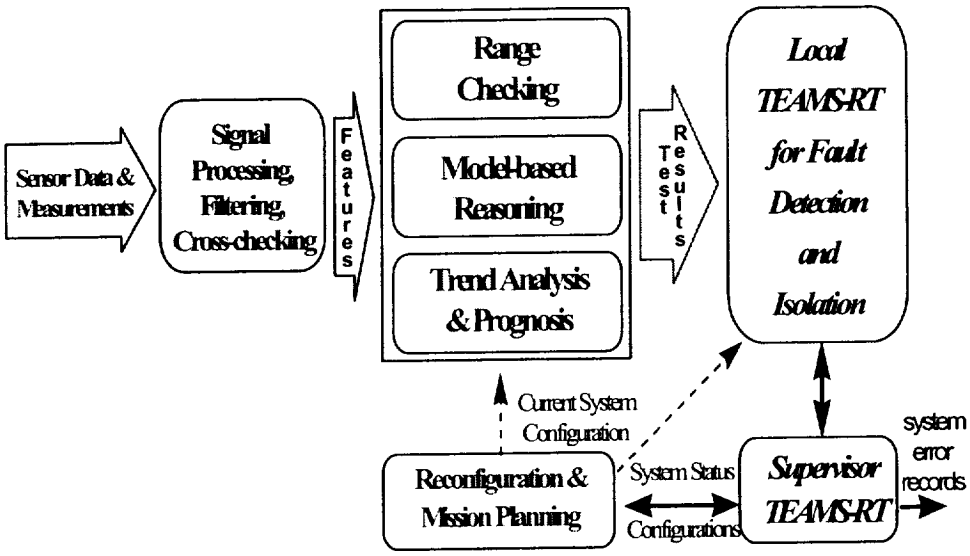

Figure 1: Real-time process monitoring using TEAMS-RT.

questions: given a test T1, which components can cause it to fail; or, if I want to check the health of component C1, which tests can observe it. Such models may be automatically generated via fault simulation (using simulators such as Saber, PSpice, VHDL simulators, MATRIX$_X$) or entered manually in TEAMS based on engineering understanding of the system or legacy data captured in FMECA reports, fault trees, CAD data, and technical documentation. The same model is then used by TEAMS-RT for onboard monitoring, and by TEAMATE for ground support systems, thus ensuring that the results predicted in the design stage by TEAMS are indeed achieved in actual application.

TEAMS-RT monitors the health of the target system by interpreting test results in real time. Such on-line tests are typically built-in self-tests and alarms, which often involve acquisition of data from sensors, filtering, estimation and decision-making. Thus, the actual deployment of TEAMS-RT also involves modules for data acquisition, filtering, and extraction of *features* (such as computing the mean, r.m.s. power or harmonic distortion of a observed response), and comparing the observed feature to (a range of) reference values to decide whether a test has passed or failed (see Fig. 1). In the rest of this section, we focus on the TEAMS-RT reasoning engine, whereas the signal and data processing module is presented in the following section.

The Basic TEAMS-RT Algorithm

The objective of the TEAMS-RT inference engine is to associate one of four distinct (failure) states with each component in the system: (1) *Good*, (2) *Bad*, (3) *Suspected*, and (4) *Unknown*. When TEAMS-RT is invoked, we assume that the state of all components is *Unknown*. If a test covering a component passes, its state is updated to *Good*. If a test covering a component fails, its state is *Suspected*. The *Bad* components are derived from these *Suspected* components by elimination of *Good* components. For notational simplicity, we define test signature Ts_j as a set of failure sources detectable by test t_j. The necessary information for fault diagnosis is stored in the following sets:

A is the set of *All* components,

B is the set of known *Bad* components,

S is the set of *Suspected* components,

U is the set of *Unknown* components,

G is the set of known *Good* components,

$F = \{ f_j \}$ is the signature of failed tests after removing the *Good* components from Ts_j.

The basic algorithm for the inference engine of TEAMS-RT is as follows:

Algorithm: TEAMS-RT Inference Engine

Step 1: Initialize:

Set state of all Faults to *Unknown*

$U=A, B=\emptyset, S=\emptyset, G=\emptyset, F=\emptyset.$

Step 2: Process Passed Tests:

i. Find the union of test signatures of passed tests,
$\cup_{tj\ passed}\ Ts_j$.

ii. Find new good components using the union of test signatures of passed tests:
$\Delta G \leftarrow (\cup_{tj\ passed}\ Ts_j) - G$

iii. Update Fault sets - remove good components from *Suspect* and *Unknown* sets
$G \leftarrow G \cup \Delta G, S \leftarrow S - \Delta G, U \leftarrow U - \Delta G.$

Step 3: Process Failed Tests:

i. Store failure sub-signatures pendingresolution
$F = \{ f_k \} \leftarrow \{ Ts_k - G \}$

ii. Add *Unknown* covered components to the set of *Suspected* components:
$S \leftarrow S \cup \{ f_k \}, U \leftarrow U - \{ f_k \}.$

Step 4: Process unresolved failure sub-signatures:

i. Update the "unexplained" failed test set F by removing the new *Good* components.
$F = \{ f_j \} \leftarrow \{ f_j - \Delta G \}$

ii. Update *Bad* component list B by identifying *one-for-sure Bad* components
If $|f_j| = 1$, $B \leftarrow B \cup f_j, \Delta B \leftarrow \Delta B \cup f_j$.

iii. Remove sub-signatures explained by newly identified *Bad* components
If $f_k \cap \Delta B \neq \emptyset$, remove f_k from F, since it
is now explained by ΔB.

Capabilities and Performance of TEAMS-RT

The inference engine algorithm outlined above is at the core of TEAMSRT's efficiency. The production version of TEAMS-RT includes additional capabilities for dynamic system modechanges, and capability of diagnosis and prognosis in fault-tolerant systems with built-in redundancy. Some unique features of TEAMS-RT are: () efficient real-time processing of sensor results, (ii) update of fault - test-point dependencies in response to system mode changes, and (iii) update of dependencies resulting from failures in redundant components. Table 1 presents simulation results for TEAMS-RT on a 1000×1000 system with 80 modes of operation. Column 1 lists the number of faults inserted. |Tp| is the number of tests that passed in spite of the failures. The remaining columns list the number of components that were declared to be good, bad and suspected (residual ambiguity) by TEAMS-RT, and the processing time. Similar timings were observed in the X33-IPTD test-stand [8].

Table 1: Performance results of TEAMS-RT for simulated system with 1000 faults and tests.

faults	|Tp|	Good	Bad	Suspect	Time(ms)
1	**993**	**997**	**1**	**2**	**50**
2	**978**	**996**	**2**	**2**	**50**
5	**931**	**991**	**5**	**4**	**50**
10	**881**	**983**	**10**	**7**	**75**
20	**819**	**973**	**20**	**7**	**87**

3. HAMMING AND MAXIMUM LIKELIHOOD DECODING FOR FAULT ISOLATION

Introduction to ML Decoding

The objective here is to develop a real-time capable processor to detect and isolate component failures in large-scale systems in the presence of inaccurate test results. A sequence of imperfect test results (as a row vector of 1's and 0's) are available to the processor. In this case, the problem is to recover the uncorrupted test result vector and match it to one of the rows in the test dictionary, which in turn will isolate the faults.

In order to recover the uncorrupted test result vector, one needs the accuracy of each test. That is, its detection and false alarm probabilities are required. In this problem, their true values are not known and, therefore, have to be estimated online. Other major aspects in this problem are the large-scale nature and the real-time capability requirement. Test dictionaries of sizes up to 1000 x 1000 are considered. That is, results from 1000 tests measuring the state of 1000 components are available. However, at any time, only 10-20% of the test results are available. Then, the objective of the current work becomes the real-time fault diagnosis using incomplete and inaccurate test results with online estimation of testaccuracies. It should also be noted that the test accuracies can vary with time --- one needs a mechanism to update them after processing each test result vector.

In general, let there be m components and n different tests. Assume that, at any time t_k when the k-th frame (incomplete test result vector) of test results are obtained, results from only n' tests are available. Using this frame of test results, one needs to diagnose the component failures, if any, as well as update test accuracy information. Figure 2, where $j_1, j_2, j_{n'}$ are the indices of the tests available at time t_k, schematically shows the input/output relationship.

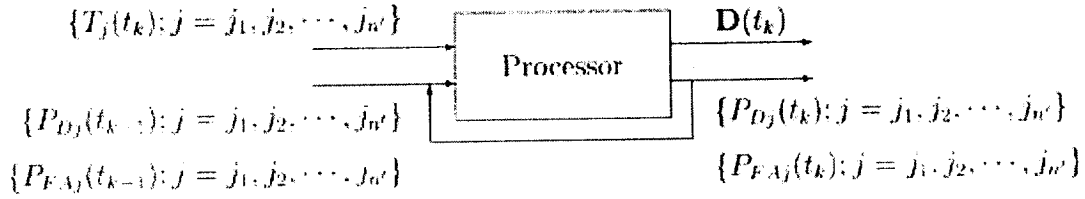

Figure 2: Input to and output from the processor for real-time fault diagnosis.

The diagnostic output $\boldsymbol{D}(t_k)$ from the processor depends on how the problem is formulated and the processor implemented. For example, $\boldsymbol{D}(t_k)$ may yield the uncorrupted test result vector which can be used to decide component failures. Alternatively, when the uncorrupted test result vector cannot be identified, one may obtain a set of probable uncorrupted test vectors with corresponding confidences in them. For the configuration shown in Figure 2, the latest incomplete test vector obtained at time t_k is used as the input. One can also use the set of last l test vectors as the input, in which case, the processor operates in batch mode. These various options for input and output are discussed in detail in the sequel.

Other issues that need attention are how to initialize the test accuracies and how to update them in view of the latest test result vector and the corresponding diagnosis. The test accuracies for test j are measured in terms of the detection probability P_{Dj} and false alarm probability P_{FAj}. These probabilities are defined as

$$P_{Dj} = \Pr\{T_j = FAIL \mid R_j = FAIL\}$$

$$P_{FAj} = \Pr\{T_j = FAIL \mid R_j = PASS\}$$

where R_j is the actual test result one would obtain for test j in the absence of any test errors.

Finally, one needs to address the distinction between test errors and component failures. In the presence of test errors, especially when results from all the tests are not available, one has to identify whether the received test vector is due to a component failure or due to a test error. Due to test errors, the received test vector can correspond to a valid component failure signature although there arent any such failures actually. Unnecessary and costly component repairs and replacements are required in the absence of such a distinction. Intermittent component failures also necessitate the need for distinction. Ideally, one would like to be able to identify intermittent failures and not decide them as permanent component failures.

With the above, the primary objectives of this work are as follows:

1. Distinction between test errors and component failures.
2. Detection and, if possible, correction of test errors.
3. Component fault isolation based on 1 and 2.
4. Online estimation of testaccuracies.

Maximum Likelihood Decoding

Denote the test vector received at time t_k by $z(t_k)$, which is given by

$$z(t_k) = \{T_j(t_k); j = j_1, j_2, \ldots, j_{n'}\}$$

For fault-diagnosis, one needs to find the uncorrupted test result vector

$$r(t_k) = \{R_j(t_k); j = j_1, j_2, \ldots, j_{n'}\}$$

When $r(t_k)$ cannot be recovered with certainty, a set of probable $r(t_k)$ values need be obtained In order to understand the solution methodology, consider the process of receiving $z(t_k)$ and how it relates to the test matrix. The binary test matrix D consisting of m unique rows relates the component failures to actual test outcomes. The i-th row d_i of D gives the tests that will fail if componenti fails --- $d_{ij} = 1 = FAIL$ indicates that test j will fail if component$i$ fails. Then $r(t_k)$ is the subset of the actual results for the tests that are available at time t_k. Due to the non-unity detection probabilities and non-zero false alarm probabilities, the outcomes in $r(t_k)$ may be corrupted and received differently as $z(t_k)$. The transition from $R_j(t_k)$ to $T_j(t_k)$ is illustrated in Figure 3. Thus the problem is to recover $r(t_k)$, and, effectively $d_i(t_k)$ from $z(t_k)$.

A simple solution is to select the row d_i in D that has the minimum Hamming distance to the received vector. While this approach can work satisfactorily when there is enough redundancy in the test matrix and most of the test results are available at all times, this will lead to ambiguous decoding with incomplete test results. The problem here is that there will be a number of rows in the test matrix D that have the same Hamming distance (differs by the same number of bits) from the received vector and, therefore, only error detection, not error correction, is possible. The optimal solution is to use a Maximum-Likelihood (ML) estimator, which yields the row in the test matrix that best matches the received imperfect test results, given the test accuracies [3].

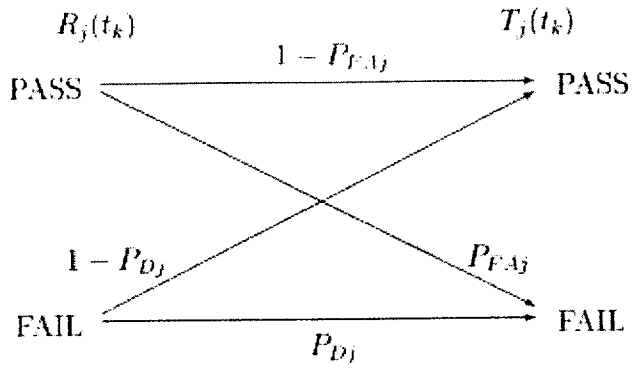

Figure 3: Test errors and their probabilities.

Since true test accuracies are not available, the estimated ones have to be used. Then

$$\hat{d}(t_k) = \arg\max_{d_i}\{p[z(t_k) \mid d_i, \{\hat{P}_{Dj}(t_{k-1}), \hat{P}_{FAj}(t_{k-1}); j = 1,\ldots,n\}]\}$$

Assuming that the errors in test j_1 are independent from those of test j_2 for $j_1 \neq j_2$, one can rewrite the above as

$$\hat{d}(t_k) = \arg\max_{d_i}\Big\{ \prod_{j = j_1,\ldots,j_{n'}} p[T_j(t_k) \mid d_{ij}, \hat{P}_{Dj}(t_{k-1}), \hat{P}_{FAj}(t_{k-1})]\Big\}$$

which is equivalent to

$$\hat{d}(t_k) = \arg\max_{d_i}\Big\{ \sum_{j = j_1,\ldots,j_{n'}} \ln\{p[T_j(t_k) \mid d_{ij}, \hat{P}_{Dj}(t_{k-1}), \hat{P}_{FAj}(t_{k-1})]\}\Big\}$$

The individuallikelihoods can be derived from Figure 3 as

$$p[T_j(t_k) = PASS \mid d_{ij} = PASS, \hat{P}_{Dj}(t_{k-1}), \hat{P}_{FAj}(t_{k-1})] = 1 - \hat{P}_{FAj}$$
$$p[T_j(t_k) = PASS \mid d_{ij} = FAIL, \hat{P}_{Dj}(t_{k-1}), \hat{P}_{FAj}(t_{k-1})] = 1 - \hat{P}_{Dj}$$
$$p[T_j(t_k) = FAIL \mid d_{ij} = PASS, \hat{P}_{Dj}(t_{k-1}), \hat{P}_{FAj}(t_{k-1})] = \hat{P}_{FAj}$$
$$p[T_j(t_k) = FAIL \mid d_{ij} = FAIL, \hat{P}_{Dj}(t_{k-1}), \hat{P}_{FAj}(t_{k-1})] = \hat{P}_{Dj}$$

With these values, the maximum likelihood estimator has to enumerate all the possibleoutcomes (all single fault conditions and the no-fault condition), evaluate theirlikelihoods and select the best row. Then, the problem becomes that of selecting the best row from a codebook of size $(m+1)$ by n.

The advantage of above approach is that it yields the optimum performance by making use of the relative reliabilities of the different tests as well as the difference between the miss probability$1-P_D$ and the false alarm probabilityP_{FA}. In contrast, the Hamming distance-based decoder assumes that all tests are equally reliable and the two error probabilities for a given test are the same. In fact, it can be shown that when the tests are equally reliable and the two error types are equally probable, the

Hamming decoder is optimal. In real systems, not all the tests in the system are equally accurate. In addition, depending on the nature of the test, one error type (for example, false alarm) may be more likely than the other In this case, the use of test accuracy information will yield superior fault detection and isolation.

The major disadvantage of using accuracy information is the added computation load due the need for floating point operations whereas the Hamming decoder requires only integer operations. It also requires the online estimation of test accuracies, which increases the computational load. However, in real systems test accuracies can vary with time (for example, due to wear and tear) and online estimation is essential for accurate decoding.

Online Test Accuracy Estimation

The unknown detection and false alarm probabilities of the tests have to be updated after processing each received test vector. Naturally, how the probabilities are updated depends on the past history as well as the latest test vector. The update mechanism should be such that it is not too sensitive to spurious test errors while giving more importance to the recent test results in view of the time-varying nature of test accuracies. Thus, the updated probabilities are evaluated as the weighted average of the past and the present probabilities. Since one has to take the time-varying nature into account, the update is carried out using a sliding-window approach.

In the following the test index j and the estimate notation "^" have been omitted for simplicity. Let $P_D(t_0)$ and $P_{FA}(t_0)$ be the estimates for the detection and false alarm probabilities, respectively, at some known time t_0 in the past. Also, let $P_{Dw}(t_0, t_k)$ and $P_{FAw}(t_0, t_k)$ be the corresponding probability estimates in the time interval (t_0, t_k). Then the weighted probability update can be written as

$$P_D(t_k) = (1-\alpha_D(t_0,t_k))P_D(t_0) + \alpha_D(t_0,t_k)P_{Dw}(t_0,t_k)$$
$$P_{FA}(t_k) = (1-\alpha_{FA}(t_0,t_k))P_{FA}(t_0) + \alpha_{FA}(t_0,t_k)P_{FAw}(t_0,t_k)$$

where the weighting parameters $\alpha_D(t_0, t_k)$ and $\alpha_{FA}(t_0, t_k)$ depend on how much importance one wants to place on the latest probability estimates. Typically, that would depend on how much data is available to update the probabilities within the interval (t_0, t_k). Note that the initial time t_0 need not really correspond to the estimation starting time. In order to handle time-varying error probabilities, one can move the initial time t_0 with a sliding window and advance it. This amounts to reinitializing the probability estimates and then updating them at regular intervals, say t_w, where t_w is the maximum width of the sliding window. The sliding window approach is illustrated in Figure 4.

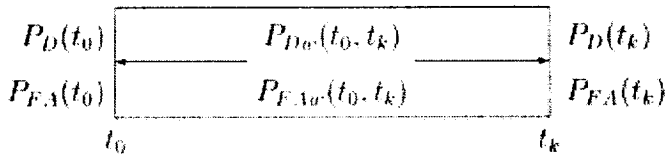

Figure 4: Updating test accuracies.

In the above, one has the option of choosing the weighting parameters $\alpha_D(t_0, t_k)$ and $\alpha_{FA}(t_0, t_k)$ in a number of ways. Intuitively, the more data one has in the interval (t_0, t_k) to update the detection and false alarm probabilities, the higher the values of the corresponding weighting parameters. For example, the weighting parameters can be proportional to the available data as shown in Figure 5.

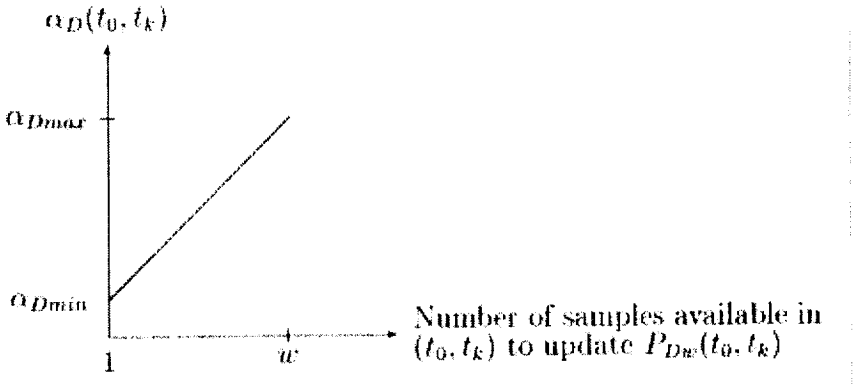

Figure 5: Linear weighting parameter for updating P_D.

The weighting parameters for updating the detection probability and the false alarm probability may be different in view of the different number of samples that can be used to update them. The online estimation approach requires the initial estimates for P_D and P_{FA}. These can be obtained using a Hamming distance-based decoder. For example, the first l test result vectors can be used for initializing the estimates. In this case, one assumes that the probabilities are unknown constants [3].

As noted earlier, one advantage of using test accuracy information is that it provides better resolution in recovering the uncorrupted test vector. With Hamming distance, one may not be able to resolve the test vector for there may be a number of rows in the test matrix with the same distance to the received vector. With the online estimation of test accuracies, this is still possible, albeit less frequently. When there are multiple rows with the same maximum likelihoods, one needs a mechanism to decide which of those rows resulted in the received vector. Another important issue is the distinction between test errors and component failures. This becomes an issue when there are multiple rows, including the one corresponding to no-failure condition, with the same maximum likelihood.

If a component's failure statistics, for example, the mean time between failures, their average durations, etc., are known, the distinction is formulated as a hypothesis testing problem where the comparison is test error vs. failure in one of the components. For the current work, the fault statistics are not known. Therefore, in this case, only the past outcomes from the decoding/updating step can be used to make the distinction.

Performance of the Decoders

The program has been tested with a dictionary of size 1024 x 1000. The testbed to simulate systems of different complexity and parameters has been developed. Preliminary results indicate perfect (100% correct at 1-10% sparsity when all test results are available) decoding at 100ms per test result vector (including data generation, decoding and saving results) on a Pentium Pro processor running at 300MHz. The major advantage of using test accuracy information is the reduction in the ambiguity set size (the set of components that are flagged as suspects — ideally the ambiguity set size should be one). That is, the number of rows in the dictionary with the maximum likelihood estimator are fewer and, therefore, one can pin point the faulty component more accurately. The size of the ambiguity set increases when results from fewer tests are available or when the sparsity of the test matrix is lower (low system redundancy). In Table 2, the ambiguity set sizes of the Hamming-distance based decoder and the maximum likelihood estimator are shown against the percentage of test results available at any time. The sparsity of the test matrix is 1%.

Table 2: Performance of the Hamming and ML decoders.

Test Availability	Hamming Decoder		ML Estimator	
	Ambiguity Size	CPU Time	Ambiguity Size	CPU Time
100%	1.01	100ms	1.0	100ms
80-90%	1.30	90ms	1.2	90ms
50-60%	4.95	75ms	2.6	85ms
30-40%	17.30	65ms	6.5	85ms
20-30%	31.5	65ms	17.7	85ms

In Table 2, the CPU times include the time for data generation and, saving the data and results to the disk. Results for the distinction between component failures and test errors are forthcoming. These proof-of-concept results are only preliminary. More improvements are possible by distinguishing between intermittent and total failures whenever a component failure is declared. More rigorous and complete testing using the testbed with different sparsities and test availability percentages is required.

5. HMM BASED DIAGNOSIS

Introduction to HMM

The use of hidden Markov models to solve the fault diagnosis problem requires the solution of two specificsubproblems:

1. *The decoding subproblem* of finding the most likely sequence of system states; and
2. *Model parameter estimation subproblem* of updating the model parameters to the most likely values according to the observed test sequence.

Figure 6 illustrates the workflow in HMM-based fault diagnosis framework. State transition probabilities and initial state probabilities are evaluated (estimated) either via component failure/repair rates or viaBaum-Welch method. Detection and false alarm probabilities are estimated from the test pass-fail history based on the Central Limit Theorem. Instantaneous probabilities of test outcomes given the system state are functions of the fault dictionary matrix, and the detection and false alarm probabilities of tests. Thus, these probabilities can be estimated, based on estimating the detection and false alarm probabilities of tests. Once the HMM parameters are available, a sliding windowViterbi algorithm is employed to find the optimal system state sequence based on the observed test sequence.

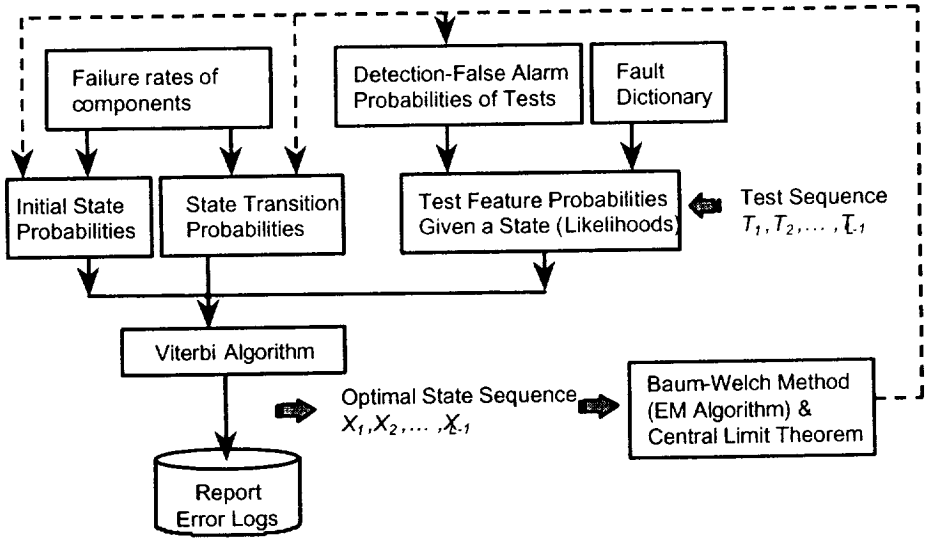

Figure 6: Overview of online fault diagnosis using HMM.

Performance of HMM Based Diagnosis

In this example, we consider a system with $m = 340$ failure sources and $n = 256$ tests. We assume that at each sampling time, about *10%* of the total tests are available. The test detection probability is assumed to be in the range *(0.9 1.0)*, while the false alarm probability is assumed to be in the range *(0.0, 0.1)*. The component mean time to failure, measured in hours, is assumed to be in the range [125, 1000]. The component mean time to repair, measured in hours, is in the range [0.5, 1].

Figure 7 compares the error rates as a function of detection and false alarm probabilities. As expected, the error rates increase as the reliabilities of tests decrease. The dark bars represent the results of HMM with known system parameters. Since they have maximum information, these models provide low error rates. The light bars represent models with adapted parameters based on observations. The error rates are about 1% ~ 2% higher than those with known HMM parameters.

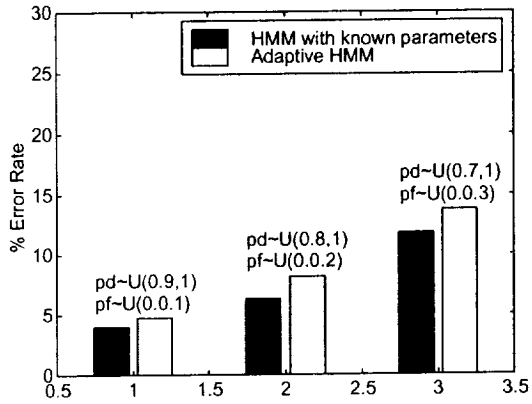

Figure 1: Error rate vs. detection - false alarm probabilities.

6. SUMMARY

In this paper, we reviewed a number of algorithms that can be used for fault diagnosis and isolation inlargee-scale systems (e.g., systems with 1000 components and 1000 sensors). This was motivated by the need for real-time capable algorithms that can process large amounts of sensor data. A more comprehensive comparison based on a standard set of systems models is recommended for further analysis. This can be done using the TEAMS-RTestbed from Qualtech, which includes the algorithms presented in this paper.

REFERENCES

1. S. Deb *et al,* "QSI's Integrated Diagnostics Toolset," *Proc. IEEE AUTOTESTCON,* Anaheim, CA, 1997.
2. T. Kirubarajan, "Real-time Fault Diagnosis Using Imperfect Test Results: Integration of Hamming and Maximum Likelihood Decoding into TEAMS-RT", Report to Qualtech, June 1999.
3. Y. Bar-Shalom, X. R. Li and T. Kirubarajan, *Estimation with Applications to Tracking and Navigation: Algorithms and Software for Information Extraction*, New York, NY: John Wiley & Sons, 2001.
4. B. Sklar, *Digital, Communications: Fundamentals and Applications*, Englewood Cliffs, NJ: Prentice Hall, 1988.
5. J. Ying; T. Kirubarajan, K. R. Pattipati and A. Patterson-Hine, "A Hidden Markov Model-Based Algorithm for Fault Diagnosis with Partial and Imperfect Tests", IEEE Transactions on Systems, Man and Cybernetics, Part C, Vol. 30, pp. 463-473, Nov. 2000.

6. L.E. Baum, "An Inequality and Associated Maximization Technique inStattistical Estimation for Probabilistic Functions of a Markov Processes", *Inequalities*, 3, 1-8.
7. S. Deb *et al*, "Multi-Signal Flow Graphs: A novel Approach for System Testability Analysis and Fault Diagnosis," *Proc. IEEE AUTOTESTCON*, Anaheim, CA, pp. 361-373, Sept. 1994.
8. M. Holthaus "Model Documentation for CLIN 0001, Engineering Support Under Qualtech's NASA Contract Entitled "Multisignal Flow Graphs for System Fault Diagnosis," Rockwell Aerospace, Final Report, May 1996.

Qualtech Systems, Inc.

100 Great Meadow Rd., Wethersfield CT 06109 Tel./Fax: (860) 527-8014/8312
E-mail: info@teamqsi.com

May 17, 2001

NASA Center for Aerospace Information (CASI)
Attn.: Accessioning Dept.
7121 Standard Drive
Hanover, MD 21076

Subject: Contract No.: NAS2-99048
SBIR 1999: Final Technical Progress Report

Dear Sir/Madam:

Please find enclosed two copies of the Final Technical Progress Report on the following contract:

Contract No.: NAS2-99048:
"An Onboard Real-Time Aircraft Diagnosis and Prognosis System".

If you need any additional information, please contact me at (860) 257-8014.

Sincerely yours,

Somnath Deb
Chief Scientist
Qualtech Systems, Inc.