# Multiple Fault Diagnosis Using Factored Evolutionary Algorithms

*John W. Sheppard and Shane Strasser*

When supporting commercial or defense systems such as aircraft avionics, guidance and control, electronic warfare, or propulsion systems, a perennial challenge is providing effective test and diagnosis strategies to minimize downtime, thereby maximizing system availability. One can argue that one of the most effective ways to maximize downtime is to be able to detect and isolate as many faults that either exist or are emerging in a system at one time as possible. This is referred to as the "multiple-fault diagnosis" problem, and it is a problem that is known to be computationally intractable (i.e., NP-complete) [1]. While several tools have been developed over the years to assist in addressing the multiple-fault diagnosis problem, considerable work remains to provide the best diagnosis possible given the information collected through observations, gripes, test results, and historical data. Recently, a new model for evolutionary computation has been developed called the "Factored Evolutionary Algorithm" (FEA) [2]. In FEA, a target optimization problem is broken down into subproblems that exhibit some kind of overlap. Then the optimization algorithm of choice (e.g., simulated annealing, genetic algorithm, particle swarm optimization) is applied to each of the subproblems, and the subproblems periodically share information with neighboring subproblems along the points of overlap.

One instantiation of FEA is known as Overlapping Swarm Intelligence (OSI) [3], and this method has been applied to the problem of abductive inference in Bayesian networks [4]. Both full and partial abductive inference in Bayesian networks are known to be NP-hard [5]-[7], yet OSI has been demonstrated to provide state-of-the-art results in performing abductive inference on several medium and large scale Bayesian networks that are available for test purposes.

In this paper, which is an extended version of [8], we combine our prior work in deriving diagnostic Bayesian networks from static fault isolation manuals and fault trees [9] with the FEA strategy to performing abductive inference. Previously, we compared a variety of search techniques, including simple hillclimbing, genetic algorithms (GA), and particle swarm optimization (PSO) with a more traditional approach to abductive inference, and results indicate that these methods should perform well on multiple-fault diagnosis as well. We demonstrate the effectiveness of our FEA approach to addressing the multiple-fault diagnosis problem on several networks derived from a model commonly used in the literature as well as several existing Fault Isolation Manuals (FIMs) used for maintaining an actual military aircraft. As extended work, we also perform an in-depth analysis of the effect of hidden failures in the diagnostic model as a means of explaining the rather surprising behavior from the initial experiments. We find that the presence or absence of hidden failures has a significant impact on overall multiple-fault diagnostic performance, and that our approach to deriving Bayesian networks from fault trees helps to mitigate the issues associated with hidden failures by providing clear signatures for each fault.

This paper provides background information necessary to understand our approach, including a brief introduction to the multiple-fault diagnosis problem and how multiple-fault diagnosis relates to Bayesian abductive inference. There is also an introduction to FEA. Once the basic concepts are in place, we review related work, discuss our technical approach, present the results of our experiments and discuss the implications of our experiments. We wrap up the paper with a review of the main conclusions and a discussion of future work.

## Background

To provide context, we begin with some formalities. First, we need to address the question of what constitutes multiple-fault diagnosis. While a relatively straightforward concept, we decided to be a bit more formal in our definition to avoid confusion. We then proceed to introduce the underlying mathematical model that defines, not only what doing multiple-fault diagnosis is, but also explains why it is such a difficult problem. Finally, we present our approach to solving the problem by introducing "Factored Evolutionary Algorithms"

The research presented in this paper is a scientific extension of the work presented at *IEEE AUTOTESTCON 2017* [8].

1094-6969/18/$25.00©2018IEEE

(FEA). All three of these topics have been discussed at length in our previous work, so much of what we discuss here is limited to a brief overview.

## Multiple Failure Diagnosis

In previous work, we provided a formal treatment of the multiple-fault diagnosis problem [1]. For completeness, we summarize those results here by defining what we mean by a fault, a test, a fault signature, and ultimately a multiple fault and its signature.

For our purposes, a fault is any specific cause of a system not being able to perform its intended function, and a test is a means of observing the behavior of the associated fault. A multiple fault corresponds to the presence of more than one fault in the system. We say that the set of test observations associated with each fault represents that fault's "signature." We can represent that signature as a vector of truth values where TRUE indicates that a test can detect the fault and FALSE indicates it cannot. Given this notion of a "fault signature," which consists of a set of test results that either pass or fail, the signature of a multiple fault then corresponds to the logical OR of the signatures for the individual faults.

Using this definition of a multiple fault, the basic approach to performing multiple-fault diagnosis can be reduced to finding the minimum set of faults in a system whose combined signature best explains the test results obtained. Unfortunately, using this definition, we showed previously that the minimum set covering problem, a known NP-complete problem [10], can be reduced to the multiple-fault diagnosis problem. Thus, finding the minimum sized multiple fault set to explain a set of test results is itself NP-complete.

## Bayesian Networks and Abductive Inference

The work discussed in this paper makes use of our additional prior work attempting to develop approaches to efficient abductive inference in Bayesian networks [4]. The approach described here involves mapping a diagnostic problem onto a Bayesian network and then performing abductive inference over that network. A Bayesian network is a directed acyclic graph where each vertex in the graph corresponds to a random variable, and each edge between two variables $X_i \rightarrow X_j$ denotes a probabilistic relationship between these variables corresponding to $P(X_j \mid X_i)$.

In general, a Bayesian network encodes a joint probability distribution over $n$ random variables $\mathbf{X} = \{X_1,\ldots,X_n\}$ such that

$$\mathrm{P}(\mathbf{X}) = \prod_{X_i \in \mathbf{X}} P\left(X_i \mid \mathrm{Pa}(X_i)\right) \tag{1}$$

where $\mathrm{Pa}(X_i)$ denotes the parents of node $X_i$.

Considerable work has been performed developing Bayesian networks for diagnostic applications; however, such networks often exhibit substantial complexity. As a method for combatting this complexity, Schwe *et al*. observed that the Noisy-Or formalism not only reduces the size of the network but more closely matches assumptions made in performing diagnosis [11], [12].

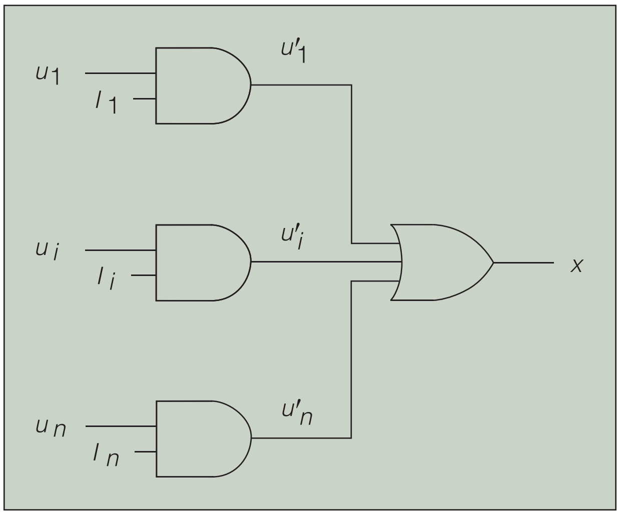


***Fig. 1.*** Line failure model for Noisy-Or in Bayesian networks.

The Noisy-Or formalism makes two assumptions and represents these assumptions using the "line failure model" as shown in Fig. 1. First, the *accountability* assumption says that at least one parent node must account for (i.e., explain) a child node being observed to be true. This is key for treating fault diagnosis as abductive inference since the child nodes are test nodes and the parent nodes are fault nodes. Second, the *exception independence* assumption says that exceptions to normal behavior are independent of one another.

We can represent this model through a series of AND gates for each parent, where the inputs include the cause and a negated mechanism representing inhibition of proper functioning. The output of these AND gates then feed an OR gate, which in combination yields the Noisy-Or model. We see this representation in Fig. 1, where we can think of $u_i$ as being a failure mechanism, and each value $I_i$, the failure inhibition mechanism, is drawn from a binary probability distribution. These inhibitor probabilities are what gives the Noisy-Or model its name, in that they introduce noise to an otherwise deterministic function. Then $u_i'$ provides the output of a corresponding AND gate. Finally, we define a function $F(x)$ that corresponds to a standard OR gate, giving us $F(x) = \bigvee_i u_i'$. Finally, Pearl provided a method for computing the conditional probability tables on the fly using this model, thereby reducing the complexity of the overall model.

Given a Bayesian network, the abductive inference problem involves assigning evidence (i.e., making observations) to a subset of variables in the network and then determining the most probable values to assign to a subset of the remaining variables. More formally, let $\mathrm{X}_O \subset \mathbf{X}$ be a set of variables for which we have observed states values, and let $\mathbf{X}_U \subseteq \mathbf{X} \setminus \mathbf{X}_O$ be a set of variables that are unknown. Then abductive inference, also sometimes referred to as the most probable explanation (MPE) problem, corresponds to the problem of finding state assignments for $\mathbf{X}_U$ such that

$$\mathrm{MPE}\left(\mathbf{X}_U, \mathbf{x}_O\right) = \underset{\mathbf{x}_U \in \mathbf{X}_U}{\arg\max}\, P\left(\mathbf{x}_U \mid \mathbf{x}_O\right) \tag{2}$$

The $k$-MPE problem attempts to find the $k$ most probable state assignments over $\mathbf{X}_U$.

If $\mathbf{X}_U = \mathbf{X} \setminus \mathbf{X}_O$, then the problem is referred to as the "full" abductive inference problem, which is the MPE problem. If, on the other hand, $\mathbf{X}_U \subset \mathbf{X} \setminus \mathbf{X}_O$, we have what is referred to as the "partial" abductive inference problem, also referred to as the maximum *a posteriori* (MAP) explanation problem. Unfortunately the full abductive inference problem was proven to be NP-hard by Shimony [7], and even constant-factor approximate inference was proven to be NP-hard for partial abductive inference [6].

An extended and significantly more difficult problem is referred to as the "most relevant explanation" (MRE) problem. For the MRE problem, the task involves determining state assignments for only a subset of $\mathbf{X}_U$ that maximizes the posterior probability, where which and how many of those variables to assign is unknown *a priori*. In particular, the MRE problem has been conjectured to be $NP^{PP}$-hard because of the exponential search space associated with determining which variables to instantiate, combined with the abductive inference problem [13].

The question of complexity is relevant to us because we can regard the multiple-fault diagnosis problem as abductive inference. In particular, since our approach involves mapping the diagnostic problem itself to a Bayesian network, we find ourselves faced with performing abductive inference on this network (whether MAP, MPE, or MRE) at the outset. Also given the fact this is a very real problem facing organizations having to perform complex system maintenance, coupled with the fact real-world test systems struggle due to uncertainty inherent in the test process [14], there is a real need to come up with effective approximation techniques to solve this problem, in spite of the discouraging complexity results. Developing an approach to meet that need is the focus of this paper.

## *Factored Evolutionary Algorithms*

Factored Evolutionary Algorithms (FEA) [2] are a new model of cooperative co-evolutionary optimization [15]-[17] that subdivides a problem to be optimized into subproblems and then optimizes the subproblems individually. We call the subproblems "factors" to relate them to the process of factorizing a function. The basic approach assumes that the factors overlap, meaning that different factors may share variables. This establishes a "communication" network between the factors. FEA then proceeds by iterating through a process of search/update, competition, and sharing of information between the factors.

More formally, suppose we are given a function $f : \mathbf{D}^N \to \mathbb{R}$, and we wish to optimize this function over the set of parameters $\mathbf{X} = (X_1,...,X_N)$. Now let $\mathbf{S}_i$ be a subset of $\mathbf{X}$. A factor can then be defined over the parameters in $\mathbf{S}_i$ that are optimizing $f$.

FEA typically employs a population-or swarm-based approach to optimize the function $f$. Because each population or swarm is only optimizing over a subset of values in $\mathbf{X}$, the factor defined for $\mathbf{S}_i$ needs to know the values of $\mathbf{R}_i = \mathbf{X} \setminus \mathbf{S}_i$ for local fitness evaluations. These values are determined from the other factors, thus enabling all of the values to be combined to optimize the whole function. The algorithm accomplishes this through competition and sharing.

First, a competition is held to find which factor holds the best state assignment for each dimension in $f$. Here, we summarize the competition algorithm described by Strasser *et al*. [2] (Algorithm 1). FEA constructs a full global solution $\mathbf{G} = (X_1,...,X_N)$ to evaluate the combined factors. Then for each $X_i \in \mathbf{X}$, the process iterates over the factors in a greedy fashion to find the best values. Next, sharing (Algorithm 2) allows overlapping factors to introduce knowledge from the global solution back into other factors. It also sets the values from $\mathbf{R}_i$ to those in the full global solution $\mathbf{G}$ so that each factor can evaluate its partial solution on $f$. When combining these processes with an appropriate underlying optimization algorithm and iterating (Algorithm 3), we get the full FEA process.

**Algorithm 1** FEA Compete

```
Input: Function $f$ to optimize, factors $S$, full global solution $G$
Output: Full solution $G$
 1: $randVarPerm \leftarrow$ RandomPermutation($N$)
 2: for $ranVarIndex = 1$ to $N$ do
 3:   $i \leftarrow randVarPerm[ranVarIndex]$
 4:   $bestFit \leftarrow f(G)$
 5:   $bestVal \leftarrow P_1[X_i]$
 6:   $S_i \leftarrow \{ S_k \mid X_i \in S_k \}$
 7:   $randPopPerm \leftarrow$ RandomPermutation($|S_i|$)
 8:   for $ranPopIndex = 1$ to $|S_i|$ do
 9:     $P_j \leftarrow S_i[randPopPerm[ranPopIndex]]$
10:     $G[X_i] \leftarrow P_j[X_i]$
11:     if $f(G)$ is better than $bestFit$ then
12:       $bestVal \leftarrow P_j[X_i]$
13:       $bestFit \leftarrow f(G)$
14:     end if
15:   end for
16: $G[X_i] \leftarrow bestVal$
17: end for
18: return $G$
```

**Algorithm 2** FEA Share

```
Input: Full global solution $G$, factors $S$
Output: Updated factors $S$
 1: for all $P_i \in S$ do
 2:   for all $X_j \in R_i$ do
 3:     $R_i[X_j] \leftarrow G[X_j]$
 4:   end for
 5:   $p_w \leftarrow P_i$.worst()
 6:   for all $X_j \in S_i$ do
 7:     $p_w[X_j] \leftarrow G[X_j]$
 8:   end for
 9:   $p_w.fitness \leftarrow f(p_w \cup R_i)$
10: end for
11: return $S$
```

### ICPSO

As mentioned, virtually any stochastic search algorithm can be applied with FEA. Since the abductive inference problem is inherently discrete, we applied a new discrete PSO algorithm that we developed previously and found to be effective on such problems—the Integer and Categorical Particle Swarm Optimization (ICPSO) algorithm [18]. In normal PSO, particles are defined to represent the states of the various variables being optimized. In ICPSO, we define a particle $p$ as $\mathbf{X}_p = [D_{p,1}, D_{p,2}, \ldots, D_{p,n}]$, where each $D_{p,i}$ denotes the probability distribution for variable $X_i$. Specifically, each dimension of the particle's position vector corresponds to a set of distributions $D_{p,i} = \left[d^a_{p,i}, d^b_{p,i}, \ldots, d^k_{p,i}\right]$, where $d^j_{p,i}$ is the probability that $X_i$ in particle $p$ takes on value $j$.

A particle's velocity then becomes a matrix made up of n vectors $\phi$, one for each variable:

$$\mathrm{V_p} = \left[\phi_{\mathrm{p},1}, \phi_{\mathrm{p},2}, \ldots, \phi_{\mathrm{p,n}}\right] \tag{3}$$

$$\phi_{p,i} = \left[\psi^a_{p,i}, \psi^b_{p,i}, \ldots, \psi^k_{p,i}\right] \tag{4}$$

where $\psi^j_{p,i}$ is particle $p$'s velocity for variable $i$ in state $j$. The velocity and position update equations are identical to those of the gBest PSO and are applied directly to the values in the distribution.

$$\mathbf{V}_p = \omega \mathbf{V}_p + \mathrm{U}(0, \varphi_1) \otimes (\mathbf{pBest} - \mathbf{X}_p) + \mathrm{U}(0, \varphi_2) \otimes (\mathbf{gBest} - \mathbf{X}_p) \tag{5}$$

$$\mathbf{X}_p = \mathbf{X}_p + \mathbf{V}_p \tag{6}$$

**Algorithm 3** Factored Evolutionary Algorithm

**Input**: Function f to optimize, optimization algorithm $A$

**Output**: Full solution $G$

```
 1: $S \leftarrow$ initializeFactors($f$, $X$, $A$)
 2: $G \leftarrow$ initializeFullGlobal($S$)
 3: repeat
 4:     for all $P_i \in S$ do
 5:         repeat
 6:             $P_i$.updateIndividuals()
 7:         until Termination criterion is met
 8:     end for
 9:     $G \leftarrow$ Compete ($f$, $S$, $G$)
10:     $S \leftarrow$ Share ($G$, $S$)
11: until Termination criterion is met
12: return $G$
```

The difference operator is defined as a component-wise difference between the two position vectors, i.e., for each variable $X_i$ and value $j \in Vals(X_i)$, $d^j_{(\mathbf{pBest}_p - \mathbf{P}_p),i} = d^j_{pB,i} - d^j_{p,i}$

Here, $d^j_{pB}$ is the personal best position's probability that variable $X_i$ takes value $j$. The global best equation is identical except $\mathbf{pBest}_p$ is replaced with $\mathbf{gBest}$ and $d^j_{pB,i}$ with $d^j_{gB,i}$. Adding the velocity to the position is also component-wise. Thus, $d'^j_{p,i} = d^j_{p,i} + \psi^j_{p,i}$.

After the velocity and position update, the resulting distributions are renormalized to ensure each probability falls within [0, 1], and all of the probabilities sum to 1.

To evaluate a particle $p$, its distributions are sampled to create a candidate solution $\mathbf{S}_p = [s_{p,1}, s_{p,2}, \ldots, s_{p,n}]$ where $s_{p,j}$ denotes the state of variable $X_j$. Then $\mathbf{S}_p$ is tested using the associated fitness function. Thus the samples generated serve as proxies to evaluate the distributions. When a particle produces a sample that beats the global or local best, both the distributions from that particle's position, $\mathbf{P}_p$, and the sample itself, $\mathbf{S}_p$, are used to update the best values.

Mathematically, for all $j \in Vals(X_i)$ the global best's probability is updated as

$$d^j_{gB,i} = \begin{cases} \epsilon \times d^j_{p,i} & \text{if } j \neq s_{p,i} \\ d^j_{p,i} + \sum\limits_{\substack{k \in Vals(X_i) \\ \wedge k \neq j}} (1-\epsilon) \times d^k_{p,i} & \text{if } j = s_{p,i} \end{cases} \tag{7}$$

where $\epsilon$, the *scaling factor*, is a user-set parameter that determines the magnitude of the shift in the distribution restricted to [0, 1). This increases the likelihood of the distribution producing samples similar to the best sample, while inherently maintaining a valid probability distribution. The procedure for setting the local best is the same.

## Related Work

A considerable amount of research has been conducted in the area of multiple-fault diagnosis, and it is impossible to provide a comprehensive treatment here. Therefore, in this section, we focus on reviewing some of the more significant historical literature as well as reviewing work related to abductive inference in Bayesian networks.

### Multiple-Fault Diagnosis

Early work in multiple "disorder" diagnosis was done in a medical setting by James Reggia [19]. His approach involved looking at the general diagnosis problem as corresponding to the *set covering problem* with the intent on using this approach to diagnose multiple simultaneous disorders in a patient. Given the complexity of set covering, Reggia proposed a sequential hypothesis and test mechanism to overlay explanations for the disorders based on the symptoms observed.

Shortly after Reggia's work on multiple disorder diagnosis, Ray Reiter and Johan de Kleer independently and simultaneously published landmark work defining the problems of fault diagnosis and multiple-fault diagnosis more formally [20], [21]. Their work led to the development of a variety of *model-based* methods for diagnosis based on first principles. In their approaches, a basic model of the proper functioning is developed, and diagnosis corresponds to coming up with explanations for deviations from nominal behavior. Reiter's approach utilized a logic-based system, where de Kleer's extended Reggia's ideas from set covering. The difference in de

Kleer's case was that he looked for minimal sets of violated assumptions to generate the diagnosis.

By the late 1980s, several diagnostic systems had been developed using a variety of paradigms. Finin and Morris summarized and compared a lot of this work by putting diagnosis into the context of *abductive reasoning* [22]. Formally, they observed that most reasoning systems assume a logical schema consisting of the following:

1. Major premise (rule): $\forall x\, [P(x) \Rightarrow Q(x)]$
2. Minor premise (case): $P(a)$
3. Conclusion (fact): $Q(a)$

Using this schema, Finin and Morris observed that deductive reasoning corresponds to inferring the third statement from statements 1 and 2, and inductive reasoning corresponds to inferring the first statement from statements 2 and 3. Both methods are generally well-founded and can be related to forward and backward chaining respectively. Abductive inference, on the other hand, is when one has statements 1 and 3 and wants to make a claim about statement 2. In this case, statement 2 would serve as an "explanation" for statement 3 based on the rule structure given in statement 1. In their work, Finin and Morris examined a variety of systems for doing abductive inference, including those based on Bayesian methods, which is of direct relevance to this work.

Several years later, Heckerman and Schwe examined the effectiveness of three different Bayesian methods for performing multiple-fault diagnosis [23]. These methods included a naïve Bayes model (which they refer to as a "simple" Bayes model), a naïve Bayes multi-net model (which they call a multidimensional Bayes model consisting of separate naïve Bayes models for each fault), and a Noisy-Or model [11]. Of note is that these same authors were instrumental in creating the QMR-DT medical diagnostic model, which was one of the first successful diagnostic Bayesian networks utilizing the Noisy-Or formalism [12]. One of their principal findings was that the Noisy-Or model was the closest to ground truth of the three models. Note that we use the Noisy-Or formulation in our work here.

Shortly after the comparison by Heckerman and Schwe, Sheppard and Simpson provided a formal definition of the multiple-fault diagnosis problem and proved that multiple fault diagnosis is NP-complete [1]. Their definition was based on a logic model-based approach, similar to the work of Reiter, and drew on reducing the set covering problem to multiple-fault diagnosis. It is likely the community was aware of the complexity result at the time of this work; however, this paper provided the first complexity proof. It also included a greedy set covering algorithm to be used as an approximation doing multiple-fault diagnosis.

Shakeri *et al*. examined multiple-fault diagnosis in the context of the optimal sequential diagnosis problem [24]. Within this context, they note that the optimization part alone is NP-hard. When embedding multiple-fault diagnosis within the optimization problem, the resulting problem is conjectured to be $NP^{PP}$-hard. Their approach was to focus on applying information theory to determine a sequence of tests covering an AND/OR graph to uncover multiple faults sequentially. They then showed a tradeoff between suboptimality and computational complexity in their approach.

Finally, we consider more recent work by Singh *et al*. who defined and developed algorithms to solve the *dynamic* multiple-fault diagnosis problem [25]. The difference from other multiple-fault diagnosis problems is that DMFD considers a sequence of tests collected over time and mapped to an underlying temporal model, such as a hidden Markov model or a dynamic Bayesian network. Even though their model is temporal, their focus was on explaining past behavior to perform diagnosis rather than attempting to predict future failure. The approach they took was to pose the DMFD problem as a primal-dual problem using Lagrangian relaxation. Because of the probabilistic formulation that results, their approach is also able to incorporate uncertainty of test results into the process.

### Bayesian Abductive Inference

In our prior work [4], we provided a fairly complete review of previous research performed in abductive inference. Among the most significant work done in this area is work by Dechter [26], [27]. In her work, Dechter adapts a form of variable elimination by imposing an elimination order based on marginalizing out nodes in the network with the fewest neighbors, In addition, her bucket elimination (and mini-bucket elimination) algorithm uses max-marginalization rather than sum-marginalization to find the most probable explanation.

A variety of "soft computing" approaches have also been applied, mostly focused on using genetic algorithms. For example, Gelsema used a GA for full abductive inference where the chromosome was a vector of Boolean values, and fitness evaluation used standard likelihood scoring [28]. Similarly, de Campos *et al*. used a GA with integer-based chromosomes to perform partial abductive inference [29]. Fitness was evaluated using probabilistic propagation.

The first application of a PSO-based approach to abductive inference was performed by Ganesan Pillai and Sheppard [30], with that work being extended to a factored version (overlapping swarm intelligence) by Fortier *et al*. [4]. This work demonstrated significant improvement in the accuracy of both full and partial abductive inference over traditional and soft computing methods. Subsequently, we developed FEA as a generalization of OSI and applied it to abductive inference using PSO, GA, and differential evolution as underlying optimization methods [2]. In every case, the FEA method yielded superior results, thus demonstrating that factoring the optimization process is what led to the improved performance.

## Approach

To evaluate our approach, we employ a process whereby we construct Bayesian networks from existing diagnostic strategies known as "fault isolation manuals" (FIM). We then use the resulting networks as the basis for performing abductive inference under uncertainty using our FEA approach. The overall experimental approach is described below under Experimental Design, however, we begin by reviewing the basic process for deriving the Bayesian networks.

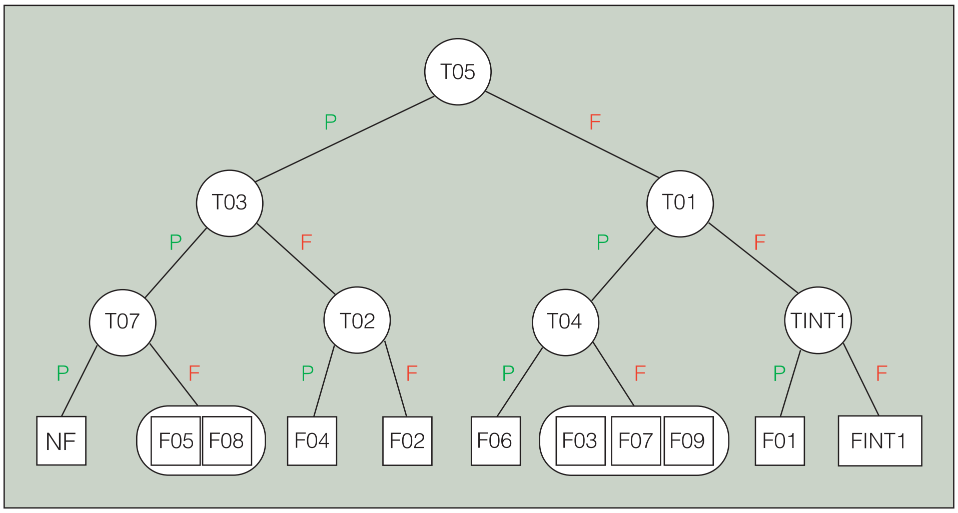


***Fig. 2.*** An example FIM derived from Simpson and Sheppard [31].

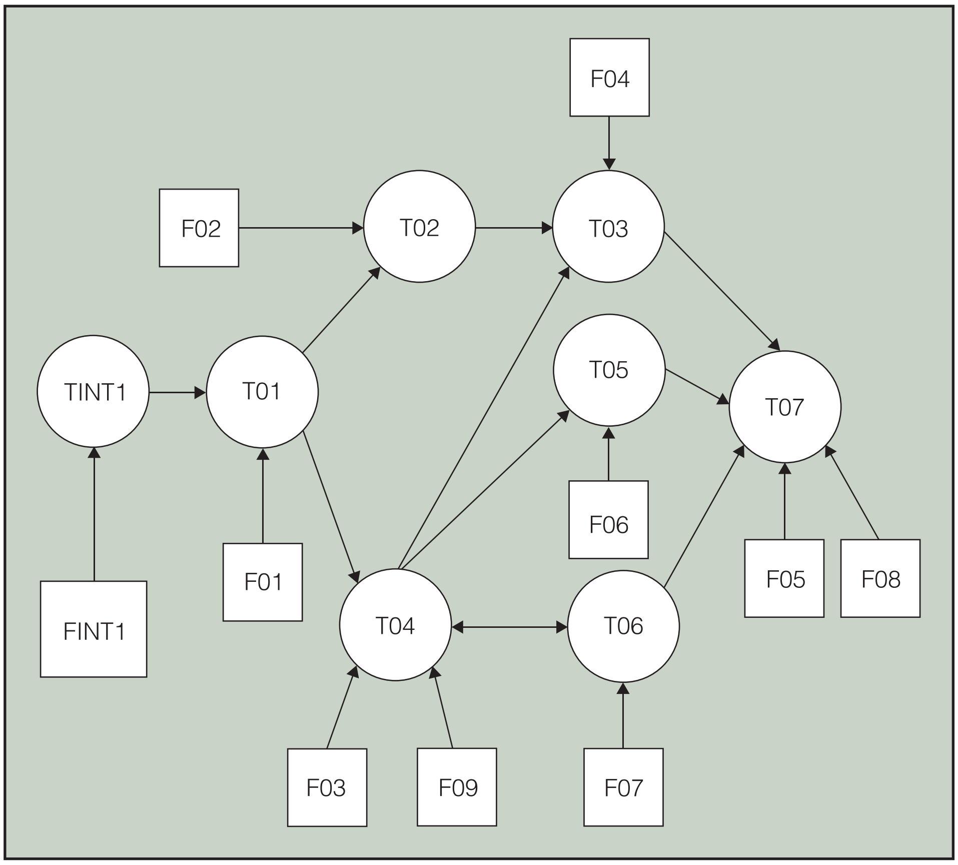


***Fig. 3.*** Simple diagnostic logic model from Simpson and Sheppard [31].

## *Generating Bayesian Networks from FIMs*

The Bayesian networks used in this study were all derived from pre-existing diagnostic strategies provided in the form of FIMs. A FIM is a decision tree or decision graph that specifies pre-defined orders for executing tests where a given test choice is determined by the path leading to that test in the tree. The most common form of FIM is a diagnostic fault tree, such as the one shown in Fig. 2. Note that this particular tree is derived from a simple diagnostic logic model (Fig. 3), as described in Simpson and Sheppard [31].

Previously, we developed a process for deriving D-matrices from FIMs [9]. The D-matrices were then used as the basis for constructing the Bayesian networks. D-matrices, as well as several algorithms for performing both fault diagnosis and testability analysis, are discussed at length in Sheppard and Simpson [31]; therefore, we refer those interested to these two sources for more detail. The key steps in the approach, however, are as follows:

- Construct a bit matrix where rows correspond to faults in the FIM and columns correspond to tests in the FIM.
- From each leaf of the FIM, traverse paths back towards the root. If the edge traversed corresponds to a passing test, insert a 0 in the matrix corresponding to the row of the fault at the leaf and the column of the test visited. If the edge traversed corresponds to a failing test, insert a 1.
- Construct a bipartite directed acyclic graph where one layer of the graph corresponds to fault nodes and the other corresponds to test nodes.
- Wherever an entry exists in the D-matrix, create a directed edge from the fault to the test.
- Assume that all test nodes in the network are Noisy-Or nodes [11].
- Parameterize the network.

The formal process for building the network, taken from [9], is shown in Algorithm 4.

**Algorithm 4** Build Bayesian Network

```
 1: // $FT$ is the fault tree
 2: // $B$ is the Bayesian network
 3: for all Faults $F_i \in FT$ do
 4:   $Fi \leftarrow$ CreateNode
 5:   $Fi \leftarrow$ SetCPT
 6:   $FT$.AddFaultNode($F_i$)
 7: end for
 8: for all Tests $T_i \in FT$ do
 9:   $T_i \leftarrow$ CreateNoisyORNode
10:   $FT$.AddTestNode($T_i$)
11:   for all Faults $F_j \in FT$ do
12:     if $T_i$ indicts $F_j$ by pass link then
13:       $T_i$.AddParent($F_j$)
14:       $T_i$.ProbOfPassGivenFault($F_j$)=0.999
15:       $T_i$.ProbOfFailGivenFault($F_j$)=0.001
16:     end if
17:     if $T_i$ indicts $F_j$ by fail link then
18:       $T_i$.AddParent($F_j$)
19:       $T_i$.ProbOfPassGivenFault($F_j$)=0.001
20:       $T_i$.ProbOfFailGivenFault($F_j$)=0.999
21:     end if
22:   end for
23:   $T_i$.ProbOfLeak =0.001
24: end for
25: return $B$
```

### *Bayesian Network Test Cases*

The experiments evaluating FEA-based diagnosis were broken into two parts. The first focused on running experiments with a known diagnostic model for which we could add multiple fault cases in a straightforward manner. For these test cases, we used the model of a hypothetical missile launcher circuit, published in Simpson and Sheppard [31]. The second set of experiments was based on multiple fault scenarios derived from real-world FIMs for a U.S. military aircraft. For these experiments, DFIMs (i.e., Bayesian networks) were generated using the procedures described in the previous section.

*Missile Launcher Circuit D-Matrix*: For the first set of experiments, we used the hypothetical missile launcher circuit published by Simpson and Sheppard [31]. For this model, we generated two different Bayesian networks. The first was generated directly from the model's D-Matrix. For the second version, we created a network from a FIM built using a decision tree.

The following describes the process for generating the multiple fault test cases from the D-matrix.

- Generate Bayesian network: The process began by generating a diagnostic Bayesian network from the D-matrix.
- Collapse redundant tests: The D-matrix enables tests that provide identical diagnostic information to be identified by looking for identical matrix columns. Where these occurred, we selected one of the columns as a representative test and eliminated the others (and associated test nodes) from the network.
- Combine ambiguity groups: The D-matrix also enables ambiguous faults to be identified by looking for identical matrix rows. As with redundant tests, we reduced the Bayesian network to include a representative fault from the model and eliminated the other members of the ambiguity group.
- Generate test sequences: These test sequences are used as the evidence collected for performing diagnosis. The sequences are generated as follows:
  - For each unique pair of faults $f_i$ and $f_j$:
    - o Assign $f_i$ and $f_j$ as "Faulty."
    - o Assign all other faults as "OK."
    - o Query the probability of all tests after inference.
    - o If P($t_k$ = Fail) > 0.50, then add $t_k$ = Fail to the evidence list.
    - o If P($t_k$ = Pass) > 0.50, then add $t_k$ = Pass to the evidence list.
  - For each test sequence:
    - o Assign test results as evidence based on the evidence list.
    - o Use FEA to diagnose faults.
    - o Use the ground truth to calculate correctness of solution returned from FEA.

Test cases were also generated from the fault tree derived from the D-matrix. The procedure for generating these test cases was identical to the procedure for the military aircraft test cases and is described in the next section.

*Military Aircraft DFIMs*: For the second set of experiments, we used ten example FIMs for a US military aircraft and derived DFIMs (i.e., Bayesian networks) using the procedure described in the *Generating Bayesian Networks from FIMs* section above. This procedure was also used to derive the DFIM from the FIM for the missile launcher circuit. For these experiments, we did not have the corresponding D-matrices, so we assumed multiple faults corresponded to different paths through the FIM. The following describes the process for generating these test cases.

- Select several different DFIMs of interest: The selection process was based on examining the structure of the available FIMs. For example, we selected one FIM that had mostly long paths, and we selected another FIM that had mostly short paths.
- Generate test sequences: As with the missile launcher circuit model, the test sequences were used as evidence collected for performing diagnosis. The sequences were generated as follows:
  - For each unique pair of faults $f_i$ and $f_j$:
    - Assign $f_i$ and $f_j$ as "Faulty."
    - Assign all other faults as "OK."
    - Query the probability of all tests.
    - If $P(t_k = \text{Fail}) > 0.50$, then add $t_k = \text{Fail}$ to the evidence list.
    - If $P(t_k = \text{Pass}) > 0.50$, then add $t_k = \text{Pass}$ to the evidence list.
  - For each test sequence:
    - Assign test results as evidence from the evidence list.
    - Use FEA to diagnose faults.
    - Use the ground truth to calculate correctness of solution returned from FEA.

Notice that the process is virtually identical to the process used when we have a D-matrix. The key difference comes in how the Bayesian networks are generated. Since DFIMs under-specify the information available for doing diagnosis, we expected the performance of DFIM-based network inference to be degraded when compared to D-matrix-based network inference. As we will see, this expectation turned out to be incorrect.

### Experimental Design

The experimental process for each problem went as follows. First, we used either the D-matrix or an existing FIM as the basis for generating the Bayesian network. We refer to the Bayesian network as a DFIM since it permits us to apply evidence from any of the tests performed in any order.

After constructing the DFIMs, we then generated all unique pairwise faults covered by that model. Thus, for a FIM with $n$ unique faults, we generated $n(n-1)/2$ fault pairs. We then extracted the test results from the original D-matrix or FIM for the individual faults and constructed an evidence set as follows. For fault pair $\langle f_i, f_j \rangle$, construct evidence sets

$$\varepsilon_i = \left[ t_i^1, \ldots, t_i^m \right] \tag{8}$$

$$\varepsilon_j = \left[ t_j^1, \ldots, t_j^m \right] \tag{9}$$

where

$$t^k = \begin{cases} 0 & \text{test } k \text{ passes} \\ 1 & \text{test } k \text{ fails} \\ \# & \text{test } k \text{ not evaluated} \end{cases} \tag{10}$$

and # denotes a don't care. Then we construct a combined evidence vector as

$$\varepsilon_{i,j} = \varepsilon_i \oplus \varepsilon_j \tag{11}$$

which corresponds to the element-wise OR of the evidence vectors. When OR-ing with a don't care, the non-don't care value is assigned.

Once the evidence vectors are specified, FEA partial abductive inference is run. Specifically, a subswarm is specified for each fault node in the DFIM with 10 particles per swarm. Unassigned test nodes are marginalized out as part of the inference process. The underlying optimization algorithm was ICPSO as described previously.

The fitness function used by FEA is defined as

$$f(\mathrm{x}) = \sum_{i=1}^{n} \Delta_i x_i \tag{12}$$

where $\Delta_i$ is the change in probability of fault $i$. $\Delta_i$ is calculated as the change in probability before and after evidence has been applied to the network. **x** is a binary array where a 0 bit indicates no fault ("OK") and 1 is "Fault." Whether or not a fault exists in the fitness function is based on a threshold applied to the inferred posterior probability for that fault node. We terminated FEA after the best solution failed to change after 10 iterations, and the corresponding vector **x** was compared to the ground truth taken from the FIMs or D-matrix.

## Results

In this section, we review the results from our experiments. We note that we did not compare to alternative multiple-fault diagnosis strategies for a few reasons. First, our prior work already demonstrated the superiority of the FEA-based approach to abductive inference as compared to evolutionary, swarm-based, and traditional inference methods. Since most of the multiple-fault diagnosis methods fall in the "traditional" category (at least with respect to using Bayesian networks), we felt that the previous comparisons were sufficient.

Second, as we found in these results, the effectiveness of multiple-fault algorithms depends heavily on the type of underlying model. Since our intent was not to evaluate models but to evaluate the effectiveness of the FEA method, we felt that using different model types would tend to confuse the issue. Therefore, we limited our model types to diagnostic Bayesian networks.

Finally, the work reported in this paper is not intended to be the final say on what constitutes the best way to perform multiple-fault diagnosis. Indeed, we recognize that it is likely that a "No Free Lunch" theorem applies here. Indeed, the focus of this research was to be more suggestive than conclusive where we sought to test the feasibility of the FEA approach.

To that end, we report the main results of our experiments in Table 1. This table shows the results on two separate sets of experiments, the first comparing two different Bayesian networks derived from the hypothetical missile launcher circuit model from Simpson and Sheppard [31] (which we denote

Table 1 – Percent of time FEA diagnosed the correct faults

| Model | % Correct | Fault pairs | Tests assigned |
|---|---|---|---|
| MLC (D-Matrix) | 42.7% | 171 | 11.16 (3.57) |
| MLC (FIM) | 100% | 171 | 6.91 (1.21) |
| DFIM-1 | 100% | 21 | 6.67 (1.53) |
| DFIM-2 | 100% | 15 | 4.33 (0.98) |
| DFIM-3 | 100% | 55 | 7.60 (2.10) |
| DFIM-4 | 100% | 190 | 10.66 (2.98) |
| DFIM-5 | 100% | 55 | 5.64 (1.30) |
| DFIM-6 | 100% | 91 | 8.34 (2.11) |
| DFIM-7 | 100% | 66 | 7.18 (1.52) |
| DFIM-8 | 100% | 83 | 7.83 (2.02) |
| DFIM-9 | 100% | 96 | 8.46 (3.01) |
| DFIM-10 | 100% | 75 | 7.42 (1.04) |

"MLC"), and the second focusing on the results applying FEA to ten actual FIMs for a military aircraft (which we denote "DFIM"). Note that this is a more extensive set of test cases than we reported in [8].

In examining the results in Table 1, the first thing to note is that these are aggregates over all of the scenarios run for each model. As reported above, this means that each entry in the table corresponds to all pairwise faults from the original D-matrix or FIM. This allowed us to consider, not only the easiest fault pairs, but the most difficult too (and everything in between).

More specifically, the results in this table show the percent of the time that FEA diagnoses the correct two faults when compared with the expected results. Thus, in every case, we used the ground truth as the basis for comparison. It is particularly interesting to note the poor performance of FEA on the DFIM derived from the MLC D-matrix as compared to the perfect performance of the DFIM derived from the MLC FIM. This result is completely counter to our expectations since D-matrices provide substantially more diagnostic information than FIMs alone [9].

Table 1 also shows the number of fault pairs diagnosed, which corresponds to the number of scenarios run for the corresponding Bayesian network. It then shows the average number of tests assigned as evidence for each of the networks with the associated standard deviation. One thing to note on the MLC models is that the number of tests assigned as evidence for the D-matrix network is almost twice the number assigned to the FIM network. Once again, this is counter intuitive as one would expect more test information to provide better resolution, especially when multiple faults are involved.

One thing that is quite striking from these results is that FEA provided perfect results on every FIM-based Bayesian network. That said, it is important to note that all of the tests were assigned accurate results based on the ground truth, so these results do not incorporate anything that tests the impact of noisy testing. Furthermore, since the test results were based on the ground-truth FIMs, we knew *a priori* that the diagnoses would be correct, at least for the corresponding single faults. Thus what we are considering is whether or not combining faults would somehow confuse the diagnostic process. This issue will be discussed more in the next section.

## Discussion

As noted in the previous section, there were two main results that were surprising and warrant additional discussion. The first is the fact that the D-matrix network performance was substantially worse (~43% accuracy) than the FIM network performance (100% accuracy on all models). On the surface, this seems to make no sense, until we take a closer look at the MLC D-matrix.

The work in [1] explains that there are two types of multiple fault scenarios captured by a D-matrix that would thwart diagnostic strategies based on these models. The first is a "false failure," which occurs when the combined fault signature of a multiple fault is equivalent to the fault signature of a separate single fault or another multiple fault. Thus, in a sense, a false failure corresponds to a multiple fault ambiguity group. The MLC model includes two false failures, one of which involves a pair of single faults.

But that is not enough to explain the 53% degradation in performance. The other type of multiple fault situation that creates problems for a D-matrix is a "hidden failure." A hidden failure corresponds to the situation where the signature of one fault subsumes (i.e., is a superset of) another fault signature. When this situation arises, all of the failed tests indicting the subsumed fault also indict the subsuming fault, so there is no way to differentiate them. Consequently, there is no way to pull the pair out as a unique fault signature. In the MLC D-matrix, a very high number of hidden failure pairs exist, so it is our hypothesis that this is what led to the poor performance of FEA.

When we consider the FIMs, on the other hand, by construction all of the evidence vectors (which are used to construct the Bayesian networks in the first place) already include clear differentiators between the single faults. This leads to a situation where hidden and false failures almost never occur. As a result, even though fewer test results are being applied to the associated networks, in every case they happen to be the right test results to differentiate the multiple fault pairs from the rest of the candidates.

To test this hypothesis, we performed an analysis of the multiple fault signatures. First, we generated the fault signature for every multiple fault pair. We then performed the following steps: 1) Let fault A and B represent the two faults in the current multiple fault signature to be analyzed. 2) For all faults, determine if the multiple fault signature for A and B is a subset of all other single fault signatures. 3) If the multiple fault signature is a subset of a fault signature (not including the signatures for faults A and B), increment a subset counter. We performed this analysis on both the MLC (D-Matrix) and MLC (FIM) models. Results are summarized in Table 2.

**Table 2 – Results of analyzing the number of times multiple fault signatures were subsets of single fault signature**

| Model | Total number of subsets | Subsets per fault pair |
|---|---|---|
| MLC (D-Matrix) | 808 | 4.23 |
| MLC (FIM) | 0 | 0 |

From the results in Table 2, we can see that for the D-Matrix MLC model there were 808 cases where the fault signature for a multiple fault was a subset of a fault signature for a single fault, and on average, a multiple fault signature had 4.23 hidden failures. This is a clear indication of hidden failures. However, on the FIM version of the MLC model, there were 0 instances where the multiple fault signature had hidden failures for a single fault.

This leads us to the question of why do the fault signatures for the multiple fault scenarios not have any hidden failures. To answer this, we analyzed how the fault signature for multiple faults differs from the fault signatures for the individual faults. We took the fault signature for each individual fault and compared them with one another. For each pair, we calculated the size of the fault signature intersect and symmetric set difference. The intersect corresponds to the fault signature of the multiple fault while the set difference is a measure on how much the fault signature of the two faults differ. The results are reported in Table 3.

**Table 3 – Results of analyzing the average set intersect and symmetric difference of all pairs of fault signatures**

| Model | Intersect | Difference |
|---|---|---|
| MLC (D-Matrix) | 10.16 | 5.83 |
| MLC (FIM) | 0.82 | 1.0 |
| DFIM-1 | 1.19 | 1.0 |
| DFIM-2 | 1.33 | 1.0 |
| DFIM-3 | 2.13 | 1.0 |
| DFIM-4 | 3.84 | 1.0 |
| DFIM-5 | 0.81 | 1.0 |
| DFIM-6 | 3.09 | 1.0 |
| DFIM-7 | 1.15 | 1.0 |
| DFIM-8 | 1.45 | 1.0 |
| DFIM-9 | 2.29 | 1.0 |
| DFIM-10 | 1.75 | 1.0 |

The biggest thing to take away from the results in Table 3 was the difference in fault signature of the DFIMs vs the D-Matrix. On the MLC (D-Matrix) model, on average the fault signatures differed by 5.83 tests whereas on the FIM and DFIMs models, the signatures differed by only one single test. This is because in the FIM and DFIM models, the fault signature is based upon a unique path in the tree that is a subset (often proper) of the total number of tests. However, in the D-Matrix model, the fault signature is defined over all of the tests, which means when combining fault signature, there is a greater chance of the fault signatures differing on a test.

The second result that was surprising to us was the perfect performance of FEA on all ten FIM-based models (MLC and DFIM-1 through DFIM-7). In fact, this result led us to question whether or not we somehow gave these multiple fault scenarios an unfair advantage that might not exist in the real world. While this is still a distinct possibility, we do not believe it explains the behavior observed.

First, we grant that the problem solved here assumes *a priori* knowledge that we are looking specifically for fault pairs. We contend this is not that significant of an issue since most diagnostic systems assume single fault or multiple faults of fairly low order (e.g., pairs or triples). It would have been straightforward to include single fault and triple fault analysis as separate runtime scenarios, and we do not believe the results would have changed substantially. We acknowledge that a full treatment of all possible multiple fault scenarios is not feasible since this amounts to the MRE problem described in the previous background discussion.

As to why the performance was perfect, we believe this is related to what we described above. Each multiple fault scenario involved combining signatures for the corresponding single faults. This meant extracting the test outcomes from the underlying FIMs for each fault's path in the tree and using that as evidence. If two paths had contradictory test results, we left those results out of the evidence set. Consequently, each of the fault pairs would have had unique evidence sets, so it was just a matter of examining the joint posterior distributions on the fault pairs for FEA to discover the best diagnosis.

## Conclusions

Based on our prior work developing distributed and population-based methods for abductive inference, we had hypothesized that these results could be extended to the multiple-fault diagnosis scenario. To that end, this paper describes preliminary experiments applying factored evolutionary algorithms (FEA) to the multiple-fault diagnosis problem, posed as an abductive inference problem. In summary, the results were astounding and suggest the FEA approach could provide a promising new method for multiple-fault diagnosis, at least when the right underlying model is being used.

Key to this conclusion is the observation that the model type matters, and that more information is not necessarily better than less information. In fact, while prior work suggests that Bayesian networks derived from fault trees could face challenges because of a loss of information and an increase in uncertainty as to how test results reflect fault state, these results suggest that the corresponding reduction in information might actually give the derived Bayesian networks an advantage.

Based on these observations, it appears further study is warranted to determine what constitutes the optimal amount of information to include in a diagnostic model to support single and multiple-fault diagnosis. As this question was not the focus of the study reported here, we leave that question to future work.

Other questions arising from this work include the following:

- To what extent will noise in the test results degrade FEA-based multiple-fault diagnosis?
- Are there reasons other than the false and hidden failure scenarios that might explain why D-matrix networks perform poorly in the multiple fault setting? For example, could the behavior be tied to an inherent single-fault assumption in the D-matrix models?
- The method for deriving the Bayesian networks discussed in this work makes heavy use of the Noisy-Or assumption. To what extent does this assumption affect performance, especially given the extreme simplification of the semantics defined for the relationships included in the networks?
- Given that a variety of fault trees can be generated for a unit under test, how might an ensemble of FIM-based networks improve (or degrade) multiple-fault diagnosis?
- It should be relatively straightforward to create a distributed version of FEA; therefore, would using a distributed architecture affect overall diagnostic performance? In particular, could the distributed architecture be exploited to improve performance?
- Finally, what would the impact be if temporal information was incorporated into the diagnostic process (e.g., by mapping the problems to dynamic Bayesian networks and capturing the timing of the test results)?

These questions represent only a few of the issues of interest to us. It is our hope that further study will provide not only better insight into the behavior of FEA on multiple-fault Bayesian diagnosis but might also yield even better methods for more general multiple-fault diagnosis problems.

## References

[1] J. W. Sheppard and W. R. Simpson, "Multiple failure diagnosis," in *Proc. IEEE AUTOTESTCON*, pp. 381-389, Sep. 1994.

[2] S. Strasser, J. Sheppard, N. Fortier, and R. Goodman, "Factored evolutionary algorithms," *IEEE Trans. Evolutionary Computation*, vol. 21, no. 2, pp. 281-293, 2017.

[3] B. K. Haberman and J. W. Sheppard, "Overlapping particle swarms for energy-efficient routing in sensor networks," *Wireless Networks*, vol. 18, no. 4, pp. 351-363, 2012.

[4] N. Fortier, J. Sheppard, and S. Strasser, "Abductive inference in Bayesian networks using distributed overlapping swarm intelligence," *Soft Computing*, vol. 19, no. 4, pp. 981-1001, 2015.

[5] L. M. de Campos, J. A. Gámez, and S. Moral, "Partial abductive inference in Bayesian belief networks using a genetic algorithm," *Pattern Recognition Letters*, vol. 20, pp. 1211-1217, 1999.

[6] P. Dagum and M. Luby, "Approximating probabilistic inference in Bayesian belief networks is NP-hard," *Artificial Intelligence*, vol. 60, no. 1, pp. 141-153, 1993.

[7] S. E. Shimony, "Finding MAPs for belief networks is NP-hard," *Artificial Intelligence*, vol. 68, no. 2, pp. 399-410, 1994.

[8] J. Sheppard and S. Strasser, "A factored evolutionary optimization approach to Bayesian abductive inference for multiplefault diagnosis," in *Proc. IEEE AUTOTESTCON*, pp. 53-62, 2017.

[9] S. Strasser and J. Sheppard, "An empirical evaluation of Bayesian networks derived from fault trees," in *Proc. IEEE Aerospace Conf.*, pp. 1-13, 2013.

[10] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York, NY, USA: W. H. Freeman, 1979.

[11] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Francisco, CA, USA: Morgan Kaufmann Publishers, Inc., 1988.

[12] M. Shwe, B. Middleton, D. Heckerman, M. Henrion, E. Horvitz, H. Lehmann, and G. Cooper, "Probabilistic diagnosis using a reformulation of the INTERNIST-1/QMR knowledge base. i. the probabilistic model and interference algorithms," *Methods of Information in Medicine*, vol. 30, no. 4, pp. 241-255, 1991.

[13] C. Yuan, H. Lim, and M. L. Littman, "Most relevant explanation: computational complexity and approximation methods," *Annals of Mathematics in Artificial Intelligence*, vol. 61, pp. 159-183, 2011.

[14] J. W. Sheppard and M. A. Kaufman, "A bayesian approach to diagnosis and prognosis using built-in test," *IEEE Trans. Instrum. Meas.*, vol. 54, no. 3, pp. 1003-1018, June 2005.

[15] M. A. Potter and K. A. DeJong, "A cooperative coevolutionary approach to function optimization," in *Proc. Int. Conf. on Evolutionary Computation, The Third Conf. on Parallel Problem Solving from Nature*, ser. PPSN III. London, UK: Springer-Verlag, pp. 249-257, 1994.

[16] F. Van den Bergh and A. P. Engelbrecht, "A cooperative approach to particle swarm optimization," *IEEE Trans. Evolutionary Computation*, vol. 8, no. 3, pp. 225-239, 2004.

[17] Y.-J. Shi, H.-F. Teng, and Z.-Q. Li, "Cooperative co-evolutionary differential evolution for function optimization," *Advances in Natural Computation*, pp. 428-428, 2005.

[18] S. Strasser, R. Goodman, J. Sheppard, and S. Butcher, "A new discrete particle swarm optimization algorithm," in *Proc. Genetic and Evolutionary Computation Conf. (GECCO)*, ACM, pp. 53-60, 2016.

[19] J. A. Reggia, D. S. Nau, and P. Y. Wang, *Diagnostic Expert Systems Based on a Set Covering Model*. New York, NY, USA: Springer New York, pp. 159-185, 1985.

[20] R. Reiter, "A theory of diagnosis from first principles," *Artificial Intelligence*, vol. 32, no. 1, pp. 57-95, Apr. 1987.

[21] J. de Kleer and B. C. Williams, "Diagnosing multiple faults," *Artificial Intelligence*, vol. 32, no. 1, pp. 97-130, Apr. 1987.

[22] T. Finin and G. Morris, "Abductive reasoning in multiple fault diagnosis," *Artificial Intelligence Review*, vol. 3, no. 2, pp. 129-158, Jun. 1989.

[23] D. Heckerman and M. Shwe, "Diagnosis of multiple faults: a sensitivity analysis," in *Proc. of the Ninth Int. Conf. on Uncertainty in Artificial Intelligence (UAI)*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., pp. 80-87.1993.

[24] M. Shakeri, V. Raghavan, K. R. Pattipati, and A. Patterson-Hine, "Sequential testing algorithms for multiple fault diagnosis," *IEEE Trans. Systems, Man, and Cybernetics – Part A: Systems and Humans*, vol. 30, no. 1, pp. 1-14, Jan. 2000.

[25] S. Singh, A. Kodali, K. Choi, K. R. Pattipati, S. M. Namburu, S. C. Sean, D. V. Prokhorov, and L. Qiao, "Dynamic multiple fault diagnosis: mathematical formulations and solution techniques," *IEEE Trans, Systems, Man, and Cybernetics – Part A: Systems and Humans*, vol. 39, no. 1, pp. 160-176, Jan. 2009.

[26] R. Dechter, "Bucket elimination: A unifying framework for probabilistic inference," in *Proc. of the 12th Int. Conf. on Uncertainty in Artificial Intelligence (UAI)*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., pp. 211-219, 1996.

[27] R. Dechter and R. Irina, "Mini-buckets: A general scheme for bounded inference," *J. of the ACM*, vol. 50, no. 2, pp. 107-153, 2003.

[28] E. Gelsema, "Abductive reasoning in Bayesian belief networks using a genetic algorithm," *Pattern Recognition Letters*, vol. 16, pp. 865-871, 1995.

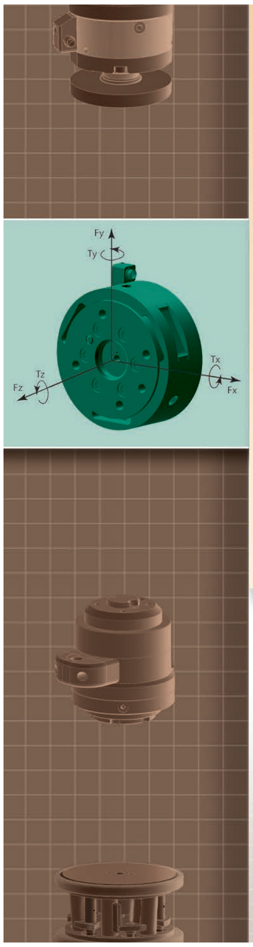
[29] L. de Campos, J. Gamez, and S. Moral, "Partial abductive inference in Bayesian belief networks using a genetic algorithm," *Pattern Recognition Letters*, vol. 20, pp. 1211-1217, 1999.

[30] K. G. Pillai and J. W. Sheppard, "Abductive inference in Bayesian belief networks using swarm intelligence," in *Proc. 6th Int. Conf. on Soft Computing and Intelligent Systems (SCIS) and 13th Int. Symp. on Advanced Intelligent Systems (ISIS)*, pp. 375-380, 2012.

[31] W. R. Simpson and J. W. Sheppard, *System Test and Diagnosis*. New York, NY, USA: Springer, 1994.

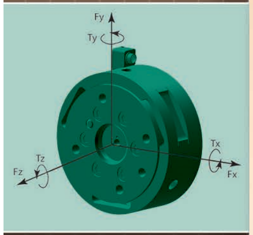
***John W. Sheppard*** (M'86–SM'97–F'07) is the Norm Asbjornson College of Engineering Distinguished Professor of Computer Science in the Gianforte School of Computing, Montana State University, Bozeman, MT, USA. Previously, he spent 20 years in industry at ARINC Inc., Annapolis, MD, USA, where he attained the rank of Fellow. He received the B.S. in computer science from Southern Methodist University, Dallas, TX, USA, and the M.S. and Ph.D. degrees in computer science from The Johns Hopkins University, Baltimore, MD, USA. Dr. Sheppard is a Fellow of the IEEE, and his current research interests include probabilistic graphical models, machine learning, evolutionary algorithms, and algorithms for prognostics and health management.

***Shane Strasser*** (M'17) is a Software Engineer at Oracle in Bozeman, MT. He holds a B.S. degree in math and computer science from the University of Sioux Falls and M.S. and Ph.D. degrees in computer science from Montana State University. His dissertation focused on formalizing the Factored Evolutionary Algorithm framework. Dr. Strasser has experience in fault diagnosis and fault prognosis and has worked on funded research with both the US Navy and NASA.



ROBOTIC END-EFFECTORS

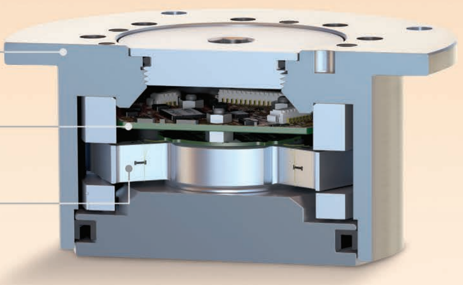
**Measure all six components of force and torque in a compact, rugged sensor.**



**Interface Structure**—high-strength alloy provides IP60, IP65, and IP68 environmental protection as needed

**Low-noise Electronics**—interfaces for Ethernet, PCI, USB, EtherNet/IP, PROFINET, CAN, EtherCAT, Wireless, and more

**Sensing Beams and Flexures**—designed for high stiffness and overload protection



The F/T Sensor outperforms traditional load cells, instantly providing all loading data in every axis. Engineered for high overload protection and low noise, it's the ultimate force/torque sensor. Only from ATI.

ATI INDUSTRIAL AUTOMATION

Engineered Products for Robotic Productivity

**www.ati-ia.com/ims**
**919.772.0115**