

Comparison-Based System-Level Fault Diagnosis: A Neural Network Approach

Mourad Elhadeif and Amiya Nayak

Abstract—We consider the fault identification problem, also known as the system-level self-diagnosis, in multiprocessor and multicomputer systems using the comparison approach. In this diagnosis model, a set of tasks is assigned to pairs of nodes and their outcomes are compared by neighboring nodes. Given that comparisons are performed by the nodes themselves, faulty nodes can incorrectly claim that fault-free nodes are faulty or that faulty ones are fault-free. The collections of all agreements and disagreements, i.e., the comparison outcomes, among the nodes are used to identify the set of permanently faulty nodes. Since the introduction of the comparison model, significant progress has been made in both theory and practice associated with the original model and its offshoots. Nevertheless, the problem of efficiently identifying the set of faulty nodes when not all the comparison outcomes are available to the diagnosis algorithm at the beginning of the diagnosis phase, i.e., partial syndromes, remains an outstanding research issue. In this paper, we introduce a novel diagnosis approach using neural networks to solve this fault identification problem using partial syndromes. Results from a thorough simulation study demonstrate the effectiveness of the neural-network-based self-diagnosis algorithm for randomly generated diagnosable systems of different sizes and under various fault scenarios. We have then conducted extensive simulations using partial syndromes and nondiagnosable systems. Simulations showed that the neural-network-based diagnosis approach provided good results making it a viable addition or alternative to existing diagnosis algorithms.

Index Terms—Fault tolerance, system-level self-diagnosis, comparison models, partial syndromes, neural networks.

1 INTRODUCTION

IN recent decades, the need for dependable computing systems for critical applications has motivated researchers to investigate self-diagnosable large-scale distributed systems and loosely coupled multiprocessor and multicomputer systems. These so-called loosely coupled multiprocessor systems are sometimes composed of hundreds (or even thousands) of interconnected processing nodes. In order to detect faults each node is assumed to test and to be tested by other nodes of the system. From the results of the tests, nodes need to be diagnosed as faulty or fault-free. This problem is known as the *system-level fault diagnosis problem*. The system-level diagnosis problem has been extensively studied in the last three decades (the reader is referred to the following surveys [1], [2], and the most recent comprehensive one by Duarte Jr. et al. [3] for more details).

Three types of diagnosis models have been studied in the context of system-level self-diagnosis: testing, comparison, and probabilistic models. *Testing models*, such as the classical PMC model [4], and its variations such as the BGM model [5] and the HK models [6], assume that each node is assigned a subset of the other units to test and the diagnosis is based on the collection of test outcomes. While, *comparison models*, such as the Malek's comparison model

[7] and the Chwa-Hakimi comparison model [8], assume that a set of jobs is assigned to pairs of distinct units, and the results are compared. The outcomes of these comparisons, i.e., the matching and mismatching results, are used as a basis in order to identify the set of faulty nodes. Fault diagnosis under the testing and comparison models assumes in general a worst case behavior. That is, only t -diagnosable systems where the maximum number of faults is bounded by t are considered in order to guarantee a certain level of diagnosis. Finally, *probabilistic models* [2] do not assume any bound, but instead, only fault sets that have a nonnegligible probability of occurrence are considered.

In this paper, we consider the comparison-based diagnosis approach. The comparison approach has been introduced independently by Malek [7] and by Chwa and Hakimi [8] giving rise to two models. The Malek's model is known as the *asymmetric comparison* model and that of Chwa and Hakimi is called the *symmetric comparison* model. In both models it is assumed that two fault-free nodes give matching (0) results while a faulty and a fault-free node give mismatching (1) outcomes. The two models differ in the assumption on tests involving a pair of faulty nodes. In the symmetric model, both test outcomes (0/1) are possible in this case, while in the asymmetric model two faulty nodes always give mismatching outputs (1).

In [9], Maeng and Malek extended the comparison model by allowing the comparisons to be conducted by the nodes themselves. Their extended model is known as the *Maeng/Malek* (MM) model. Furthermore, in [9], it was assumed that a comparison is performed by each node for each pair of distinct neighbors with which it can communicate directly; this special case of the MM model is referred to as the MM* model. In [10], Sengupta and Dahbura presented a generalization of the testing and comparison models by introducing a new model, known as the *generalized comparison* model.

• M. Elhadeif is with the College of Engineering and Computer Science, Abu Dhabi University, PO Box 59911, Abu Dhabi, UAE.
E-mail: elhadeif@site.uottawa.ca.

• A. Nayak is with the School of Electrical Engineering and Computer Science, University of Ottawa, 800 King Edward Avenue, Ottawa, Ontario K1N 6N5, Canada. E-mail: anayak@site.uottawa.ca.

Manuscript received 5 Jan. 2011; revised 15 June 2011; accepted 16 Sept. 2011; published online 30 Sept. 2011.

Recommended for acceptance by P. Santi.

For information on obtaining reprints of this article, please send e-mail to: tpsds@computer.org, and reference IEEECS Log Number TPDS-2011-01-0013. Digital Object Identifier no. 10.1109/TPDS.2011.248.

Authorized licensed use limited to: University of New South Wales. Downloaded on March 24, 2024 at 10:50:14 UTC from IEEE Xplore. Restrictions apply.
1045-9219/12/\$31.00 © 2012 IEEE

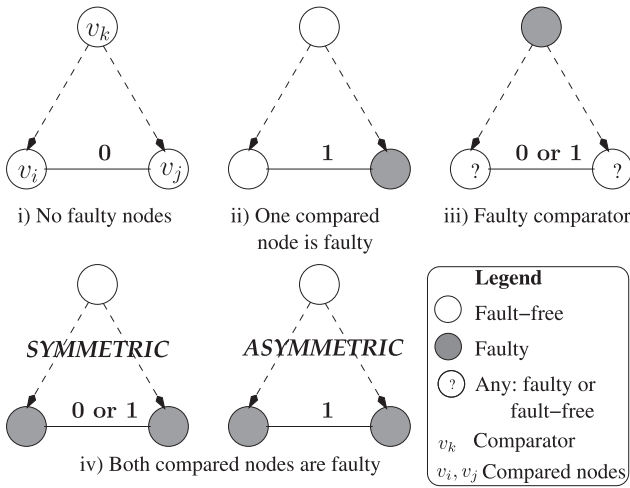


Fig. 1. GCM's invalidation rules. The value besides the undirected edge denotes the comparison outcome.

(GCM), in which the comparator node can be one of the two nodes under comparison. Fig. 1 depicts the GCM's invalidation rules. Blough and Brown introduced next in [11] a combination of a distributed diagnosis and the generalized comparison model in systems having weak reliable broadcast capacity. They developed the first *broadcast comparison model* (BCM), in which two nodes under comparison broadcast their outputs to all nodes in the system. In [12], Chessa and Santi applied the comparison-based system-level fault diagnosis approach to ad hoc networks. More recently, Albini et al. [13] introduced the generalized distributed comparison-based (GDC) model which is based on the same assumptions of the MM model, plus, it requires that fault-free nodes execute tasks within a bounded time.

Identifying the complete and correct set of faulty nodes using a comparison model has been shown to be NP-hard [14], but if the system is t -diagnosable, the problem is solvable in polynomial time. This problem has been extensively studied leading to elegant and efficient solutions [8], [9], [10], [11], [12], [13], [14], [15], [22], [23], [24], [25], [27], [28], [29], [30], [31], [32], [34], [37]. In this paper, we present a totally different approach based on artificial neural networks (ANNs) for solving the system-level diagnosis problem. We consider different comparison models including the Chwa and Hakimi's symmetric comparison model [8] and the generalized comparison model [10]. Note that the GCM is a general class of the testing models, and hence, the new ANNs-based diagnosis approach could be easily applied to these models as well with minor adaptations. ANNs have been successfully applied to a broad spectrum of applications including medical [16], industrial [17], financial [18], data mining [19], etc. In particular, ANNs have been also used to solve various fault diagnosis problems [20], [21], but to the best of our knowledge, this is the first time where ANNs are used to solve the system-level fault diagnosis.

The neural network diagnosis approach is one of the few diagnosis algorithms [10], [22], [23], [24] that have been devised for the GCM-based fault identification problem. We believe that this new type of diagnosis approach could be

important in the design of future generations of dependable systems. One of the main contributions of the new diagnosis approach compared to existing deterministic polynomial time diagnosis algorithms is the fact that it can provide a certain level of correct diagnosis even when not all the comparison outcomes are available to the diagnosis algorithm at the beginning of the diagnosis phase, i.e., partial syndromes. In addition, the neural-network-based diagnosis exploits the offline learning phase of neural networks to speed up the diagnosis algorithm providing, hence, online diagnosis. It must be noted that a shorter paper describing an initial version of the neural-network-based diagnosis to the symmetric comparison model was published in [25]. The main difference between the two papers is the addition in this paper of the generalization of the neural network approach to GCM and the new interpolation phase, and inclusion of a more comprehensive set of simulation results. The interpolation process aims at speeding up the diagnosis algorithm by allowing the computation of results from precomputed values.

The remainder of this paper is organized as follow: we first provide a general view of the fault and comparison models, followed by basic definitions and notations in Section 2. Section 3 discusses related work. The neural-network-based diagnosis approach is detailed in Section 4, followed by the main steps for improving the diagnosis correctness, i.e., a postprocessing phase and an interpolation phase. A concrete example is also provided in Section 4 to help understand the interpolation process. Simulation results are provided in Section 5. Section 6 concludes the discussion and motivates future investigations on the system-level fault diagnosis problem.

2 PRELIMINARY DEFINITIONS

The system we consider is composed of stable nodes that are interconnected with each other via a wired or wireless communication network. In comparison models, the system is modeled by an undirected graph, and it is assumed that pairs of nodes are assigned the same task to be performed. The agreements (0) and disagreements (1) among the nodes are the basis for identifying the set of faulty nodes. It is assumed that two fault-free nodes always give matching results, while a faulty and a fault-free unit always give mismatching outcomes. The comparison model for system-level fault self-diagnosis can be described by two graphs: a *communication graph* and a *comparison* (or *test*) *graph*. The communication graph $G(V, E)$ is assumed to be undirected, and represents the interconnection topology of the system (see examples in Fig. 2a and Fig. 3a); an undirected edge $e = (u, v)$ represents a communication link between the two nodes u and v . Whereas, the comparison graph shows the comparison tests that are performed in order to identify the set of faulty nodes once a fault situation is detected, i.e., when the system deviates from its expected behavior due to faults in the nodes. Examples of comparison graphs are provided in Fig. 2b and Fig. 3b. The set of all comparison outcomes is called the *syndrome*, and it is denoted by σ . The set of all faulty nodes in the system, denoted by F , is called the *fault set*. We will refer to any comparison syndrome that can be generated under F and a given comparison model

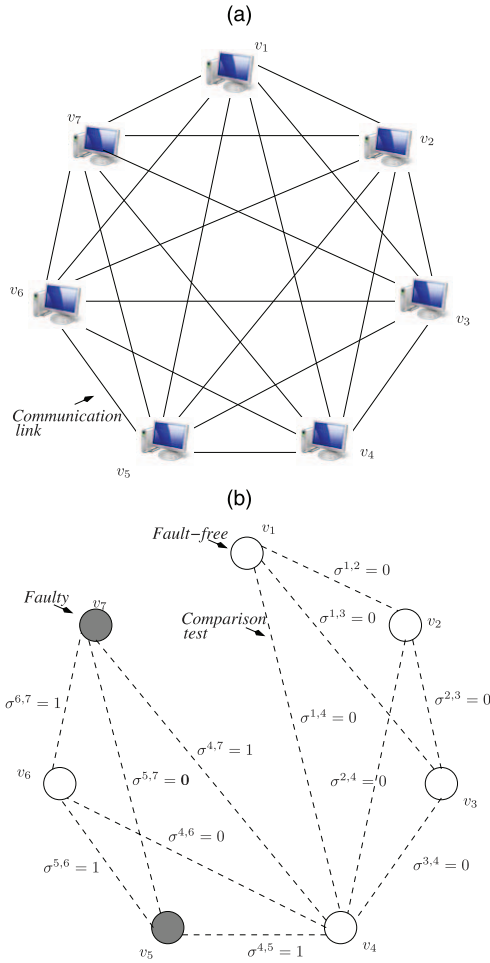


Fig. 2. A three-diagnosable SCM-based system. (a) Communication graph. (b) A comparison assignment and a symmetric comparison syndrome.

by σ_F . The objective of the fault identification algorithm is to identify F given σ_F .

In this work, we consider only the static permanent faults [26], i.e., software or hardware faults that always produce errors when they are fully exercised. However, we consider both hard and soft faults [12]. A hard-faulted node is unable to communicate with the rest of the system, whereas a soft-faulted node can continue to operate and communicate with the other nodes with altered behavior.

Definition 1. A system is t -diagnosable if each node can be correctly identified as fault-free or faulty based on a valid collection of comparison results, assuming that the number of faulty nodes does not exceed a bound t .

The fault diagnosis process is based on the comparison syndrome output by the system's nodes. For a diagnosis to be possible, the behavior of soft faulty nodes should be constrained (or invalidated). Various comparison invalidation rules have been defined leading to different comparison models. The main difference between the comparison models is the assumption on the comparisons involving a pair of faulty nodes, once the comparator node is nonfaulty. Two types of comparison models have been studied in the literature: *symmetric* and *asymmetric* comparisons (see Fig. 1).

Authorized licensed use limited to: University of New South Wales. Downloaded on March 24, 2024 at 10:50:14 UTC from IEEE Xplore. Restrictions apply.

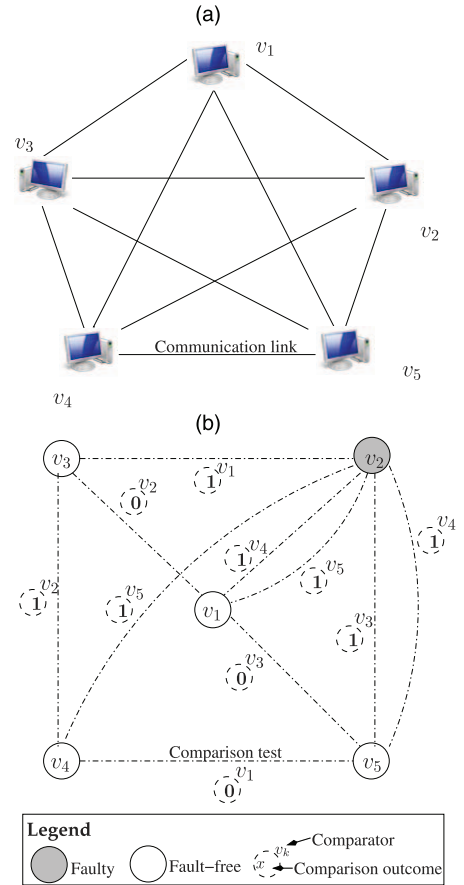


Fig. 3. (a) An interconnection graph. (b) A comparison multigraph.

In the symmetric model, both comparison outcomes are possible in this case (0 or 1), while in the asymmetric model two faulty compared nodes always give mismatching outputs, and hence, the comparison outcome is 1.

In this paper, we consider two of these comparison models—the *Chwa-Hakimi comparison model* developed in [8] and the *generalized comparison model* introduced by Sengupta and Dahbura [10]. The main difference between these two models is that in GCM the comparator node can be one of the nodes being compared. Whereas, in the Chwa-Hakimi model all comparisons are performed by a central observer that monitors the system. However, in both models the diagnosis of faults using the comparison outcomes is performed by the central observer. In the following, a complete description of these two comparison models is provided.

2.1 Chwa-Hakimi Comparison Model

In the comparison model developed by Chwa and Hakimi [8], it is assumed that a central observer (comparator) is responsible of performing the comparisons between pairs of nodes by assigning them some tasks from the set of tasks $T = \{T_1, T_2, \dots\}$. Each pair of nodes v_i and v_j is assigned a task $T_l \in T$. Once the task T_l is completed by both nodes, their results are compared. The comparison graph in this case, is an undirected graph $G = (V, C)$, where V denotes the set of nodes and $C = \{(v_i, v_j) : (v_i, v_j) \text{ is a pair of nodes performing the same task } T_l \in T\}$. We will denote a node pair (v_i, v_j) or (v_j, v_i) by c_{ij} . The result of the comparison test between the nodes v_i and v_j , a binary value, is associated

with c_{ij} . This comparison result is 0 if the results generated by both nodes are identical; and it is 1, otherwise, i.e., if their results mismatch. From now on, we will refer to the Chwa-Hakimi model by the *simple comparison model* (SCM) since it is a special case of the GCM.

The SCM is a symmetric comparison model, that is, the outcome of a comparison test involving a pair of faulty nodes is unreliable (0 or 1). Both test outcomes are possible in this model, while in the asymmetric comparison model (the Malek's model [7]) two faulty nodes always give mismatching outputs.

The notation used for the SCM is as follows:

- Γ_i : denotes the set of nodes with which a node v_i is compared, and is given by $\Gamma_i = \{v_j : c_{ij} \in C\}$.
- σ^{ij} : refers to the outcome of the comparison c_{ij} in the syndrome σ , i.e., the outcome of the comparison conducted between v_i and v_j .
- $\sigma(v_i)$: defines the set of results of the comparison tests that are carried out between the node v_i and all its neighbors, and is given by $\sigma(v_i) = \{(v_j, \sigma^{ij}) : v_j \in \Gamma_i \& c_{ij} \in C\}$.

Definition 2. A comparison syndrome σ is said to be consistent (or compatible) with a fault set $F \subset V$ under SCM if for any $\sigma^{ij} \in \sigma$, such that $v_i \in V - F$, $\sigma^{ij} = 1$ iff $v_j \in F$.

Definition 3. A comparison assignment graph $G(V, C)$ under SCM is a $D_\alpha(|V|)$ design iff for all $v_i \in V$, $|\Gamma_i| \geq \alpha$, i.e., each node is at least compared with α other nodes.

Systems belonging to this special design $D_\alpha(|V|)$ have been shown to be t -diagnosable in [27] and they can be easily generated.

A small system connecting seven nodes is shown in Fig. 2a. A typical comparison assignment is provided in Fig. 2b, and a symmetric comparison syndrome corresponding to the actual fault set $F = \{v_5, v_7\}$ is also given. Note that $\sigma^{5,7} = 0$ according to the symmetric invalidation rules. This example is a SCM-based three-diagnosable system [27].

2.2 Generalized Comparison Model

Maeng and Malek extended in [9] the SCM by allowing the comparisons to be conducted by the nodes themselves for each pair of neighbors with which they can communicate directly. We will refer to their model as the MM* model. A more elegant generalization of testing [4], [5], [6] and comparison models [7], [8], known as the *generalized comparison model*, has been introduced next in [10] by Sengupta and Dahbura, in which the comparator can be one of the two nodes under comparison. Recall that under the testing models nodes test each other directly, i.e., the comparator can be one of the nodes under comparison. According to GCM, if the comparator node is fault-free, then the comparison outcome is 0 if none of the compared nodes is faulty, and it is 1 if one of them is faulty. However, if the comparator itself is faulty, then the comparison outcome is unreliable, and hence, may be 0 or 1 (see Fig. 1).

In this model, the comparison graph is a multigraph whose edges represent comparison tests performed by the system's nodes themselves. Each node is assigned a subset of the other nodes to test. Testing is based on assigning a set of tasks $T = \{T_1, T_2, \dots\}$ to the system's nodes, and

comparing their outcomes. A pair of nodes (v_i, v_j) (or (v_j, v_i)) is assigned a task $T_l \in T$. Once the task T_l is executed by both processing nodes, the results are compared. We represent the comparison multigraph by an undirected graph $M(V, C)$, where V and C denote, respectively, the vertices (nodes) and the edges (comparison tests). For every $v_i, v_j, v_k \in V$, $(v_i, v_j, v_k) \in C$, or simply $c_{ijk} \in C$, iff node v_k tests nodes v_i and v_j by assigning them the same task. Each edge c_{ijk} has a label associated with it. The binary value assigned to the label depends on the GCM's invalidation rules as shown in Fig. 1.

Fig. 3a shows an example of a five-nodes interconnection graph, and Fig. 3b, adopted from [10], depicts an example of a GCM-based comparison multigraph.

In the following, we generalize SCM's notations to GCM. Let $v_i, v_j, v_k \in V$ be any nodes in V . The subscript k will be used to designate a comparator node.

- c_{ijk} : denotes a comparison test performed by node v_k between nodes v_i and v_j , $c_{ijk} \in C$.
- Γ_i : denotes to the set of nodes (neighbors) to which v_i is compared and is defined by $\Gamma_i = \{v_j : \exists v_k \in V \text{ such that } c_{ijk} \in C\}$.
- Γ_i^{-1} : defines the set of all comparators of node v_i and all its neighbors, and is given by $\Gamma_i^{-1} = \{v_k : \exists v_j \in \Gamma_i \text{ such that } c_{ijk} \in C\}$.
- Γ_k^{+1} : denotes the set of node pairs that are compared by the node v_k , and is defined by $\Gamma_k^{+1} = \{(v_i, v_j) : c_{ijk} \in C\}$.
- σ^{ijk} : refers to the outcome of the comparison test c_{ijk} in the syndrome σ , i.e., the result of the comparison test that is conducted by v_k on both nodes v_i and v_j .
- $\sigma(v_k, \Gamma_k^{+1})$: refers to the subset of the syndrome σ containing only the outcomes of comparison tests executed by node v_k , and is defined by $\sigma(v_k, \Gamma_k^{+1}) = \{(v_i, v_j, \sigma^{ijk}) : (v_i, v_j) \in \Gamma_k^{+1}\}$.
- $\sigma(v_i, \Gamma_i)$: refers to the subset of syndrome σ containing only the outcomes of comparisons performed between node v_i and its neighbors, and is given by $\sigma(v_i, \Gamma_i) = \{(v_j, v_k, \sigma^{ijk}) : v_j \in \Gamma_i \& v_k \in \Gamma_i^{-1} \& c_{ijk} \in C\}$.

Definition 4. A syndrome σ is consistent (or compatible) with the fault set $F \subset V$ under GCM if for each $\sigma^{ijk} \in \sigma$ such that $v_k \in V - F$, $\sigma^{ijk} = 1$ iff $v_i \in F$ and/or $v_j \in F$, i.e., at least one compared node is faulty.

Definition 5. A system with interconnection topology represented by the undirected graph $G = (V, E)$ is a $ID_\alpha(|V|)$ design if for each pair of vertices $v_i, v_j \in V$ such that $j - i = k \text{ modulo } |V|$, where $k = 1, 2, \dots, \alpha$, an undirected edge between v_i and v_j exists.

Fig. 4 provides an example of a $ID_3(12)$ interconnection topology.

Definition 6. A multigraph $M = (V, C)$ is a $D_t^*(|V|)$ design if it is produced from a $ID_{t+1}(|V|)$ system topology under the MM* model.

Sengupta and Dahbura showed in [10] that the special design $D_t^*(|V|)$ is t -diagnosable. Note that it can be easily generated even when very large systems are considered.

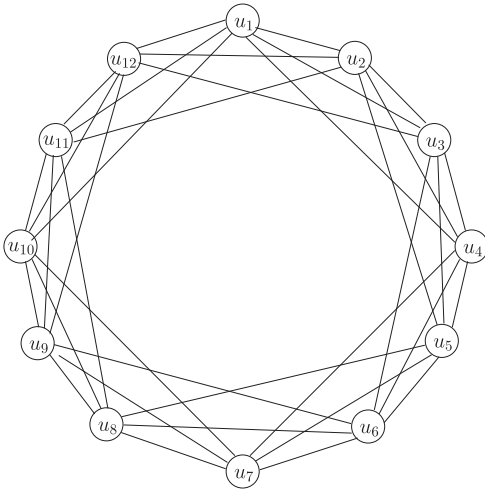


Fig. 4. An example of a $D_3(12)$ interconnection topology.

3 RELATED WORK

Identifying the correct set of all faulty nodes using the comparison approach has been shown to be NP-Hard [14], but if the system is t -diagnosable, the problem is solvable in polynomial time. This problem has been extensively studied leading to various solutions. Nevertheless, the problem of efficiently identifying the set of faulty units when not all the comparison outcomes are available to the diagnosis algorithm at the beginning of the diagnosis phase, i.e., partial syndromes, remains an outstanding research issue. In the following, t , V , and C denote the maximum number of faults allowed in a system, the set of nodes, and the set of comparison tests, respectively. For their symmetric comparison model, Chwa and Hakimi developed an $O(|C|)$ diagnosis algorithm [8]. While, for the asymmetric comparison model, various fault diagnosis algorithms have been proposed. In [28], Ammann and Dal Cin proposed a $O(|V|^2)$ sequential diagnosis algorithm for a subset of t -diagnosable systems. Evolutionary approaches (genetic algorithms [29], artificial immune systems [30], and swarm intelligence [31]) have been also used to solve the comparison-based fault diagnosis problem. Recently, in [12], Chessa and Santi presented a new comparison-based diagnostic model based on one-to-many communication paradigm which takes advantage of the shared nature of ad hoc networks. They introduced a diagnosis protocol and two implementations of their model considering whether the network topology can change during diagnosis or not. Their work has been improved more recently in [32] using a more adaptable approach.

Since the introduction and the characterization of GCM, by Sengupta and Dahbura [10], very little work has been done to tackle the system-level diagnosis problem under GCM. In [10], Sengupta and Dahbura gave necessary and sufficient conditions for GCM-based systems to be t -diagnosable. Moreover, they developed a diagnosis algorithm having a time complexity of $O(|V|^5)$, which makes it impractical specially when considering large systems composed of hundreds or even thousands of nodes. Recently, Yang and Tang [22] developed a more efficient diagnosis algorithm, inspired from the solution provided by Dahbura and Masson

in [33], which requires only $O(|V|\Delta^3\delta)$, where Δ and δ denote the maximum and minimum degrees of a node, respectively. A parallel evolutionary approach has been proposed by Abrougui and Elhadeif [23] to solve the self-diagnosis problem under GCM. The parallel genetic approach has been shown to correctly identify the set of faulty nodes. In [34], Stewart presented a diagnosis algorithm for special systems under the GCM with time complexity $O(\Delta_{max}|V|)$, where Δ_{max} is the maximal degree of any node of the graph.

In [14], Blough and Pelc studied the complexity of fault diagnosis under comparison models and they provided efficient algorithms for diagnosing systems for which the comparison assignment is a bipartite graph. Wang et al. presented in [24] new necessary and sufficient condition for a system to be t -diagnosable under GCM. In addition, they presented a class of systems that uses the minimum number of communication links to obtain a given degree of diagnosability. An interesting distributed diagnosis algorithm has been also introduced in [24] that aims at reducing the number of tests necessary for diagnosis when the number of faults is relatively small. In [11], Blough and Brown introduced an hybrid model by combining distributed diagnosis and GCM. The Blough and Brown's model is known as the *broadcast comparison model*. The originality of the BCM comes from the fact that any two nodes under comparison broadcast their results to all nodes in the system. Once the results are received by all fault-free nodes, they are compared. Based on a sufficient set of comparison outcomes, fault-free nodes can identify the set of faulty ones. The time complexity of the Blough and Brown's algorithm is estimated as $O(|C| + |V|t^2)$. In [13], the *generalized distributed comparison-based model* has been introduced. The GDC model assumes, in addition to all the MM model assumptions, that fault-free nodes execute tasks within a bounded time. The GDC model is fully distributed and does not assume a reliable broadcast system primitive. Albini et al. [13] developed a hierarchical comparison adaptive distributed diagnosis algorithm, called *Hi-Comp*, which is $(|V| - 1)$ -diagnosable and requires $\log_2|V|$ testing rounds, with a maximum number of executed tests evaluated to $O(|V|^3)$. Other extensions of the GDC model have been proposed. Readers are referred to the most recent survey [3] for more details.

In [15], Elhadeif has shown that a perceptron-based neural network can be used to solve the system-level diagnosis problem under the Malek's asymmetric comparison model, and that the perceptron-based diagnosis has failed (i.e., correctness less than 100 percent) when the symmetric SCM was considered. In fact, the diagnosis problem under the Malek's asymmetric comparison model is a separable one given that any fault set produces only one consistent syndrome. That is, any fault set can only be identified by a unique syndrome since all Malek's invalidation rules are reliable [7]. While, under the symmetric SCM a fault set may generate many consistent syndromes since comparisons involving pairs of faulty nodes are unreliable. Hence, the diagnosis problem under the symmetric SCM is a nonseparable one. The present work is an extension of the perceptron-based diagnosis in which a multilayered artificial neural network is used since it is known to be efficient for nonseparable problems. This new diagnosis approach can

efficiently identify the set of faulty units when not all the comparison outcomes are available to the diagnosis algorithm at the beginning of the diagnosis phase. Hence, it becomes attractive for situations where not all comparison outcomes are available, or takes long period of time to compute all of them especially for large systems. Simulation results have shown that the ANN-based fault identification algorithm is efficient, making it a viable addition to present diagnosis techniques.

4 BACKPROPAGATION NEURAL NETWORK (BPNN)-BASED DIAGNOSIS ALGORITHM

The key idea of using artificial neural networks for solving the system-level diagnosis problem was inspired by the fact that ANNs have been extensively studied as classification and/or pattern recognition mechanisms. The fault diagnosis problem under the comparison models can be described as a classification problem which aims at classifying the system's nodes either as faulty or fault-free. It can also be described as a pattern recognition problem which objective is to organize the raw data (i.e., the input syndromes) into meaningful categories (i.e., their corresponding fault sets). In recent years an enormous number of publications on refinements and improvements of various types of neural networks have been published. However most of the suggested improvements are only useful if the problem meets certain conditions [35]. Nevertheless, the multilayer feedforward networks trained with the backpropagation method are probably the most practically used networks for real world applications.

In this work, we will mainly focus on backpropagation neural networks which have been extensively studied in the literature, and various surveys, tutorials, and implementations exist [36]. As a matter of fact, in this section, we will not go into too much details on how to implement a BPNN, rather then we will focus on how to adapt BPNNs to the system-level diagnosis problem under comparison models. However, in order to make the paper self-contained, we included below a short description and the main steps of our BPNN-based diagnosis algorithm.

4.1 Diagnosis Algorithm

The multilayer neural network that will be used to model the system-level diagnosis problem can be described as follows:

$$x^0 \xrightarrow{W^1, b^1} x^1 \xrightarrow{W^2, b^2} \dots \xrightarrow{W^L, b^L} x^L$$

where L denotes the number of layers and $x^0 = \sigma$, that is, the input layer is the set of comparison outcomes. The other layers are of various sizes depending on the size of the systems (see Section 5 for more details). For all $l = 1, \dots, L$, W^l is an $n_l \times n_{l-1}$ weight matrix for layer l . There are $L + 1$ layers of neurons, and L layers of synaptic weights. The first layer is the input layer and the last layer is the output layer. The aim of the BPNN is to change the weights W and biases b so that the actual output x^L becomes closer to the desired output d .

Fig. 5 describes an example of the neural network that will be created for the comparison multigraph and syndrome included in Fig. 3b. Note from the figure that a comparison outcome affects only the nodes it involves. For example, the

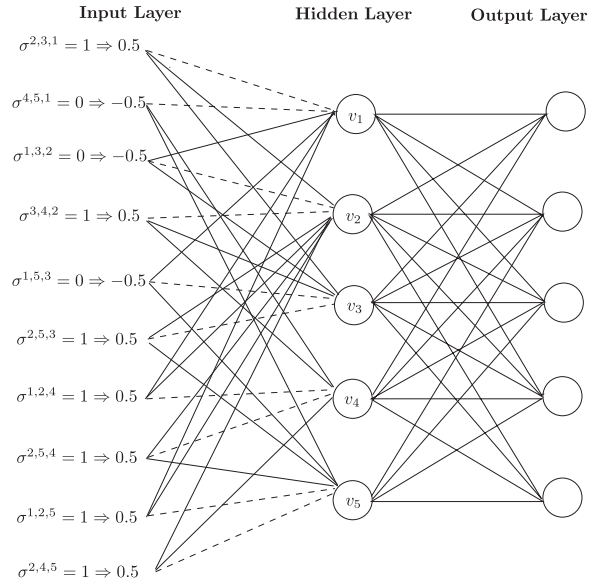


Fig. 5. The multilayer neural network generated for the comparison multigraph and syndrome of Fig. 3.

comparison outcome $\sigma^{2,3,1} = 1$ affects the neuron v_1 since it is the comparator (showed in dashed line in Fig. 5), and the neurons v_2 and v_3 since these are the compared nodes. Note also that we have converted the comparison outcomes as follows: a 0 comparison outcome is converted to -0.5 , while a 1 comparison outcome is converted to 0.5 .

The backpropagation algorithm consists of the following main steps.

- **Step 1: Forward pass.** The input vector x^0 is transformed into the output vector x^L , by evaluating the following equation: $\forall l = 1, \dots, L$

$$x_i^l = f(\theta_i^l) = f\left(\sum_{j=1}^{n_{l-1}} W_{ij}^l x_j^{l-1} + b_i^l\right), \quad (1)$$

where n_l denotes the size of the layer l . The activation function f used for each neuron is the sigmoid function: $f(x) = \frac{1}{1+\exp^{-x}}$ which has the very nice property that its derivative is given by $f'(x) = f(x)(1 - f(x))$. In other words, the forward pass step updates the output value for each neuron. That is, for each layer l , starting with the first hidden layer, it takes the input vector x^{l-1} to each neuron and finds the output by first calculating the weighted sum of inputs and then applying the sigmoid function f to it, and passes it forward to the next layer until the output layer is updated.

- **Step 2: Error computation.** The difference between the desired output d and the actual output x^L is computed

$$\begin{aligned} \delta_i^L &= f'(\theta_i^L)(d_i - x_i^L) \\ &= f(\theta_i^L)(1 - f(\theta_i^L))(d_i - x_i^L), \end{aligned}$$

where f' is the derivative of the sigmoid activation function f .

- **Step 3: Backward pass.** The error signal at the output units is propagated backwards through the entire network, by evaluating

Procedure DiagnoseFaultySituation()**Inputs:** V , Comparison assignment C , Input syndrome σ **Outputs:** FaultSet F , FaultFreeSet FF **Begin** $x^0 = \sigma$

$$x_i^l = f \left(\sum_{j=1}^{n_{l-1}} W_{ij}^l x_j^{l-1} + b_i^l \right), \forall l = 1, \dots, L$$

for i from 1 to $|V|$ **do** $x_i^L = \text{ComputeNeuronOutput}(i);$ // using Equation (1) $\text{ConvertNetworkOutput}(x_i^L)$ **End**

Fig. 6. Pseudocode for BPNN-based diagnosis.

$$\delta_j^{l-1} = f'(\theta_j^{l-1}) \sum_{i=1}^n \delta_i^l W_{ij}^l, \forall l = 2, \dots, L.$$

- **Step 4: Learning updates.** The synaptic weights and biases are updated using the results of the forward and backward passes,

$$\Delta W_{ij}^l = \eta \delta_i^l x_j^{l-1}, \forall l = 1, \dots, L,$$

$$\Delta b_i^l = \eta \delta_i^l, \forall l = 1, \dots, L,$$

$$W(t+1) = W(t) + \Delta W(t) + \alpha \Delta W(t-1),$$

where t refers to the time or to the epoch number.

- **Step 5: Stopping condition.** The above four steps are repeated using a training set of fault situations and their corresponding consistent syndromes until a stopping criterion is satisfied. Possible stopping criteria in error backpropagation learning include: 1) total error of the network falling below some predetermined level, 2) a certain number of epochs having been completed (as shown in our simulation results in Section 5), or 3) combinations of the two (e.g., whichever of the two occurs first). Other stopping conditions are also possible [35].

The learning coefficient η determines the size of the weight changes. A small value for η will result in a very slow learning process. If the learning coefficient is too large it will result in large weight changes that may cause the desired minimum to be missed. A useful range is between 0.05 and 2 dependent on the problem. An improved technique is to use an adaptive learning rate. A large initial learning coefficient should help to escape from local minima, while reducing η later should prevent the learning process from overshooting the reached minimum.

The momentum α causes the weight changes to be dependent on more than one input pattern. The change is a linear combination of the current gradient and the previous gradient. The useful range for this parameter is between 0 and 1. For some data sets the momentum makes the training faster, while for others there may be no improvements. The momentum usually makes it less likely that the training process will get stuck in a local minimum.

Once the neural network is trained it can be used to diagnose any fault situation that may occur. The pseudocode for diagnosing a fault set is given in Fig. 6.

An important step in our diagnosis algorithm is how to convert the neural network outputs to a fault set. We

Procedure ConvertNetworkOutputSCM**Inputs:** x^L **Outputs:** FaultSet F , FaultFreeSet FF **Begin****for** i from 1 to $|V|$ **do****if** $(x_i^L \leq 0.5)$ $FF = FF \cup \{v_i\};$ **else** $F = F \cup \{v_i\};$ **End**

Fig. 7. Pseudocode for converting the neural network outputs to a fault set under SCM.

TABLE 1
Neurons' Outputs for a $D_4^*(10)$ GCM-Based Multigraph

i	x_i^L
8	0.999809
5	0.651824
9	0.646183
3	0.605281
7	0.503175
6	0.352320
4	0.002018
0	0.000361 ◀
2	0.000174 ◀◀
1	0.000005 ◀◀◀

referred to this in Fig. 6 by calling the procedure $\text{ConvertNetworkOutput}(x^L)$. The objective of such procedure is to convert the output vector x^L into the set of faulty nodes F and the set of fault-free ones FF . For SCM, we used a simple mapping function $\text{ConvertNetworkOutputSCM}$, see Fig. 7, that maps any output less than or equal to 0.5 to a faulty state. Given the simplicity of SCM, this conversion provided good results as shown in Section 5.

For GCM, we have adopted a different approach. To clearly explain how we extract the set of faulty nodes consider the neurons' outputs, sorted in a descending order as shown in Table 1, for the comparison multigraph $D_4^*(10)$. Faulty nodes are pointed by ◀. First note that the BPNN was able to separate between the two classes: the faulty nodes and the fault-free ones. However, from the extensive simulations, we have conducted we have noticed that the boundary between the two classes is not well defined all the time. Hence, we have adopted the following heuristic to be able to extract the set of faulty nodes by using the comparison multigraph C and the input syndrome σ_F .

The heuristic proceeds in the following steps:

1. Set position variable pos to the start of the array.
2. Label the node in position pos as fault-free and add it to the set $PendingFF$. In the provided example, the state of node v_8 will be fault-free, and the set $PendingFF = \{v_8\}$. Increase value of pos by one.
3. Repeat the following steps until all nodes are labeled either as fault-free or as faulty, or $pendingFF$ is empty.

- a. Consider $v_k \in PendingFF$ if not empty. For each pair of nodes $(v_i, v_j) \in \Gamma_k^{+1}$, if $\sigma_F^{ijk} = 0$ and the state of one of the two compared nodes v_i and v_j is fault-free, then set the state of the unknown node as fault-free and add it to $pendingFF$; otherwise, set it to faulty if the number of faulty

nodes did not reach the bound t . If $\sigma_F^{ijk} = 1$ and the state of one of the two compared nodes is fault-free, then set the state of the unknown node as faulty if the number of faulty nodes did not reach the bound t ; otherwise set it to fault-free.

- b. If $\text{pendingFF} = \emptyset$, then goto Step 2.

The described heuristic has been implemented and extensively tested as shown in Section 5. In all scenarios, this heuristic has been successful in correctly converting the neurons' outputs to faulty or fault-free states.

4.2 Avoiding Local Optima

In order to avoid local optima a postprocessing phase has been developed. In fact, during the simulations of the new BPNN-based diagnosis algorithm, we have noticed that the neural network sometimes reached local optima; hence, providing us with incorrect results especially when the number of faulty nodes is too high, i.e., reaches the bound t . One of the main reasons could be either lack of training (as we stop the training after a fixed number of training epochs), or it could be the size of the neural network (the number of layers and the number of neurons per layer) as this was determined only based on the number of nodes in the system. In order to correct this deficiency, we developed a postprocessing phase that corrects the solution proposed by the BPNN-based diagnosis. The postprocessing phase takes as input the potential fault set \mathbb{F} output by the neural network, and corrects it. In the following, we describe how it works and we formalize it. In this formalization, we will refer only to the GCM, and we will show that it can be easily specialized to the SCM.

Let's consider a fault situation, where F is the actual fault set and σ_F denotes one of its compatible syndromes. Consider a potential fault set \mathbb{F} (output by our BPNN-based diagnosis algorithm) and let $\sigma_{\mathbb{F}}$ denote one of its compatible syndromes. Intuitively if both syndromes σ_F and $\sigma_{\mathbb{F}}$ are identical, then one can easily conclude that $F = \mathbb{F}$, i.e., \mathbb{F} is the actual fault set, since we are considering only t -diagnosable systems. To check whether syndromes σ_F and $\sigma_{\mathbb{F}}$ are identical, we compute the probability, P , of both fault sets being identical, and is given by

$$P(\sigma_{\mathbb{F}}, \sigma_F) = \frac{\sum_{v \in V} P(\sigma_{\mathbb{F}}, \sigma_F, v)}{|V|}, \quad (2)$$

where $P(\sigma_{\mathbb{F}}, \sigma_F, v)$ denotes the probability of v 's state being correct according to fault set F and is defined by

$$P(\sigma_{\mathbb{F}}, \sigma_F, v) = \frac{P^{+1}(\sigma_{\mathbb{F}}, \sigma_F, v) + P^{-1}(\sigma_{\mathbb{F}}, \sigma_F, v)}{2}, \quad (3)$$

$$P^{+1}(\sigma_{\mathbb{F}}, \sigma_F, v) = \frac{|\sigma_{\mathbb{F}}(v, \Gamma_v^{+1}) \cap \sigma_F(v, \Gamma_v^{+1})|}{|\Gamma_v^{+1}|}, \quad (4)$$

$$P^{-1}(\sigma_{\mathbb{F}}, \sigma_F, v) = \frac{|\sigma_{\mathbb{F}}(v, \Gamma_v) \cap \sigma_F(v, \Gamma_v)|}{|\Gamma_v|}. \quad (5)$$

Note that the probability of v 's state being correct according to fault set F is composed of two components: $P^{+1}(\sigma_{\mathbb{F}}, \sigma_F, v)$ and $P^{-1}(\sigma_{\mathbb{F}}, \sigma_F, v)$. $P^{+1}(\sigma_{\mathbb{F}}, \sigma_F, v)$ measures the correctness probability of v 's state as a tester (or

comparator) node, and $P^{-1}(\sigma_{\mathbb{F}}, \sigma_F, v)$ calculates that as a tested (or compared) node. We assume that for the special case where $|\Gamma_v^{+1}| = 0$, i.e., v does not perform any comparisons, $P^{+1}(\sigma_{\mathbb{F}}, \sigma_F, v) = 1$.

In the SCM, any node v does not perform comparisons, and hence, the set Γ_v^{+1} is empty. It follows that for SCM, $P^{+1}(\sigma_{\mathbb{F}}, \sigma_F, v) = 1, \forall v \in V$, and hence (3) can be reduced to

$$P(\sigma_{\mathbb{F}}, \sigma_F, v) = \frac{|\sigma_{\mathbb{F}}(v) \cap \sigma_F(v)|}{|\Gamma_v|}. \quad (6)$$

Equations (4) and (5) state that the individual probabilities can be easily computed by checking how many comparison outcomes, in both syndromes σ_F and $\sigma_{\mathbb{F}}$, corresponding to comparison tests performed between the node v and all its neighbors are identical. Note that $|\Gamma_v^{+1}|$ and $|\Gamma_v|$ are used just as normalization factors. It follows that $P(\sigma_{\mathbb{F}}, \sigma_F, v) \in [0, 1]$. When $P(\sigma_{\mathbb{F}}, \sigma_F, v) = 1$ this means that according to node v (local view) both syndromes are identical, and hence, suggests a common origin, i.e., the same fault set. In addition, $P(\sigma_{\mathbb{F}}, \sigma_F) \in [0, 1]$ and can be seen as the degree of resemblance of both syndromes or as the correctness probability of the potential fault set \mathbb{F} .

The postprocessing phase consists in calculating $P(\sigma_{\mathbb{F}}, \sigma_F)$. If $P(\sigma_{\mathbb{F}}, \sigma_F) = 1$ then the fault set is found; Otherwise, the proposed potential solution needs to be corrected as follows. The nodes are ordered, in an ascending order, based on $P(\sigma_{\mathbb{F}}, \sigma_F, v), \forall v \in V$. Then, the state of the node with the smallest $P(\sigma_{\mathbb{F}}, \sigma_F, v)$ is changed, i.e., flipped from faulty (1) to fault-free (0) and vice-versa, and the postprocessing phase is repeated. The stopping criteria is either $P(\sigma_{\mathbb{F}}, \sigma_F) = 1$, that is the potential fault set \mathbb{F} was corrected, or the number of changes has exceeded the size of the system, $|V|$, i.e., the correction failed. In Section 5, we show the efficiency of such postprocessing phase.

Computing $P(\sigma_{\mathbb{F}}, \sigma_F)$ could be time consuming especially for large systems. In fact, the time complexity of this step is $O(|C|)$. In a comparison graph generated from a fully connected interconnection graph using the MM* model, we have $|C| = |V|(|V| - 2)(|V| - 3)$ comparison tests. Hence, $O(|C|) \cong O(|V|^3)$. To overcome this, we have developed a method for interpolating $P(\sigma_{\mathbb{F}}, \sigma_F)$ from another precomputed value of $P(\sigma_{\emptyset}, \sigma_F)$. The interpolation process has reduced the time complexity of the computing $P(\sigma_{\mathbb{F}}, \sigma_F)$ to $O(t|V|^2)$. In the following, we describe in details how this interpolation works, and we provide a concrete example to help understand this interpolation process.

4.3 Interpolation Process

To speedup the postprocessing phase, and hence reduce the time taken to correct a potential solution output by our BPNN-based diagnosis algorithm we have developed an interpolation method whose objective is to compute $P(\sigma_{\mathbb{F}}, \sigma_F)$ from a pre-computed value of $P(\sigma_{\emptyset}, \sigma_F)$. Before we describe in details how it works we need first to introduce simplified versions of (2)-(5) by omitting the normalization factors. This will help in avoiding the problem of rounding-off errors in computations based on small probabilities. The simplified versions are as follows:

TABLE 2

Syndromes for Fault Sets $F = \{v_2\}$, $\mathbb{F} = \{v_1\}$, and \emptyset

$\sigma^{i,j,k}$	σ_F	σ_\emptyset	$\sigma_{\mathbb{F}}$
$\sigma^{2,3,1}$	1	0	0
$\sigma^{4,5,1}$	0	0	1
► $\sigma^{1,3,2}$	0	0	1
$\sigma^{3,4,2}$	1	0	0
► $\sigma^{1,5,3}$	0	0	1
$\sigma^{2,5,3}$	1	0	0
► $\sigma^{1,2,4}$	1	0	1
$\sigma^{2,5,4}$	1	0	0
► $\sigma^{1,2,5}$	1	0	1
$\sigma^{2,4,5}$	1	0	0

$$\mathbb{P}(\sigma_{\mathbb{F}}, \sigma_F) = \sum_{v \in V} \mathbb{P}(\sigma_{\mathbb{F}}, \sigma_F, v),$$

$$\mathbb{P}(\sigma_{\mathbb{F}}, \sigma_F, v) = \mathbb{P}^{+1}(\sigma_{\mathbb{F}}, \sigma_F, v) + \mathbb{P}^{-1}(\sigma_{\mathbb{F}}, \sigma_F, v),$$

$$\mathbb{P}^{+1}(\sigma_{\mathbb{F}}, \sigma_F, v) = |\sigma_{\mathbb{F}}(v, \Gamma_v^{+1}) \cap \sigma_F(v, \Gamma_v^{+1})|, \quad (7)$$

$$\mathbb{P}^{-1}(\sigma_{\mathbb{F}}, \sigma_F, v) = |\sigma_{\mathbb{F}}(v, \Gamma_v) \cap \sigma_F(v, \Gamma_v)|. \quad (8)$$

The new versions are based on integer calculations, and hence, the stopping criteria $P(\sigma_{\mathbb{F}}, \sigma_F) = 1$ turns out to be now $\mathbb{P}(\sigma_{\mathbb{F}}, \sigma_F) = 3|C|$ since a comparison test is counted three times: one in (7) as a comparator and twice in (8) one for each compared node.

The interpolation method we used takes as inputs the pre-computed values of $\mathbb{P}^{+1}(\sigma_\emptyset, \sigma_F, v)$ and $\mathbb{P}^{-1}(\sigma_\emptyset, \sigma_F, v)$. To interpolate $\mathbb{P}(\sigma_{\mathbb{F}}, \sigma_F)$ from $\mathbb{P}(\sigma_\emptyset, \sigma_F)$ we proceed as follows:

I1. For each $u \in V - \mathbb{F}$,

$$\mathbb{P}^{+1}(\sigma_{\mathbb{F}}, \sigma_F, u) = \mathbb{P}^{+1}(\sigma_\emptyset, \sigma_F, u),$$

$$\mathbb{P}^{-1}(\sigma_{\mathbb{F}}, \sigma_F, u) = \mathbb{P}^{-1}(\sigma_\emptyset, \sigma_F, u).$$

- I2. For each $v_k \in \mathbb{F}$, compute $\mathbb{P}^{+1}(\sigma_{\mathbb{F}}, \sigma_F, v_k)$ and adjust the values of $\mathbb{P}^{-1}(\sigma_{\mathbb{F}}, \sigma_F, v_i)$ and $\mathbb{P}^{-1}(\sigma_{\mathbb{F}}, \sigma_F, v_j)$ for all $(v_i, v_j) \in \Gamma_k^{+1}$, only if $\sigma_{\mathbb{F}}^{ijk} \neq \sigma_\emptyset^{ijk}$.
- I3. For each $v_i \in \mathbb{F}$, compute $\mathbb{P}^{-1}(\sigma_{\mathbb{F}}, \sigma_F, v_i)$ and adjust the values of $\mathbb{P}^{-1}(\sigma_{\mathbb{F}}, \sigma_F, v_j)$ and $\mathbb{P}^{+1}(\sigma_{\mathbb{F}}, \sigma_F, v_k)$ for all $v_j \in \Gamma_i$ and $v_k \in \Gamma_j^{-1}$, only if $\sigma_{\mathbb{F}}^{ijk} \neq \sigma_\emptyset^{ijk}$.

The adjustment of the values in Steps I2 and I3 is done by either adding or subtracting one, and it is performed using the following rule: if $\sigma_F^{ijk} = \sigma_{\mathbb{F}}^{ijk}$, add one; otherwise, subtract one. A concrete example is shown below to help understand the interpolation process.

The time complexity of the interpolation process is reduced to $O(t|V|^2)$. In fact, in the worst case a node can be involved in $(n-2)(n-3)$ comparison tests as a comparator, where $n = |V|$. Hence, step I2 can be done in $O(tn^2)$, since $|F| = t$ in the worst case. On the other hand, a node can be part of $(n-1)(n-2)$ comparison tests as a compared node. It follows that the time complexity of step I3 is also $O(tn^2)$, since in the worst case $|F| = t$. As a result, the time complexity of the interpolation process is $O(t|V|^2)$.

4.3.1 How Does Interpolation Work?

Consider the comparison multigraph of Fig. 3b. The actual fault set is $F = \{v_2\}$. A consistent syndrome for each of the fault sets F , $\mathbb{F} = \{v_1\}$, and \emptyset is provided in Table 2

TABLE 3

Precomputed and Expected Values

(A) Pre-computed values for empty fault set					
v_i	v_1	v_2	v_3	v_4	v_5
$\mathbb{P}^{+1}(\sigma_\emptyset, \sigma_F, v_i)$	1	1	1	0	0
$\mathbb{P}^{-1}(\sigma_\emptyset, \sigma_F, v_i)$	2	0	1	1	2

(B) Expected values for fault set $\mathbb{F} = \{v_1\}$					
v_i	v_1	v_2	v_3	v_4	v_5
$\mathbb{P}^{+1}(\sigma_{\mathbb{F}}, \sigma_F, v_i)$	0	0	0	1	1
$\mathbb{P}^{-1}(\sigma_{\mathbb{F}}, \sigma_F, v_i)$	2	2	0	0	0

TABLE 4
Interpolation Process

v_i	$\mathbb{P}^{+1}(\sigma_{\mathbb{F}}, \sigma_F, v_i)$	$\mathbb{P}^{-1}(\sigma_{\mathbb{F}}, \sigma_F, v_i)$
v_1	$= 0$	$= 2$
v_2	$1 - 1^b = 0$	$0 + 1^d + 1^e = 2$
v_3	$1 - 1^c = 0$	$1 - 1^b = 0$
v_4	$0 + 1^d = 1$	$1 - 1^a = 0$
v_5	$0 + 1^e = 1$	$2 - 1^a - 1^c = 0$

a Rule I2: since $\sigma_F^{4,5,1} \neq \sigma_{\mathbb{F}}^{4,5,1}$, then subtract one. b Rule I3: since $\sigma_F^{1,3,2} \neq \sigma_{\mathbb{F}}^{1,3,2}$, then subtract one. c Rule I3: since $\sigma_F^{1,5,3} \neq \sigma_{\mathbb{F}}^{1,5,3}$, then subtract one. d Rule I3: since $\sigma_F^{1,2,4} = \sigma_{\mathbb{F}}^{1,2,4}$, then add one. e Rule I3: since $\sigma_F^{1,2,5} = \sigma_{\mathbb{F}}^{1,2,5}$, then add one.

assuming symmetric invalidation rules, where \mathbb{F} is the potential fault set output by the diagnosis algorithm. We will show in details the interpolation of $P(\sigma_{\mathbb{F}}, \sigma_F)$ from $P(\sigma_\emptyset, \sigma_F)$.

First note that $P(\sigma_\emptyset, \sigma_F)$ is already precomputed, that is, for each $v_i \in V$, $\mathbb{P}^{+1}(\sigma_\emptyset, \sigma_F, v_i)$ and $\mathbb{P}^{-1}(\sigma_\emptyset, \sigma_F, v_i)$ are known as shown in Table 3A. The objective is to interpolate those of fault set \mathbb{F} . The interpolation rule I1 is straightforward. Let consider rule I2, we compute $\mathbb{P}^{+1}(\sigma_{\mathbb{F}}, \sigma_F, v_1)$ following (7), and at the same time we will adjust only $\mathbb{P}^{-1}(\sigma_{\mathbb{F}}, \sigma_F, v_4)$ and $\mathbb{P}^{-1}(\sigma_{\mathbb{F}}, \sigma_F, v_5)$ as we have $\sigma_{\mathbb{F}}^{4,5,1} \neq \sigma_\emptyset^{4,5,1}$. Since $\sigma_{\mathbb{F}}^{4,5,1} \neq \sigma_F^{4,5,1}$, the adjustment is done by subtracting one.

Finally, we consider the interpolation Rule I3. We compute $\mathbb{P}^{-1}(\sigma_{\mathbb{F}}, \sigma_F, v_1)$ using (8), and simultaneously we adjust $\mathbb{P}^{-1}(\sigma_{\mathbb{F}}, \sigma_F, v_j)$ and $\mathbb{P}^{+1}(\sigma_{\mathbb{F}}, \sigma_F, v_k)$ for all $v_j \in \Gamma_1 = \{v_2, v_3, v_5\}$ and $v_k \in \Gamma_j^{-1}$, only if $\sigma_{\mathbb{F}}^{1jk} \neq \sigma_\emptyset^{1jk}$. Note that $\sigma_{\mathbb{F}}^{1jk} \neq \sigma_\emptyset^{1jk}, \forall j, k$ (pointed by ► in Table 2). Since $\sigma_F^{1,3,2} \neq \sigma_{\mathbb{F}}^{1,3,2}$, it follows that the adjustment is by subtracting one from $\mathbb{P}^{-1}(\sigma_{\mathbb{F}}, \sigma_F, v_3)$ and $\mathbb{P}^{+1}(\sigma_{\mathbb{F}}, \sigma_F, v_2)$. Similarly, since $\sigma_F^{1,5,3} \neq \sigma_{\mathbb{F}}^{1,5,3}$ we subtract one from $\mathbb{P}^{-1}(\sigma_{\mathbb{F}}, \sigma_F, v_5)$ and from $\mathbb{P}^{+1}(\sigma_{\mathbb{F}}, \sigma_F, v_3)$.

Now, since $\sigma_F^{1,2,4} = \sigma_{\mathbb{F}}^{1,2,4}$, it follows that the adjustment is by adding one to $\mathbb{P}^{-1}(\sigma_{\mathbb{F}}, \sigma_F, v_2)$ and $\mathbb{P}^{+1}(\sigma_{\mathbb{F}}, \sigma_F, v_4)$. Similarly, we perform the same adjustment when considering $\sigma_{\mathbb{F}}^{1,2,5}$. As result, we get the values $\mathbb{P}^{+1}(\sigma_{\mathbb{F}}, \sigma_F, v_i)$ and $\mathbb{P}^{-1}(\sigma_{\mathbb{F}}, \sigma_F, v_i)$ shown in Table 4. Note that they are identical to the ones we were expecting (see Table 3B).

5 SIMULATION RESULTS

We have developed two versions of the new diagnosis algorithm—one for the simple comparison model and the other for the generalized comparison model. Randomly

generated t -diagnosable and nondiagnosable comparison graphs have been used for the experiments. To make the generation of the various comparison graphs easy, we used t -diagnosable comparison graphs from the special designs, $D_\alpha(|V|)$ and $D_t^*(|V|)$, introduced in Definitions 3 and 6. We have also experimented the BPNN-based diagnosis algorithm with partial syndromes, i.e., when not all the comparison outcomes are available to the diagnosis algorithm at the beginning of the diagnosis phase.

The BPNN-based diagnosis algorithm relies mainly on five parameters, which are: the number of hidden layers $numLayers$, the number of neurons per layer, the momentum α , the learning rate η , and the number of epochs $maxEpochs$. The momentum α is set to 0.5 for GCM, and to 0.1 for SCM. The learning rate η was fixed to 0.5 and gradually decreased at the end of the learning phase depending on the number of epochs. Let $n = |V|$. The number of hidden layers was set as follows: if $n \leq 50$, $numLayers = 3$, else if $n \leq 100$, $numLayers = 4$, else $numLayers = 5$. Each hidden layer is composed of $n + \log(n \bmod 10)n$ neurons (if $n \bmod 10 = 0$, then we have chosen $n + \log(\frac{n}{10} + n)n$). In Appendix A, which can be found on the Computer Society Digital Library at <http://doi.ieeecomputersociety.org/10.1109/TPDS.2011.248>, we have included the description and the results of the experiments, we have conducted to find these settings.

To clearly show that the BPNN-based fault diagnosis algorithm works efficiently, and to check its practical performance, we have conducted various extensive experiments using both diagnosable and nondiagnosable systems. We have first showed that the BPNN-based diagnosis works correctly for t -diagnosable systems. Then, we have considered nondiagnosable systems, and we have showed that the new diagnosis approach provided good results even though not all the comparison outcomes were available to the diagnosis algorithm at the beginning of the diagnosis phase. In the following, we describe the results of the various experiments we have conducted.

5.1 Performance Using t -Diagnosable Systems

The BPNN-based fault diagnosis algorithm has been implemented in C++, and it has been extensively simulated on a PC equipped with an Intel Core 2 QUAD Q8300 CPU 2.5 GHz and 4 GB of RAM. Randomly generated t -diagnosable comparison graphs have been used for the experiments. In addition, all possible fault sets that may occur in a t -diagnosable system have been simulated, when it is possible, by varying the number of faults from 1 to t . However, a more practical approach would be by generating fault sets according to the reliability and the probability of failure of the nodes. But, we have decided to generate the faults randomly in order to be able to test the performance of our fault identification algorithm even when fault sets with very low probability of occurring are considered.

5.1.1 Performance under SCM

In order to check the performance of the new diagnosis algorithm under SCM, we have run various types of simulations. In the first set of experiments, we considered a $D_9(20)$ symmetric SCM-based comparison graph. We have created a BPNN that was extensively trained during 10,000 epochs. We have then tested it using all possible fault

sets of cardinality ranging from 1 to 9. For each fault set with cardinality c , $1000c$ randomly generated syndromes have been tested. In all these tested fault situations the BPNN-based diagnosis was able to identify the corresponding faulty nodes. That is, 100 percent correctness.

In the second set of simulations, we have considered a large symmetric SCM-based graph $D_{24}(100)$. We have created and trained a BPNN for this large system, and then we have tested it using 100,000 randomly generated fault sets of cardinality in the range $[1..24]$. All fault situations have been correctly identified by the diagnosis algorithm, resulting hence in a 100 percent correctness.

In another set of experiments, we have run the diagnosis algorithm for various symmetric SCM-based comparison graphs ranging from small systems composed of tens of nodes to large systems composed of hundreds of nodes. The number of nodes, n , was varied from 10 to 1,000 with different paces as follows. If $n \in [10..100[$ the pace is 10. But, if $n \in [100..1000]$ the considered pace was 100. For each comparison graph, a neural network have been created and trained during 10,000 epochs. Then, we have randomly generated, for each value of n , $100n$ comparison graphs and tested them for $1,000n$ times using randomly generated fault sets, and where the maximum number of faults $maxFaults$ was also random. The size of the randomly generated fault sets was equally distributed in the range from 1 to $maxFaults$. Over all these extensive simulations, the diagnosis algorithm was able to determine the exact fault sets, providing us hence with almost 100 percent correctness. In fact, the diagnosis algorithm has missed very few fault sets specifically those very large.

5.1.2 Performance under GCM

Due to the fact that both characterizations of GCM-based t -diagnosable systems, the one provided by Sengupta and Dahbura [10] and that developed by Wang et al. [24], are rather theoretical and difficult to implement specially for large systems, we have used the special class $D_t^*(|V|)$ of diagnosable systems [10] provided in Definition 6. Since the BPNN-based diagnosis approach does not rely on the comparison multigraph topology, then it follows that the following results remain valid for general graph structures. The simulations we have conducted can be grouped into three sets of experiments.

In the first set of experiments, we considered the GCM-based t -diagnosable systems $D_4^*(10)$, $D_6^*(15)$, $D_9^*(20)$, and $D_{11}^*(25)$ and all possible combinations of faults. That is, we have experimented our diagnosis algorithm with all possible fault sets with cardinality ranging from 1 to t . Note that for small systems, we were able to generate all possible fault sets which are bounded by $\sum_{i=1}^t C_i^n$, where $n = |V|$ and $C_i^n = \frac{n!}{(n-i)!i!}$. For all fault sets with cardinality smaller than t , our BPNN-based fault diagnosis algorithm was able to identify all faulty nodes. Fig. 8 depicts the diagnosis correctness. However, for very large fault sets with cardinality approaching the bound t the diagnosis correctness was around 99.5 percent. Even though, we were expecting a 100 percent correctness, we are not worried about this small degradation in performance as in practice the probability of having such fault situation, that is, almost half of the system's nodes faulty at the same time, is very low. Such extreme fault situation are rare, if

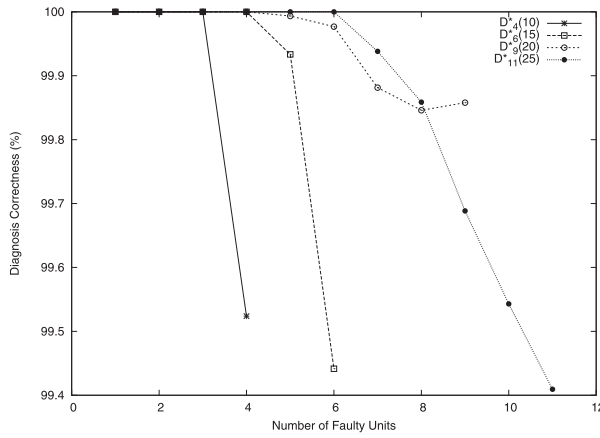


Fig. 8. Diagnosis correctness versus number of faulty nodes using $D_4^*(10)$, $D_6^*(15)$, $D_9^*(20)$, and $D_{11}^*(25)$.

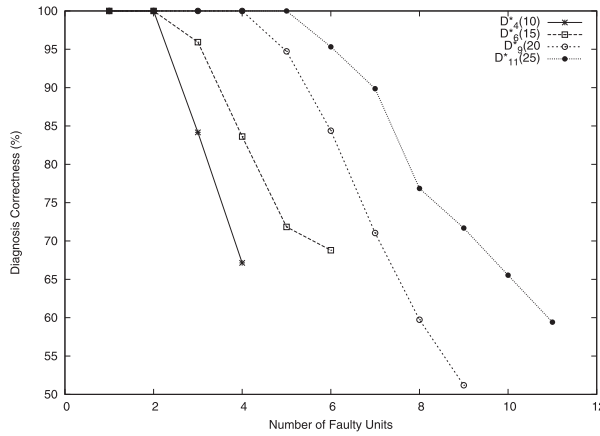


Fig. 9. Diagnosis performance under systematic faulty scenarios using $D_4^*(10)$, $D_6^*(15)$, $D_9^*(20)$, and $D_{11}^*(25)$.

not impossible. In addition, we believe that this performance could be improved by changing our postprocessing phase to handle these special cases where the number of faulty nodes approaches the bound t . This improvement is under investigation.

We have conducted a second type of experiments where more systematic faulty scenarios have been considered, i.e., where all faulty nodes respond in the same way. Fig. 9 shows the results when considering that all comparisons conducted by faulty nodes will provide 0s as outcomes, and all outcomes of comparisons conducted between two faulty nodes will provide 1s. We can notice, compared to Fig. 8, that there is a degradation in the diagnosis performance especially when the number of faults is very high (almost t). We believe that this can be improved by a newer version of the postprocessing phase that takes into account the number of 1s in the syndrome to estimate the number of faults, and hence, help in correctly updating the neural network's outputs.

In a third set of experiments, we looked at the scalability of the BPNN-based system-level diagnosis approach. When considering larger diagnosable systems, our diagnosis algorithm performed very well even in the worst case scenarios that were generated randomly. All these results confirm one thing, neural networks can be used to solve the system-level fault diagnosis problem, and they can be considered as a viable addition to existing diagnosis solutions.

Authorized licensed use limited to: University of New South Wales. Downloaded on March 24, 2024 at 10:50:14 UTC from IEEE Xplore. Restrictions apply.

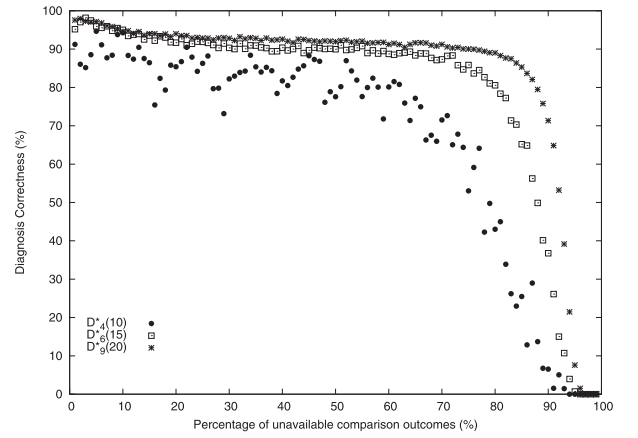


Fig. 10. Random partial syndromes.

5.2 Performance Using Partial Syndromes

Diagnosing large systems using the comparison approach presents many challenges. To date, all proposed diagnosis algorithms assumed that all comparison outcomes are available prior to initiating the diagnosis phase. This could be time consuming as waiting for all comparison tests to be conducted may take long period of time affecting hence the diagnosis latency. We believe that the diagnosis latency needs to be reduced as much as possible in order to improve the dependability of the system, and hence, provide online diagnosis. The neural-network-based approach could be considered as an online system-level diagnosis such the work presented by Blough and Brown in [11]. Note that Blough and Brown presented a distributed diagnosis approach, while the neural-network-based approach is a centralized one. By exploiting the offline learning phase of neural networks our approach was able to speed up the diagnosis. Moreover, when the system is working correctly the diagnosis algorithm can continue learning how to diagnose new fault situations, and as it learns it improves its efficiency as described in Appendix A, available in the online supplementary material. This constitutes the first main contribution of the BPNN-based diagnosis approach compared to exiting diagnosis algorithms. The second main contribution of the BPNN-based diagnosis approach compared to exiting diagnosis algorithms is that it works with partial syndromes. In [11], Blough and Brown have shown that partial syndromes occur in real systems as faults may come up during the execution of the diagnosis algorithm. Hence, the syndrome is incomplete (partial) without reflecting the new faults. The new diagnosis approach does not rely fully on the whole syndrome to diagnose a fault situation, and hence, it can provide a certain level of correct diagnosis when exposed to partial syndromes.

In the following, we describe the performance results when GCM-based partial syndromes are used. Similar results can be obtained for SCM-based systems.

5.2.1 Case 1: Random Partial Syndromes

Fig. 10 shows the results of the first type of experiments where we have randomly omitted comparison outcomes. As one can easily deduce from the figure, the BPNN-based diagnosis algorithm was able to provide a certain level of correct diagnosis even though many comparison outcomes were missing. Of course, we were expecting a degradation in diagnosis correctness, but we believe that the partial

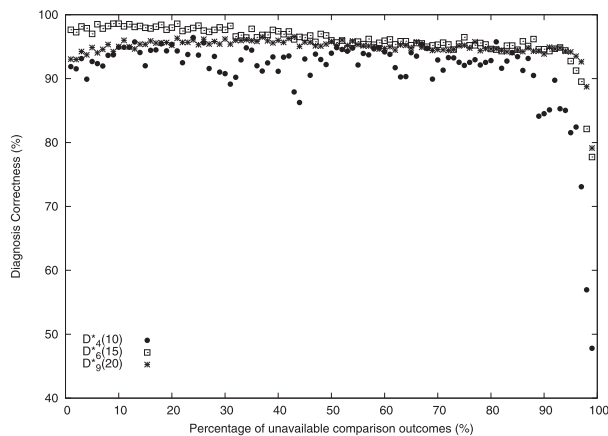


Fig. 11. Randomly omitting one node's comparisons.

diagnosis results provided will be useful in situations where not all comparison outcomes are, or until they become, available. To date, none of the existing diagnosis algorithms can work with partial syndromes.

5.2.2 Case 2: Omitting One Node's Comparison Outcomes

In a second type of experiments, we have selected randomly one node and then randomly omitted comparison outcomes that involved this node either as a comparator or as a compared node. Fig. 11 describes the obtained results. The diagnosis algorithm saw a small degradation in performance that was expectable. We have also noticed that when the selected node was part of a large number of comparison tests, its impact on the diagnosis correctness is considerable especially for small systems.

5.2.3 Case 3: Omitting One Group of Nodes' Comparison Outcomes

Our third experiment aimed at checking the impact of missing the comparison outcomes of a group of nodes. The experiment consisted in generating partial syndromes by randomly selecting a group of nodes and then randomly omitting a certain percentage of the comparison outcomes that involved them either as comparators or as compared nodes. For example, consider two nodes (20 percent of the number of nodes) randomly selected from a $D_4^*(10)$ system. If 20 percent of the comparison outcomes involving these two nodes are missing then the diagnosis correctness is around 89 percent, and it is almost 96 percent for a $D_6^*(15)$ system. Fig. 12 describes the obtained results. We can notice that the diagnosis algorithm continues providing a certain level of correct diagnosis even when large number of comparison outcomes are missing.

6 CONCLUSIONS

The backpropagation neural network-based diagnosis approach presented in this paper aims at solving the well-known system-level diagnosis problem using comparison models such the simple comparison model and the generalized comparison model, and partial syndromes. The proposed approach exploits the offline learning phase of neural networks to speed up the diagnosis algorithm and uses a postprocessing phase to escape local optima. The results from an extensive simulation study have shown that

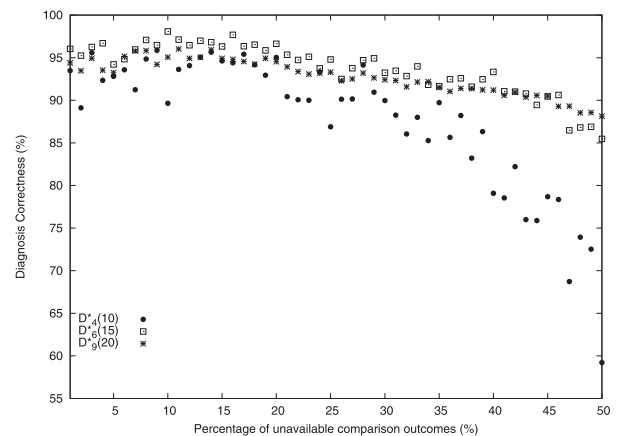


Fig. 12. Omitting a group of nodes' comparisons.

the efficiency of this novel approach in detecting all fault situations. We have shown also that the neural network approach was able to provide good results when partial syndromes and nondiagnosable systems are considered.

Future investigations will be twofold. First, we plan to perform an extensive performance evaluation whose objective is to compare all developed system-level diagnosis algorithms to the neural network approach. Second, we are in the process of applying the BPNN-based diagnosis to other diagnosis models, such as the PMC model [4] and the broadcast comparison model [11]. We also believe that given the features of the neural network diagnosis approach, a natural extension would be to apply this new approach to probabilistic models [2], and to hybrid faults as described by Kozłowski and Krawczyk in [37]. It would be also interesting to experiment and analyze the use of alternative mechanisms, such as Hopfield networks and Support vector machines [38], for solving the system-level diagnosis problem.

REFERENCES

- [1] M. Barborak, M. Malek, and A. Dahbura, "The Consensus Problem in Fault-Tolerant Computing," *ACM Computing Surveys*, vol. 25, no. 2, pp. 171-220, June 1993.
- [2] S. Lee and K. Shin, "Probabilistic Diagnosis of Multiprocessor Systems," *ACM Computing Surveys*, vol. 26, no. 1, pp. 121-139, Mar. 1994.
- [3] E.P. Duarte Jr, R.P. Ziwich, and L.C. Albin, "A Survey of Comparison-Based System-Level Diagnosis," *ACM Computing Surveys*, vol. 43, no. 22, Apr. 2011.
- [4] F. Preparata, G. Metze, and R. Chien, "On the Connection Assignment of Diagnosable Systems," *IEEE Trans. Electronic Computer*, vol. EC-16, no. 6, pp. 848-854, Dec. 1967.
- [5] F. Barsi, F. Grandoni, and P. Maestrini, "A Theory of Diagnosability without Repairs," *IEEE Trans. Computers*, vol. C-25, no. 6, pp. 585-593, June 1976.
- [6] S. Kreutzer and S. Hakimi, "Adaptive Fault Identification in Two New Diagnostic Models," *Proc. 21st Allerton Conf. Comm., Control and Computing*, pp. 353-362, 1983.
- [7] M. Malek, "A Comparison Connection Assignment for Diagnosis of Multiprocessor Systems," *Proc. Seventh Int'l Symp. Computer Architecture*, pp. 31-35, 1980.
- [8] K. Chwa and S. Hakimi, "Schemes for Fault Tolerant Computing: A Comparison of Modularly Redundant and t -Diagnosable Systems," *Information and Control*, vol. 49, pp. 212-238, June 1981.
- [9] J. Maeng and M. Malek, "A Comparison Connection Assignment for Self-Diagnosis of Multiprocessor Systems," *Proc. 11th Int'l Symp. Fault-Tolerant Computing*, pp. 173-175, 1981.
- [10] A. Sengupta and A. Dahbura, "On Self-Diagnosable Multiprocessor Systems: Diagnosis by the Comparison Approach," *IEEE Trans. Computers*, vol. 41, no. 11, pp. 1386-1395, Nov. 1992.

- [11] D. Blough and H. Brown, "The Broadcast Comparison Model for On-Line Fault Diagnosis in Multiprocessor Systems: Theory and Implementation," *IEEE Trans. Computers*, vol. 48, no. 5, pp. 470-493, May 1999.
- [12] S. Chessa and P. Santi, "Comparison-Based System-Level Fault Diagnosis in Ad Hoc Networks," *Proc. IEEE 20th Symp. Reliable Distributed Systems*, pp. 257-266, 2001.
- [13] L.C. Albin, E.P. Duarte Jr, and R.P. Ziwich, "A Generalized Model for Distributed Comparison-Based System-Level Diagnosis," *J. Brazilian Computer Soc.*, vol. 10, no. 3, pp. 44-56, 2005.
- [14] D. Blough and A. Pelc, "Complexity of Fault Diagnosis in Comparison Models," *IEEE Trans. Computers*, vol. 41, no. 3, pp. 318-324, Mar. 1992.
- [15] M. Elhadeif, "A Perceptron Neural Network for Asymmetric Comparison-Based System-Level Fault Diagnosis," *Proc. Fifth Int'l Conf. Availability, Reliability and Security (ARES '09)*, Mar. 2009.
- [16] A. Hassanien, A. Abraham, J. Peters, G. Schaefer, and C. Henry, "Rough Sets and Near Sets in Medical Imaging: A Review," *IEEE Trans. Information Technology in Biomedicine*, vol. 13, no. 6, pp. 955-968, Nov. 2009.
- [17] M. Meireles, P. Almeida, and M. Simoes, "A Comprehensive Review for Industrial Applicability of Artificial Neural Networks," *IEEE Trans. Industrial Electronics*, vol. 50, no. 3, pp. 585-601, June 2003.
- [18] A.-P.N. Refenes, A. Burgess, and Y. Bentz, "Neural Networks in Financial Engineering: A Study in Methodology," *IEEE Trans. Neural Networks*, vol. 8, no. 6, pp. 1222-1267, Nov. 1997.
- [19] S. Mitra, S. Pal, and P. Mitra, "Data Mining in Soft Computing Framework: A Survey," *IEEE Trans. Neural Networks*, vol. 13, no. 1, pp. 3-14, Jan. 2002.
- [20] J. Korbicz, J.M. Koscielny, Z. Kowalczyk, and W. Cholewa, *Fault Diagnosis: Models, Artificial Intelligence, Applications*. Springer, 2004.
- [21] J. He, Z. Zhou, X. Yin, and S. Chen, "Using Neural Networks for Fault Diagnosis," *Proc. IEEE-INNS-ENNS Int'l Joint Conf. Neural Networks (IJCNN '00)*, Oct. 2000.
- [22] X. Yang and Y.Y. Tang, "Efficient Fault Identification of Diagnosable Systems under the Comparison Model," *IEEE Trans. Computers*, vol. 56, no. 12, pp. 1612-1618, Dec. 2007.
- [23] K. Abrougui and M. Elhadeif, "Parallel Self-Diagnosis of Large Multiprocessor Systems under the Generalized Comparison Model," *Proc. 11th Int'l Conf. Parallel and Distributed Systems*, pp. 78-84, July 2005.
- [24] H. Wang, D. Blough, and L. Alkalaj, "Analysis and Experimental Evaluation of Comparison-Based System-Level Diagnosis of Multiprocessor Systems," *Proc. 24th Int'l Symp. Fault-Tolerant Computing*, pp. 55-64, 1994.
- [25] M. Elhadeif and A. Nayak, "Efficient Symmetric Comparison-Based Self-Diagnosis Using Backpropagation Artificial Neural Networks," *Proc. IEEE 28th Int'l Performance Computing and Comm. Conf. (IPCCC '10)*, pp. 264-271, Dec. 2010.
- [26] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic Concepts and Taxonomy of Dependable and Secure Computing," *IEEE Trans. Dependable Secure Computing*, vol. 1, no. 1, pp. 11-33, Jan.-Mar. 2004.
- [27] A. Pelc, "Undirected Graph Models for System Level Fault Diagnosis," *IEEE Trans. Computers*, vol. 40, no. 11, pp. 1271-1276, Nov. 1991.
- [28] E. Ammann and M. DalCin, "Efficient Algorithms for Comparison-Based Self-Diagnosis," *Proc. Self-Diagnosis and Fault-Tolerance*, pp. 1-18, 1981.
- [29] M. Elhadeif and B. Ayeb, "Efficient Comparison-Based Fault Diagnosis of Multiprocessor Systems Using an Evolutionary Approach," *Proc. 15th Int'l Parallel and Distributed Processing Symp. (IPDPS '01)*, Apr. 2001.
- [30] M. Elhadeif, S. Das, and A. Nayak, "System-Level Fault Diagnosis Using Comparison Models: An Artificial-Immune-Systems-Based Approach," *J. Networks*, vol. 1, no. 5, pp. 43-53, Nov. 2007.
- [31] M. Elhadeif, A. Nayak, and N. Zeng, "An Ant-Based Fault Identification Algorithm for Distributed and Parallel Systems," *Proc. 10th World Conf. Integrated Design and Process Technology Conf. (IDPT '07)*, June 2007.
- [32] M. Elhadeif, A. Boukerche, and H. Elkadiki, "A Distributed Fault Identification Protocol for Mobile Ad-Hoc and Wireless Mesh Networks," *J. Parallel and Distributed Computing*, vol. 68, no. 3, pp. 321-335, Mar. 2008.
- [33] A. Dahbura and G. Masson, "An $O(n^{2.5})$ Fault Identification Algorithm for Diagnosable Systems," *IEEE Trans. Computer*, vol. C-33, no. 6, pp. 486-492, June 1984.
- [34] I. Stewart, "A General Algorithm for Detecting Faults Under the Comparison Diagnosis Model," *Proc. 24th Int'l Parallel and Distributed Processing Symp. (IPDPS '10)*, pp. 19-23, Apr. 2010.
- [35] D.W. Patterson, *Artificial Neural Networks: Theory and Applications*. Prentice Hall, 1996.
- [36] "Neural Network Warehouse: AI Depot," <http://neuralnetworks.ai-depot.com/>, 2011.
- [37] W.E. Kozlowski and H. Krawczyk, "A Comparison-Based Approach to Multicomputer System Diagnosis in Hybrid Fault Situations," *IEEE Trans. Computers*, vol. 40, no. 11, pp. 1283-1287, Nov. 1991.
- [38] J. Shawe-Taylor and N. Cristianini, *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*. Cambridge Univ. Press, 2000.



Mourad Elhadeif received the BSc and MSc degrees in computer science from the Institut Supérieur de Gestion, Tunis, Tunisia, and the PhD degree in computer science from the university of Sherbrooke, Québec, Canada. Currently, he is working as an associate professor at the College of Engineering and Computer Science, Abu Dhabi University. Prior to this, he held a faculty position at the University of Ottawa, Ontario, Canada. His research interests include fault tolerance and fault diagnosis in distributed, wireless and ad hoc networks, fault tolerance of optical networks, artificial intelligence, swarm intelligence, artificial immune systems, and genetic algorithms.



Amiya Nayak received the BMath degree in computer science and combinatorics and optimization from the University of Waterloo, in 1981 and the PhD in systems and computer engineering from Carleton University, in 1991. He has more than 17 years of industrial experience in software engineering, avionics and navigation systems, simulation and system-level performance analysis. He is in the editorial board of several journals, including *IEEE Transactions on Parallel and Distributed Systems*, *International Journal of Parallel, Emergent and Distributed Systems*, *International Journal of Computers and Applications*, and *EURASIP Journal of Wireless Communications and Networking*. Currently, he is working as a full professor at the School of Electrical Engineering and Computer Science at the University of Ottawa. His research interests include the area of fault tolerance, distributed systems/algorithms, and mobile ad hoc networks with more than 150 publications in refereed journals and conference proceedings.

► For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/publications/dlib.