A Binary Particle Swarm Optimization Approach to Fault Diagnosis in Parallel and Distributed Systems

Rafael Falcon, *Graduate Student Member, IEEE*, Marcio Almeida and Amiya Nayak, *Senior Member, IEEE*

***Abstract*— The efficient diagnosis of hardware and software faults in parallel and distributed systems remains a challenge in today's most prolific decentralized environments. System-level fault diagnosis is concerned with the identification of all faulty components among a set of hundreds (or even thousands) of interconnected units, usually by thoroughly examining a collection of test outcomes carried out by the nodes under a specific test model. This task has non-polynomial complexity and can be posed as a combinatorial optimization problem. Here, we apply a binary version of the Particle Swarm Optimization meta-heuristic approach to solve the system-level fault diagnosis problem (BPSO-FD) under the invalidation and comparison diagnosis models. Our method is computationally simpler than those already published in literature and, according to our empirical results, BPSO-FD quickly and reliably identifies the true ensemble of faulty units and scales well for large parallel and distributed systems.**

I. Introduction

Distributed and parallel systems continue to increasingly permeate societies nowadays. From cellular networks to distributed database management systems, the emergence of innovative architectural and communication protocols has given rise to the next generation of decentralized systems such as wireless sensor and robot networks and cloud computing [17]. On the other hand, many groundbreaking research projects largely rest on powerful multiprocessor systems due to their unrivaled processing capabilities, rapid growth and improved affordability.

It is from the standpoint of these technological advancements that we witness a revival among the scientific community when it comes to fault tolerance protocols, as processing units in decentralized systems are subject to both hardware and software faults. Since undetected faults lead to system errors with unpredictable results, efficiently diagnosing the system's status (i.e., identifying which nodes are faulty and which are fault-free) still remains a serious challenge for committed researchers.

The aforementioned problem is known as "system-level fault diagnosis" and has drawn a significant amount of research over the last thirty years. Different test models [7] [12] [15] have been proposed in literature, each relying on the common assumption that every system unit is tested by several other units. Two well known models are the invalidation and the comparison models. They constitute classical benchmarks in the field. Both models assume that each test outcome indicates the status of the tested node and and the system's statuses are diagnosed from the results of the tests. The collection of all test outcomes is referred to as the *input syndrome* and stands as the major fault identification vehicle. Thorough examination of the input syndrome under the set of rules enforced by each test model turns out to be a non-polynomial complexity problem. Furthermore, given the discrete nature of the node assignment process (every node is labeled as either faulty or fault-free), system-level fault diagnosis can be modeled as a combinatorial optimization problem, whose (optimal) solution is the set of node assignments that entirely matches the system's input syndrome.

Although several algorithms have been put forward with this goal in mind (e.g. branch-and-bound [8] [11] [16], logical framework [1], etc.), it wasn't until a few years ago that nature-inspired meta-heuristic optimization approaches [18] were brought into the context of fault identification in distributed systems. This sort of methods have become very popular among the optimization community for their proved ability to overcome local optima through a parallel exploration of the search space and the exploitation of social communication mechanisms to drive the population toward promising search regions. Ant colonies [5], genetic algorithms (GA) [3] and artificial immune systems (AIS) [19] have all proved successful in reliably and quickly spotting the ensemble of damaged nodes within reasonable time and space bounds, even exhibiting a good performance in large-size systems.

Yet the implementation of these meta-heuristic algorithms could be rendered computationally prohibitive in many real-life scenarios given their underlying intricacy. In the former case of artificial ant colonies, the authors maintained two sets of pheromone trails and heuristic information per node, whereas GA and AIS develop their search strategy on the basis of the procreation (either by recombination or cloning) of the individuals in their population. It is clear that memory and computing power requirements can severely restrict the settings in which the previous methods will be of any use in practical terms, e.g. an ordinary sensor, acting as a cluster-head in a wireless sensor network with cluster-based topology, is responsible for processing the test outcomes reported by all sensors in its cluster.

While parallelization stands as a feasible workaround [4] to alleviate complexity issues, simpler optimization approaches will yield greater long-term benefits. We propose the application of a discrete (binary) version of the successful Particle Swarm Optimization (PSO) meta-heuristic algorithm to solve the system-level fault diagnosis problem. In our

All authors are affiliated with the School of Information Technology and Engineering, University of Ottawa, 800 King Edward Ave., Ottawa ON, Canada K1N 6N5 (phone: +1 613 562 5800 x2141; email: {rfalc032, malmeida, anayak}@site.uottawa.ca).

978-1-4244-8126-2/10/$26.00

approach, each particle encodes a possible set of faulty units and moves for the search space trying to find the possible set of faulty units that matches the collection of test outcomes contained in the input syndrome. Despite being PSO a population-based optimization method [9], it has the advantage that the number of individuals remains constant throughout the algorithm's execution (as opposed to evolutionary computation representatives) and each search agent only requires a minimal amount of storage (contrary to the ant colony systems in which environment-mediated communication is done in the form of pheromone deposit for each edge/node of the problem graph). Being this so, the simplicity of our PSO-based algorithm is an appealing feature which opens the door to its applicability in resource-constrained environments. Moreover, our method was tested under two widespread diagnosis models, exhibiting a fast convergence to the actual set of faulty nodes and good scalability properties.

The manuscript has been shaped as follows. Section II is related work while Section III elaborates on the two most commonly used fault diagnosis models. The fundamentals of the original PSO formulation for the continuous case and its adaptation to the discrete world are the subject of Section IV. Then we dissect our binary PSO algorithm (Section V), provide simulation results (Section VI) and outline final remarks (Section VII).

II. Related Work

The extent to which a system can proactively react to underlying hardware or software faults depends on how fast those anomalies can be detected. No wonder then that convergence time in fault identification is a highly sought-after goal for any competitive diagnosis method. The explained methods in this section have the same goal: to find a set of faulty nodes from the input syndrome.

For example, authors in [8] and [11] put forward branch-and-bound-based heuristics whose time complexity is $O(n^3)$, where n is the system size (number of units). Sullivan [16] introduced a backtracking-based enhancement to [8] which caused the time complexity to drop to $O(t^3 + |E|)$, where t is the number of tolerated faults and $|E|$ the total number of tests carried out in the system. This bound becomes more relevant as fewer units are found defective among a large group of nodes. In practical scenarios, however, this consideration does not always hold. In [1], Ayeb presented a logical-framework-based fault diagnosis algorithm that runs in $O(n^2\sqrt{t}/\sqrt{\log n})$. The problem with all these approaches is their tendency to get stuck in suboptimal solutions (sets of faulty nodes), as the examination of the search space tends to be not comprehensive. This behavior leads to incorrect guesses on the nodes' status, which impacts the system's overall ability to recover successfully from potential failures caused by undiagnosed faults.

As stated in Section I, evolutionary and heuristic algorithms were applied to system-level fault diagnosis by Elhadef et al [3] [19], in particular genetic algorithms (GA) and artificial immune systems (AIS). Both methods suffer from a lack of adequate population diversity given the biased exploration brought about by an adaptive mutation operator (openly criticized in [19] but finally adopted). Mutating genes according to their fitness values not only narrows the search capabilities of the approaches, but in this context can also be deceptive, since a gene with high fitness value doesn't always correspond to a correctly guessed node, as explained in Section V later on. More important, the worst-case identification of faulty nodes turns severely hindered in large systems composed of hundreds or thousands of nodes. This is due to the enormous number of individuals generated across the algorithm lifetime in an attempt to find the optimal solution, which consequently triggered the need for a parallel version [4]. Such demeanor is made even worst in the AIS-based model [19], where the number of elitist (fittest) antibodies and the number of clones per elitist antibody are both set to be the population size. These are no longer concerns in our binary PSO implementation given that the number of individuals is never altered and an unbiased exploration takes place, i.e. all components of the binary vector representing the candidate solution are allowed to change.

More recently, an algorithm inspired on ant colony optimization (ACO) [18] was devised for system-level fault diagnosis [5] and tester under two test models. The chief idea is to have every ant construct a full tour of the graph and label each node as faulty/fault-free as it goes by. Unlike widely recognized ACO models, authors don't use neither pheromone nor heuristic values to perform internodal transitions but to guess the node's status. The drawback here lies in the redundant storage of dual sets of pheromone trails and heuristic information per node for each ant journey, which aggravates the memory consumption issue in very large systems. Furthermore, it is not clear what is the heuristic value assigned to a node by a "flying ant", as no link between the current and "leaping-to" nodes might exist. Another downside of their meta-heuristic implementation is the handling of infeasible solutions, which are discarded by default, thus wasting valuable information gathered during the ant's tour along the graph. Our PSO-derived approach requires less information at the search agent level (particle), gracefully turns infeasible into feasible candidate solutions and is theoretically simpler than its ant-based peer.

III. Fault Diagnosis Models

Several models to identify faults in distributed systems appear in literature. In this section, we will focus on the popular *invalidation* and *comparison* models. Both of them assume that each entity is capable of testing a particular subset of the other entities in the system and every test has a binary character. A test outcome indicates whether the node is either *faulty* (1) or *fault-free* (0). Moreover, the group of all faulty units in a system is called the *fault set*. One attribute of the so-called *t-diagnosable* systems is that they can only have at most t faulty nodes.

Although both diagnosis models represent the problem in a graph-like fashion and use the collection of test outcomes

as a means to figure out what the status of every node is, yet the graph representation and implicit assumptions on the test results are model-specific.

A. The PMC Model

In the invalidation or PMC model (named after its authors in [15]), the problem is represented as a directed graph $G = (V, E)$ with $V = \{v_1, v_2, \ldots, v_n\}$ being the set of entities (processors, sensors, etc.) and E the set of edges. Each edge $e_{ij} \in E$ stands for a test performed by node v_i upon node v_j and whose binary outcome is denoted as σ_{ij}. The set of all test outcomes is called a *syndrome* and is symbolized by σ. Since a syndrome σ is a physical manifestation of an underlying fault set F, we usually write σ_F.

According to the PMC model, the system has an stationary nature, i.e. the statuses of all nodes do not change during the diagnosis phase. It is also assumed that tests carried out by fault-free nodes are always correct while those executed by faulty devices are unreliable, i.e. yield arbitrary results.

Let $v_i \in V$ be any node in G. Then the set of nodes tested by v_i is $\Gamma(v_i) = \{v_j \in V : e_{ij} \in E\}$ and $\sigma(v_i) = \{\sigma_{ij} \in \sigma : j \in \Gamma(v_i)\}$ is the subset of the syndrome σ containing the results of the tests realized by v_i. In a similar way, $\Gamma^{-1}(v_i) = \{v_j \in V : e_{ji} \in E\}$ is the set of v_i's testers and $\sigma^{-1}(v_i) = \{\sigma_{ji} \in \sigma : j \in \Gamma^{-1}(v_i)\}$ the outcomes of all tests carried out upon v_i.

Definition III.1 A fault set $F \subseteq V$ is *consistent* with the syndrome σ under the PMC model if $\forall\ e_{ij} \in E$, neither $\sigma_{ij} = 0$ where $v_i \in V - F$ and $v_j \in F$, nor $\sigma_{ij} = 1$ where $v_i, v_j \in V - F$, holds.

The above definition states that only fault-free nodes always give correct results. This assumption will play a pivotal role later on during the generation of candidate syndromes as part of the quest for the true fault set.

B. Comparison Model

In the comparison model [14], nodes perform tests in a pairwise fashion, i.e. some pairs of units test each other. The comparison between the two test outcomes (match/mismatch) stands as the foundation for deriving the nodes' statuses. More formally, in a comparison-based scenario, an undirected graph $G = (V, E)$ is used to model the system, where V is the set of units and E the collection of edges. Now each edge e_{ij} has an associated weight $\sigma_{ij} = 0$ when the two test results carried out by units v_i and v_j are identical and $\sigma_{ij} = 1$ otherwise. As in the PMC model, the collection of all edge weights (test comparisons) creates the syndrome σ.

The comparison model also regards the system as static, likewise PMC. Now the main assumption is that two fault-free units yield identical test outcomes whereas any couple of faulty and fault-free nodes will yield mismatching results. However divergent standpoints arise when it comes to the test comparison of two faulty nodes. The asymmetric version of the comparison model [12] considers that two damaged nodes will always produce a disagreement while in the symmetric version [7], both a match and a mismatch are likely choices.

Let $v_i \in V$ be any node in G. Then the set of neighbors of v_i is $\Gamma(v_i) = \{v_j \in V : e_{ij} \in E\}$ and $\sigma(v_i) = \{\sigma_{ij} \in \sigma : j \in \Gamma(v_i)\}$ is the subset of the syndrome σ containing the test agreements/disagreements involving v_i.

Definition III.2 A fault set $F \subseteq V$ is *consistent* with the syndrome σ under the comparison model if $\forall\ e_{ij} \in E$, neither $\sigma_{ij} = 0$ where $v_i \in V - F$ and $v_j \in F$ (or $v_i \in F$ and $v_j \in V - F$), nor $\sigma_{ij} = 1$ where $v_i, v_j \in V - F$, holds.

C. More on t-Diagnosable Systems

From the previous two subsections one may realize that, given an input syndrome σ which is compliant with the specifications of some model, multiple faults sets could possibly give rise to it. This makes system-level fault diagnosis a nondeterministic problem, which is of course very undesirable since the actual set of faulty units cannot be guessed with full certainty. To prevent this, we introduce the following definition.

Definition III.3 A system of n units is *t-diagnosable* if and only if the number of faulty units doesn't exceed t and for any consistent syndrome σ, there is only one fault set F that originated it.

In the sequel, we confine ourselves to a type of system which is easy to generate and known to be t-diagnosable. As such, it will be used in our experiments.

Definition III.4 A system represented by a test graph $G = (V, E)$ with $n = |V|$ is a $G_t(n)$ design iff i) $n \geq 2t + 1$ and ii) each node is tested by t others.

For the PMC model, it was proved in [6] that $G_t(n)$ system in which no two units test each other is $t-$diagnosable. As to the comparison model, recall that we consider a single link between any two entities, so the $G_t(n)$ design fits very well once it was demonstrated to be t-diagnosable for this model too [14]. Figures 1 and 2 display a $G_t(n)$ graph under the PMC and symmetric comparison models, respectively.

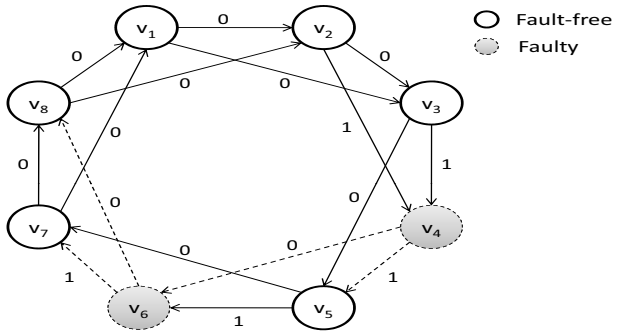


Fig. 1. A $G_2(8)$ test assignment graph under the PMC model. The directed edge labels represent test outcomes.

Finally, let us point out that, irrespective of the diagnosis model being used, the problem of fault detection in distributed and parallel systems can be posed as a combinatorial optimization problem if we undertake a quest, over the discrete space of all possible fault sets F, of the actual fault set $\overline{F}$ which univocally generates the known input syndrome

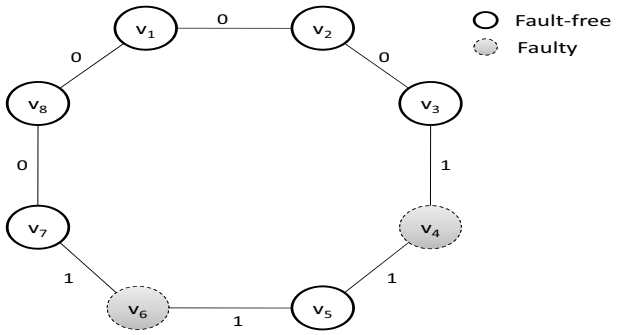


Fig. 2. A $G_2(8)$ graph under the symmetric comparison model. The undirected edge labels represent the comparison result of pairwise test outcomes.

$\sigma_{\overline{F}}$. This task has non-polynomial time complexity [21] and calls for the application of heuristic methods to informedly navigate over the search space until $\overline{F}$ is found. The next section is devoted to outline the fundamentals of one of such meta-heuristic procedures.

IV. Particle Swarm Optimization

One of the most widespread swarm intelligence algorithms [18] is Particle Swarm Optimization (PSO), which exploits the social behavior of some animal communities like bird flocks or fish schools in the pursuit of desirable locations in a given area, e.g. birds flocking to a food source. The inherently collaborative nature of these biological swarm systems led to the design of an optimization procedure for efficiently exploring non-linear, non-differential and multi-modal, real-valued search spaces. Originally introduced by Kennedy and Eberhart in 1995 [9], PSO has earned a well-deserved renown among population-based meta-heuristic approaches for its intuitive description, algorithmic simplicity and proven competence to reach the global optimum in a wide array of challenging optimization landscapes. As such, it has become one of the most active research areas in bio-inspired optimization models.

In PSO, a swarm S of search agents (particles) concurrently moves throughout a continuous search space in its quest for the global optimum of any multidimensional function. The swarm size is n and each particle represents a candidate solution to the optimization problem. The i-th particle is defined by a position vector $\vec{x_i} = (x_{i1}, x_{i2}, \ldots, x_{id})$ where d is the dimensionality of the search space and a velocity vector $\vec{v_i} = (v_{i1}, v_{i2}, \ldots, v_{id})$. The position of every agent is evaluated according to a problem-dependent fitness function $f : \Re^d \rightarrow \Re$ and each particle remembers its best-ever-found position $\vec{p_i} = (p_{i1}, p_{i2}, \ldots, p_{id})$. This is the *cognitive* component of PSO, often referred to as the particle's "local best".

Additionally, a neighborhood is defined for every particle as the subset of individuals which it is able to communicate with. The first PSO model used an Euclidean neighborhood by measuring the actual distance among particles (as an imitation of the biological models) to determine the nearest ones. This formulation was indeed computationally-intensive and became eventually replaced with topological neighborhoods, which do not regard the particles' vicinity. Among them, the so-called "global-best" neighborhood allows full communication among all swarm individuals. A good analysis of "local-best" versus "global-best" neighborhood definitions can be found in [2]. The particle's view of its neighborhood determines the "social" component of PSO.

Particles fly along the search space attracted by their best individual solution ("local best") and the best solution $\vec{p_g}$ found by any particle in their neighborhood ("global best"). The entire swarm is updated at each time step by modifying the velocity and position of each particle in every dimension according to the following rules:

$$v_{ij} = \chi \left(v_{ij} + c_1 \epsilon_1 (p_{ij} - x_{ij}) + c_2 \epsilon_2 (p_{gj} - x_{ij}) \right) \quad (1)$$

$$x_{ij} = x_{ij} + v_{ij} \quad (2)$$

where c_1, c_2 are called "acceleration constants", ϵ_1 and ϵ_2 are independent random numbers uniquely generated at every update for each dimension $j \in \{1 \ldots d\}$, χ is the "constriction factor" meant to balance the global and local exploration of the algorithm. This parameter plays a fundamental role in the algorithm achieving good convergence capabilities and can be computed as shown in (3). Most implementations adjust the values of c_1 and c_2 to 2.05 to obtain $\varphi > 4$ to guarantee the convergence, because when $\varphi < 4$ the swarm would "spiral" toward and around the best solution found in the search space [2].

$$\chi = \frac{2}{|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}|}, \varphi = c_1 + c_2 \quad (3)$$

The algorithm terminates usually after a certain number of iterations. Notice that PSO doesn't generate new individuals unlike evolutionary approaches but dynamically modifies the swarm members, which is a remarkable feature that saves time and memory when dealing with overly complex systems.

A. Binary PSO

Though at first conceived for coping with continuous optimization problems, PSO has undergone many further developments since its early inception that make it now fit for finding the most promising combination of discrete vector components that optimizes a given scenario. The first PSO plunge into the realm of combinatorial optimization took place shortly after its birth when a discrete or binary version of the algorithm was put forward by its authors [10].

In discrete PSO, each particle is encoded as a binary vector of length d and the position update equation in (2) is redefined as a probabilistic rule based on a sigmoidal transformation applied to the real-valued particle velocity.

$$x_{ij} = \begin{cases} 1, & \text{if } r \leq \frac{1}{1+e^{-v_{ij}}} \\ 0, & \text{otherwise} \end{cases} \quad (4)$$

where $r \sim U(0,1)$. Expression (4) forces the particle's components to be binary values while still allows the swarm members to benefit from the cognitive and social components of the original PSO formulation for the velocity calculation in a continuous search space. The simplicity behind this idea is one of the major appeals of the algorithm, which in turn has become the focus of more intensive research for speeding up convergence.

V. BPSO-FD: The Proposed Approach

We model system-level fault diagnosis as a combinatorial search problem undertaken by a binary PSO approach. The goal is to find, among many possible fault sets (see Section III), the true set $\overline{F}$ that entirely matches the collection of test outcomes contained in the input syndrome $\sigma_{\overline{F}}$. In our approach the diagnosis process needs to run in a separate node.

A. Particle Encoding and Initialization

In BPSO-FD, each particle stands for a candidate fault set F. Therefore, we encode its position $\vec{x_i}$ as a binary vector whose dimensionality $d = |V|$ is given by the system size (number of distributed or parallel components in the system). The status of the j-th node ($j \in \{1..d\}$) will be denoted by $x_{ij} = 1$ if we guess that v_j is faulty or $x_{ij} = 0$ otherwise.

At the algorithm outset, all particle velocities are randomly initialized with real values in $[0,1]$ whereas for each particle position, it is first decided at random the number of faulty units *nf* $\sim U(1,t)$ the particle will encode and then *nf* bits will be arbitrarily set to one. By doing so, we are trying to promote diversity in the initial swarm since the cardinality of the initial fault sets will vary stochastically within the feasible bounds imposed by a t-diagnosable system.

For the potential fault set F encoded by each particle, we generate its corresponding candidate syndrome σ_F as a previous step to compute the particle's fitness.

B. Fitness Function

The quality of any particle (fault set F) in our problem can be measured as the resemblance between the input syndrome $\sigma_{\overline{F}}$ and the candidate syndrome σ_F. The computation of the latter one and of the fitness function itself is strongly model-dependent although they share some similarities.

1) Candidate Syndrome Generation: Out of all diagnosis models considered in this study, only the asymmetric comparison model is entirely deterministic. That is, given the assumption of which nodes are faulty and which are fault-free (particle encoding), we follow the asymmetric model rules stated in Section III-B and assign a label to every edge e_{ij} in the test assignment graph G. The collection of these labeled edges becomes our candidate syndrome σ_F.

Since the invalidation and symmetric comparison models have a non-deterministic nature concerning the tests carried out by faulty nodes (in the former case) and the agreement/disagreement between a pair of mutually-tested damaged devices (in the latter case), one can make the reasonable assumption that these edges in the candidate syndrome are equal to those in the input syndrome, i.e.

$$\sigma_F(v_i) = \sigma_{\overline{F}}(v_i) \quad \forall\, v_i \in F$$

Then we generate the remaining edge labels in σ_F in full compliance with the rules defined for each model, which were clearly portrayed in Section III-B.

2) The Particle Fitness: A particle is said to have better quality (fitness) than others if the candidate syndrome associated with its encoding (fault set F) resembles the input syndrome better. From the local viewpoint of node v_i, this is equivalent to count how many labels of the incoming/outgoing edges coincide in both syndromes. For the comparison models, expression (5) models the node-level similarity.

$$f\left(\sigma_F, \sigma_{\overline{F}}, v_i\right) = \frac{|\sigma_F(v_i) \cap \sigma_{\overline{F}}(v_i)|}{|\Gamma(v_i)|} \tag{5}$$

For the PMC model, we are to take into account that all nodes will be tested but not necessarily testers. Equations (6) to (8) model the similarity from v_i's viewpoint as a tester node, tested node and both viewpoints, respectively.

$$f^{+1}\left(\sigma_F, \sigma_{\overline{F}}, v_i\right) = \begin{cases} 1, & \text{if } |\Gamma(v_i)| = 0 \\ \frac{|\sigma_F(v_i) \cap \sigma_{\overline{F}}(v_i)|}{|\Gamma(v_i)|}, & \text{otherwise} \end{cases} \tag{6}$$

$$f^{-1}\left(\sigma_F, \sigma_{\overline{F}}, v_i\right) = \frac{|\sigma_F^{-1}(v_i) \cap \sigma_{\overline{F}}^{-1}(v_i)|}{|\Gamma^{-1}(v_i)|} \tag{7}$$

$$f\left(\sigma_F, \sigma_{\overline{F}}, v_i\right) = \frac{f^{+1}\left(\sigma_F, \sigma_{\overline{F}}, v_i\right) + f^{-1}\left(\sigma_F, \sigma_{\overline{F}}, v_i\right)}{2} \tag{8}$$

Now we can define in (9), regardless of the diagnosis model used, the overall resemblance function between the input and candidate syndromes, which in turns becomes the fitness function for BPSO-FD.

$$f(\sigma_F, \sigma_{\overline{F}}) = \frac{\sum_{v_i \in V} f\left(\sigma_F, \sigma_{\overline{F}}, v_i\right)}{|V|} \tag{9}$$

The above equation can be seen as the correctness probability of the potential fault set F being the actual fault set $\overline{F}$. Because we are working with t-diagnosable systems solely, then it is guaranteed that there is a unique fault set F that makes $\sigma_F = \sigma_{\overline{F}}$. Remark that (5) and (8) take values in $[0,1]$, which consequently defines the image of (9) to be the same interval.

C. Position Update Rule

We have used expression (10) introduced in a recent study [20] which has yielded better experimental results for our problem.

$$x_{ij} = \begin{cases} 1, & \text{if } r \leq \frac{1}{1+\, e^{-1.5 \cdot d \cdot v_{ij}}} \\ 0, & \text{otherwise} \end{cases} \tag{10}$$

where $d = |V|$ is the particle dimensionality.

D. Handling Infeasible Solutions

The application of the stochastic rule in (10) to every component of the particle's encoding can lead us to an infeasible solution, which is a certain configuration of node guesses in which either all units are fault-free or there are more than t faulty units. Both scenarios are unacceptable in t-diagnosable systems and are to be dealt with. We gracefully manage this situation in the following way:

1) Compute the node-level resemblance by either (5) or (8) depending on the underlying diagnosis model.
2) If the particle encoding is the zero vector, generate a random number $b \in [1..t]$ and flip (turn to 1) the b bits with the lowest fitness values. Otherwise, let k be the number of 1's in the particle (notice that $k > t$). Then generate a random number $b \in [k-t..k-1]$ and flip (turn to 0) the b bits set to 1 with the lowest fitness values.

The above procedure turns infeasible into feasible particles by flipping the bits with lowest local resemblance to the input syndrome, in an attempt to accelerate the convergence of the algorithm towards the actual fault set.

E. Stop Criterion

Unlike many common PSO implementations, our algorithm doesn't stop when it reaches an upper bound in the number of iterations to carry out but when it finds the actual fault set $\overline{F}$, i.e. a particle encoding F with fitness value $f(\sigma_F, \sigma_{\overline{F}}) = 1$. This fault set F is proved to be unique in t-diagnosable systems [6] [14].

F. The BPSO-FD Algorithm

Algorithm 1 unfolds the entire pseudo-code for the BPSO-FD scheme.

Algorithm 1 Binary PSO to Fault Diagnosis (BPSO-FD)

Require: input syndrome $\sigma_{\overline{F}}$, consts c_1, c_2, swarm size n
iteration $\leftarrow 0$
compute constriction factor χ using (3)
initialize swarm as shown in Section V-A
repeat
  iteration $\leftarrow$ iteration $+1$
  for each particle i in the swarm **do**
    update particle velocity $\vec{v_i}$ using (1)
    update particle position $\vec{x_i}$ using (10)
    if $\vec{x_i}$ infeasible **then**
      apply feasibility scheme in Section V-D
    end if
  end for
  for each particle i in the swarm **do**
    $F \leftarrow \vec{x_i}$; generate σ_F as shown in Section V-B.1
    compute the fitness $f(\sigma_F, \sigma_{\overline{F}})$ using (5) – (9)
    update local best $\vec{p_i}$ accordingly
  end for
  update global best $\vec{p_g}$ accordingly
until $f(\vec{p_g}, \sigma_{\overline{F}}) = 1$ or $iteraction >= 10000$
return $\vec{p_g}$

VI. Simulation Results

We have empirically tested the performance of BPSO-FD under t-diagnosable systems of various sizes for both invalidation and comparison diagnosis models. Our algorithm has been contrasted to the one in [19], which employs an artificial immune system (AIS) as the optimization approach, in terms of (1) number of individuals (candidate fault sets) probed before identifying the true fault set and (2) the CPU time needed to find $\overline{F}$. The simulations were conducted on an Intel Core2 Duo E4500 2.2GHz with 2 GB of RAM and both schemes were coded in Java using JDK 1.6 and the t-diagnosable systems tried are $G_{24}(50)$ and $G_{49}(100)$.

We used the AIS parameter configuration outlined in [19] whereas for BPSO-FD we set our parameters as follows: $n = 10$, $c_1 = c_2 = 2.05$ and implemented the particle neighborhood after the "global best" topology with the use of constriction factor for the velocity update rule.

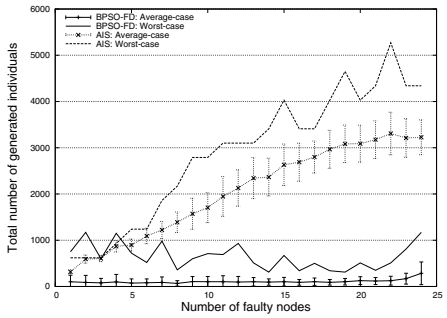
For each of the above systems, the number of faults x has been varied from 1 to t and for every value of x, we have randomly generated 100 fault sets of cardinality x and their average results and standard deviations have been reported. In all cases, both BPSO-FD and AIS succeeded in discovering the actual fault set yet they exhibit remarkable performance differences.

A. Performance under the PMC Model

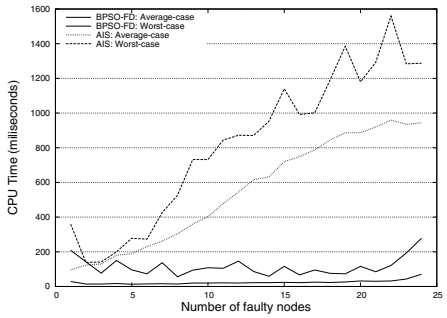
Figure 3 displays the behavior of AIS and BPSO-FD with the invalidation diagnosis model. As expected, an increase in the number of faulty nodes to be detected results in a greater amount of allocated computational resources, i.e. number of search agents (particles or antibodies) and running time. Nevertheless, we witness a nearly constant growth rate for BPSO-FD in both types of resources for the $G_{24}(50)$ graph as opposed to AIS which undergoes a polynomial growth pace (in some cases it is actually exponential).

Furthermore, observe the huge gap in the number of candidate fault sets explored by both schemes (42x wide in its peak) and execution time consumed (5.2x longer in the worst scenario) in order to arrive at the actual set $\overline{F}$. The rapid convergence of the binary PSO method to the optimum is due to the presence of the constriction factor in the velocity update equation (1) which enforces a greater exploitation of the best solutions found by the swarm and also to the use of the global best topology, which has been proved to be quite effective in unimodal optimization functions like (9). We contrast this behavior with the erratic convergence rate portrayed by AIS, whose reliance upon cloning and adaptive mutation operators leads to a poor and aimless exploration of the discrete search space.

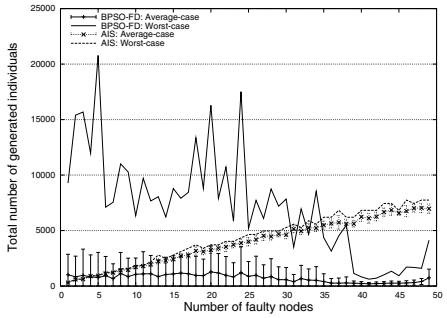
Regarding the $G_{49}(100)$ system, Figures 3(c) and 3(d) reveal that BPSO-FD behaves in a disappointing fashion for the worst case when it comes to the number of solutions explored for systems with up to 33 faults, yet from that point on it shows better performance than AIS. However it always



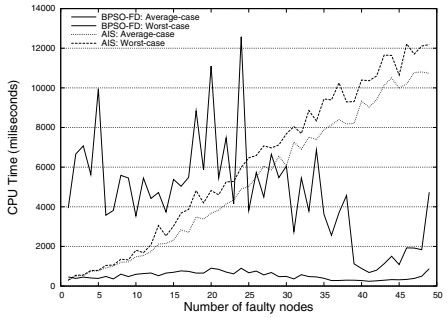
(a) Number of individuals vs. number of faulty nodes for $G_{24}(50)$



(b) CPU time vs. number of faulty nodes for $G_{24}(50)$



(c) Number of individuals vs. number of faulty nodes for $G_{49}(100)$



(d) CPU time vs. number of faulty nodes for $G_{49}(100)$

Fig. 3. Performance metrics for BPSO-FD and AIS under the PMC model.

shows itself far superior to its competitor in the average case under any fault set cardinality. It is worthwhile mentioning that AIS managed to reduce the gap between its average and worst case performance in this medium-size system.

B. Performance under the Comparison Model

The findings uncovered by Figure 4 for the asymmetric comparison model are even more encouraging. For the small system, the breach between the nature-inspired and the evolutionary representatives is more noticeable, with low values of standard deviations for the average case in the total number of individuals generated by the BPSO-FD approach. The higher degree of uncertainty among the population members in the AIS algorithm is evidenced by its fairly wide standard deviation. We witness the same behavior in terms of computational time for both approaches.

Again, the PSO-based meta-heuristic shows an irregular tendency in the worst case identification of the true fault set for the large system. However, observe that despite the non-negligible number of faults to be spotted, the performance in the average case strictly follows the quasi-constant pattern disclosed in the PMC model as well, which confirms our claim that BPSO-FD is a resilient fault diagnosis protocol for multiple test models.

VII. Conclusions and Future Work

This research study is concerned with the application of BPSO-FD, a bio-inspired combinatorial optimization algorithm, to the problem of system-level fault diagnosis under multiple test models. The convergence rate in the average case of BPSO-FD, both regarding execution time and number of solutions generated seems not to be altered regardless of the number of faults present in the system. This speaks highly of the scalability properties of our algorithm. BPSO-FD outperformed an existing artificial-immune-system-based approach in both metrics and reports a steady performance for large systems.

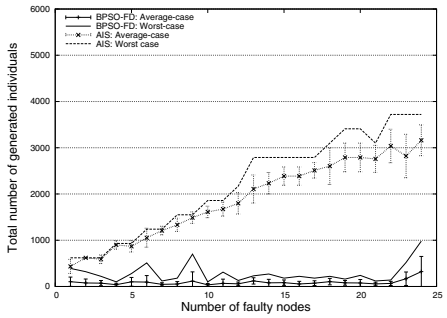
As future work we plan to incorporate other fault diagnosis models into the experimental setting and improve the worst-case behavior of the binary PSO-based scheme, possibly either through a hybridization with other population-based meta-heuristics or via the application of local search mechanisms that intensify the exploitation of the swarm-generated prospective fault sets.

References

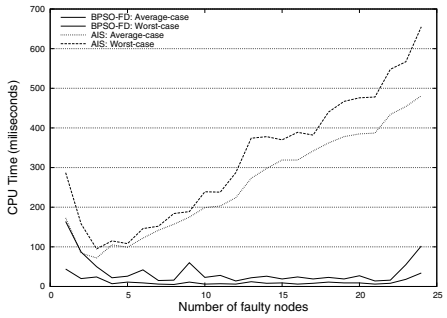
[1] B. Ayeb, "Fault Identification Algorithmic: A New Formal Approach", *Proc. 29th Int'l Symposium on Fault-Tolerant Computing*, 1999, pp. 138–145.

[2] D. Bratton and J. Kennedy, "Defining a Standard for Particle Swarm Optimization", *Proc. IEEE Swarm Intelligence Symposium SIS 2007*, Honolulu HI, USA, Apr. 2007, pp. 120–127.

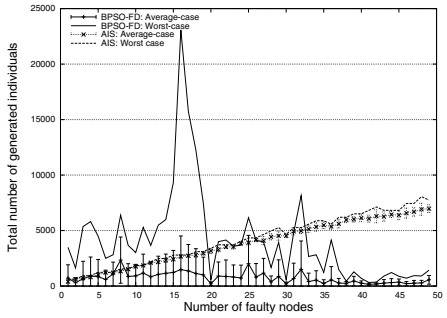
[3] M. Elhadef and B. Ayeb, "An Evolutionary Algorithm for Identifying Faults in *t*-Diagnosable Systems", *Proc. IEEE Symposium on Reliable Distributed Systems SRDS 2000*, Nürnberg, Germany, Oct. 2000, pp. 74–83.



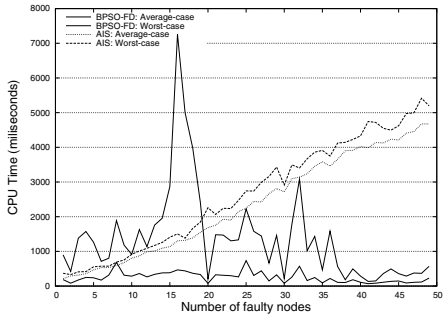
(a) Number of individuals vs. number of faulty nodes for $G_{24}(50)$



(b) CPU time vs. number of faulty nodes for $G_{24}(50)$



(c) Number of individuals vs. number of faulty nodes for $G_{49}(100)$



(d) CPU time vs. number of faulty nodes for $G_{49}(100)$

Fig. 4. Performance metrics for BPSO-FD and AIS under the asymmetric comparison model.

[4] M. Elhadef, S. Das and A. Nayak, "A Parallel Genetic Algorithm for Identifying Faults in Large Diagnosable Systems", *Int'l Journal of Parallel, Emergent and Distributed Systems*, vol. 20(2), pp. 113–125, Jun. 2005.

[5] M. Elhadef, A. Nayak and N. Zeng, "An Ant-based Fault Identification Algorithm for Distributed and Parallel Systems", *Proc. 10^{th} World Conference on Integrated Design & Process Technology IDPT-2007*, Antalya, Turkey, Jun. 2007, pp. 1–6.

[6] S. L. Hakimi and A. T. Amin, "Characterization of the Connection Assignment of Diagnosable Systems", *IEEE Trans. on Computers*, vol. C-23, pp. 86–88, Jan. 1974.

[7] S. L. Hakimi and K. Y. Chwa, "Schemes for Fault Tolerant Computing: a Comparison of Modularly Redundant and $\hat{t}$-Diagnosable Systems", *Information & Control*, vol. 49, pp. 212–238, Jun. 1981.

[8] T. Kameda, S. Toida and F.J. Allan, "A Diagnosis Algorithm for Networks", *Information & Control*, vol. 29, pp. 141–148, 1975.

[9] J. Kennedy and R. C. Eberhart, "Particle Swarm Optimization", *Proc. IEEE International Conference on Neural Networks*, Perth, Australia, 1995, pp. 1942–1948.

[10] J. Kennedy and R. C. Eberhart, "A Discrete Binary Version of the Particle Swarm Algorithm", *IEEE Conference on Systems, Man, and Cybernetics*, vol. 5 issue 12–15, pp. 4104–4108, Oct. 1997.

[11] R.F. Madden, "On Fault-Set Identification in Some System-Level Diagnostic Models", *Proc. Int'l Symposium on Fault-Tolerant Computing*, Jun. 1977.

[12] J. Maeng and M. Malek, "A Comparison Connection Assignment for Self-Diagnosis of Multiprocessor Systems", *Proc. 11^{th} International Symposium on Fault-Tolerant Computing*, New York, USA, 1981, pp. 173–175.

[13] B. T. Nassu, E. P. Duarte and A. T. Ramirez Pozo, "A Comparison of Evolutionary Algorithms for System-Level Diagnosis", *Proc. Genetic and Evolutionary Computation Conference GECCO 2005*, Washington DC, USA, Jun. 2005, pp. 2053–2060.

[14] A. Pelc, "Undirected Graph Models for System-Level Fault Diagnosis" *IEEE Trans. on Electronic Computers*, vol. 40(11), pp. 1271–1276, 1991.

[15] F. P. Preparata, G. Metze and R. T. Chien, "On the Connection Assignment of Diagnosable Systems", *IEEE Transactions on Electronic Computing*, vol. 16(6), pp. 848–854, Dec. 1967.

[16] G. Sullivan, "An $O(t^3 + |E|)$ Fault Identification Algorithm for Diagnosable Systems", *IEEE Transactions on Computers*, vol. 37(4), pp. 388–397, 1988.

[17] L. Vaquero, L. Rodero-Merino, J. Caceres and M. Lindner, "A Break in the Clouds: Towards a Cloud Definition" *ACM SIGCOMM Computer Communication Review*, vol. 39(1), pp. 50–55, Jan. 2009.

[18] X.S. Yang, *Nature-Inspired Metaheuristic Algorithms*, Luniver Press, 2008.

[19] H. Yang, M. Elhadef, A. Nayak and X. Yang, "Network Fault Diagnosis: An Artificial Immune System Approach", *Proc. 14th IEEE Int'l Conference on Parallel and Distributed Systems*, Melbourne, Australia, Dec. 2008, pp. 463–469.

[20] S. Yuan and F. Chu, "Fault Diagnostics based on Particle Swarm Optimization and Support Vector Machines" *Mechanical Systems and Signal Processing*, vol. 21, pp. 1787–1798, 2007.

[21] D.M. Blough and A. Pelc., "Complexity of Fault Diagnosis in Comparison Models", *Proc. IEEE Transactions on Computers*, vol 41(3), 1992, pp. 318–324.