System-Level Fault Diagnosis with Dynamic Mesh Optimization

Rafael Falcon[1], Marcio Almeida[1], Amiya Nayak[1] and Rafael Bello[2]

[1] School of Information Technology and Engineering (SITE), University of Ottawa, Canada

[2] Artificial Intelligence Lab, Centre on Computing Studies, Universidad Central "Marta Abreu" de Las Villas, Santa Clara, Cuba

rfalc032@site.uottawa.ca, malmeida@uottawa.ca, anayak@site.uottawa.ca, rbellop@uclv.edu.cu

Abstract. The efficient identification of hardware and software faults in parallel and distributed systems still remains a challenge in today's most prolific decentralized environments. System-level fault diagnosis is concerned with the detection of all faulty nodes in a set of hundreds (or even thousands) of interconnected units. This is accomplished by thoroughly examining the collection of outcomes of all tests carried out by the nodes under a particular test model. Such task has non-polynomial complexity and can be posed as a combinatorial optimization problem. In this paper we employ Dynamic Mesh Optimization (DMO) to detect faulty units in diagnosable systems. The proposed method encodes the potential solutions as binary vectors and exploits problem-specific knowledge to cope with infeasible individuals. The empirical analysis confirms that the DMO-based scheme outperforms existing techniques in terms of convergence speed and memory requirements, thus becoming a viable approach for real-time fault diagnosis in large-size systems.

Keywords. Fault diagnosis, input syndrome, dynamic mesh optimization, invalidation model, comparison model.

Detección de fallas en sistemas con optimización basada en mallas

Resumen. La identificación eficiente de fallas de hardware y software ensistemas paralelos y distribuidos todavía sigue siendo un desafío en loscada vez más prolíficos sistemas decentralizados de estos tiempos. Eldiagnóstico de fallas en sistemas tiene que ver con la detección de todoslos nodos defectuosos en un conjunto de cientos (o quizá miles) de unidadesinterconectadas. Esto se logra mediante un minucioso examen de la colecciónde los resultados de las verificaciones realizadas por los nodos de acuerdoa un modelo de verificación en particular. Un examen así de detallado tieneuna complejidad no polinomial y puede ser presentado como un problema deoptimización combinatoria. En este artículo se emplea la Optimización Basadaen Mallas Dinámicas (Dynamic Mesh Optimization, DMO), para detectar unidadesdefectuosas en sistemas diagnosticables. El método propuesto representa lassoluciones potenciales como vectores binarios y explota el conocimiento específico del problema para lidiar con soluciones no factibles. El análisisempírico confirma que el enfoque basado en DMO supera en rendimiento atécnicas existentes en cuanto a la velocidad de convergencia y losrequerimientos de memoria, convirtiéndose así en un enfoque viable para eldiagnóstico en tiempo real de fallas en sistemas de largo alcance.

Palabras clave. Diagnóstico de fallas; síndrome de entrada; optimización basada en mallas dinámicas; modelo de invalidación; modelo de comparación.

1 Introduction

Parallel and distributed 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 actuator networks Verdone *et al.* [25] and cloud computing Vaquero *et al*. [24]. 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 networked systems are subject to both hardware and software faults. Since undetected faults lead to system errors with unpredictable consequences, 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 [6, 17, 20] have been proposed in literature, each relying on the common assumption that every system unit is tested by one or more remaining units. Two well-known scenarios are the *invalidation* and *comparison* models. They constitute classical benchmarks in the field and assume that a test outcome indicates the correctness of the tested node and that the system's status can be entirely derived out of the *input syndrome*, which is the collection of all test results and stands as the major fault identification vehicle.

Thorough examination of the input syndrome under the set of rules enforced by a particular test model turns out to be a non-polynomial complexity task. Furthermore, given the discrete nature of the node labeling process (i.e. every node is tagged as either faulty or fault-free), system-level fault diagnosis can be modeled as a *combinatorial optimization problem*, whose optimal solution is the system labeling (set of node tags) that completely matches the known input syndrome.

Although several algorithms have been put forward with this goal in mind (e.g. branch-and-bound Kameda *et al.* [13]; R.F. Madden [16]; G.F.Sullivan [22], logical framework B. Ayeb [2], etc.), it wasn't until a few years ago that nature-inspired meta-heuristic optimization approaches Yang *et al*. [26] were brought into the context of fault diagnosis in distributed systems. This sort of methods has become very popular in 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 (ACO) [9], genetic algorithms (GA) [7], and artificial immune systems (AIS) [29] have all succeeded in reliably spotting the actual ensemble of damaged nodes, even with reasonable performance in large-scale settings.

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 ACO case, 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 their population members. It is clear that memory and computing power can severely hinder the applicability of the above techniques. For instance, consider the novel problem of *carrier-based coverage repair* [11] in a wireless sensor network with cluster-based topology. Sensors within the same cluster periodically test among themselves and report the test outcomes to their cluster head, which runs the fault diagnosis algorithm and subsequently informs a mobile robot on which are the damaged nodes in its cluster so they can be timely replaced with passive sensors. Because the cluster head is like another resource-constrained device, the fault detection algorithm must be "light" and accurate enough so that realistic conclusions are drawn with minimal memory and processing power requirements.

A recent step in this direction was undertaken in Falcon *et al.* [11] by using a discrete formulation of Particle Swarm Optimization (PSO) [14]. The authors encoded the system status as a binary vector (particle) which "flies" along the search space of all feasible solutions. Because the population size remained constant and infeasible particles were turned into feasible ones by using problem-specific hints, PSO exhibited a promising behavior in terms of memory consumption and convergence speed to the actual *fault set* (i.e., the group of all faulty units) in presence of small and medium-size systems.

We build on the success of PSO-based fault detection by applying Dynamic Mesh Optimization (DMO) [21, 33], a novel metaheuristic approach which borrows key elements from PSO and

evolutionary algorithms and proved to be effective in discrete optimization domains like the feature selection problem [3]. In our implementation, each prospective solution to the optimization problem (fault set) is represented as a *mesh node*. The collective behavior of the artificial mesh is such that topological changes, triggered by node generation towards local optima, the global optimum, and the boundaries of the known search space, allow rapid discovery of the actual fault set. Infeasible solutions are efficiently conducted into feasibility by flipping the status of poorly guessed system units. The empirical analysis embraced the two aforementioned test models (invalidation and comparison) and the overall result is an accelerated convergence towards the optimal solution with negligible rise in processing overhead. This makes the proposed method suitable for real-time fault diagnosis in a wide array of compelling scenarios.

The manuscript has been structured as follows. Section 2 is related work and Section 3 elaborates on the two above fault identification models. The building blocks of Dynamic Mesh Optimization are the subject of Section 4. Then we unveil our DMO-based protocol (Section 5), provide simulation results (Section 6), and outline some final remarks (Section 7).

2 Related Work

The extent to which a system can proactively react to unexpected hardware or software faults largely depends on how fast those anomalies are detected. No wonder then that convergence time in fault identification is a highly sought-after feature of any competitive diagnosis method. All the approaches described in this section share the same goal: to locate the set of faulty nodes, given the input syndrome, in the least amount of time and with high reliability. Many research works [1, 23, 27, 28] aim at designing efficient algorithms based on the exploitation of structural properties of the testing graph. Topologies as hypercubes, crossed cubes, extended stars, and twisted cubes are among the preferred topologies because they simplify the conceptual design of the proposed solution approaches.

We stick to a more flexible methodology in which no restrictions on the topological features of the system are laid, other than those to guarantee a reliable (i.e. deterministic) result. In this avenue, system-level fault diagnosis is posed as an optimization problem under a particular test model and an arbitrary structure of the testing graph, as explained in Section 3. Exact, heuristic, and metaheuristic methods are needed to explore the discrete search space in polynomial time. The complexity of such approaches is generally higher compared to those leaning on a simplified topology yet they bear a much wider applicability and still yield satisfactory results.

For example, authors in Kameda *et al,* [13], R.F. Madden [16] put forward exact algorithms of the branch-and-bound type whose time complexity is $O(N^3)$, where N is the system size (number of units). G.F. Sullivan [22] introduced a backtracking-based enhancement to Kameda *et al.* [13] which caused the time complexity to drop to $O(t^3 + |E|)$, where *t* is the number of tolerated faults and $|E|$ is the total number of tests carried out in the system. This bound becomes more relevant as fewer units are deemed defective among a large group of nodes.

As stated in Section 1, evolutionary methods were applied to system-level fault diagnosis [7; 26], in particular GA and AIS. Both methods lack an adequate population diversity given the biased exploration induced by an adaptive mutation operator (openly criticized in Yang *et al.* [26] but finally adopted). As a result of that, the convergence is slowed down and the worst-case identification of faulty nodes turns severely hindered in large systems composed of hundreds or thousands of units. Memory requirements are another important concern because of the enormous number of individuals generated by the algorithms over time in an attempt to find the optimal solution, which consequently triggered the need for a parallel version Elhadef *et al* [8]. Such demeanor turns even worse in the AIS-based model [29], where the number of elitist (fittest) antibodies and the number of clones per elite antibody are both equal to the population size. Our binary DMO implementation requires quite fewer individuals to converge to the optimal solution and does not bias the exploration of the

search space, i.e., all components of the binary vector representing the population member (mesh node) are allowed to change.

An ACO-inspired scheme was designed for system-level fault diagnosis [9] and tested with the invalidation and comparison models. The chief idea is to have every ant construct a full tour of the graph representing the system units and label each node as faulty/fault-free as it goes by. Unlike widely recognized ACO models, authors do not use either pheromone or heuristic values to perform internodal transitions but to guess the node's status. The disadvantage here lies in the redundant storage of dual sets of pheromone trails and heuristic information per node during the ant's journey, which aggravates the memory consumption issue for fairly large systems. Furthermore, it is not clear what heuristic value is assigned to a node by a "flying ant", as no link between the current and the "leaping-to" nodes might exist. Another downside of this meta-heuristic implementation is the handling of infeasible solutions, which are discarded by default, thus wasting valuable information gathered during the ant's incremental tour along the graph. Our proposed approach requires less information at the search agent level (mesh node), gracefully turns infeasible solutions into feasible ones and is theoretically simpler than its ant-based peer.

The binary PSO put forward in Falcon *et al*. [10] has lower computational demands with respect to the previously discussed algorithms and exhibits a superior convergence rate for small and moderate-sized distributed systems. Each particle is encoded as a binary vector representing the status of every node (faulty/fault-free) and "flies" across the search space of candidate fault sets driven by its own best position and the best position ever reached by the swarm. Yet experiments in Falcon *et al.* [11] uncovered a disappointing PSO's worst-case behavior (which might jeopardize the detection capabilities of the algorithm for real-time diagnosable systems) and did not include larger diagnosable systems. We observe in Section 6 how DMO manages to substantially reduce the worst-case diagnosis latency for networks composed of hundreds of interconnected units.

3 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 assume that each node will be tested by a particular subset of the other nodes in the system and every test has a binary nature. 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, they differ in the type of graph representation and the implicit assumptions on how tests are conducted.

3.1 The Invalidation Model

In the invalidation or PMC model (named after its authors in [20], the problem is represented as a directed graph $G = (V, E)$ with $V = \{v_1, v_2, \dots, v_n\}$ being the set of nodes (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 a 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 $v_j \in V$ 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, $\Gamma^{-1}(v_i) = \{v_j \in V : e_{ij} \in E\}$ is the set of v_i's testers and $\sigma^{-1}(v_i) =$

$\{\sigma_{ij} \in \sigma : j \in \Gamma^{-1}(v_i)\}$ the outcomes of all tests carried out upon v_i.

Definition 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 yield 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.

3.2 Comparison Model

In the comparison model [19], 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 is 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) makes up the syndrome σ.

The comparison model also regards the system as static, likewise PMC. Now the main assumption is that two fault-free units will always agree on their test outcomes whereas any couple of faulty and fault-free nodes will always disagree. However, divergent standpoints arise when it comes to the test comparison involving two faulty nodes. The *asymmetric* version of the comparison model [17] considers that two damaged nodes will only produce a mismatch while in the *symmetric* variant [6, 12], 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 concerning v_i.

Definition 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.

3.3 t-Diagnosable Systems

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

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

Hereafter, 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 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 [12] that a $G_t(N)$ 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 [19]. Figure 1 (a) and (b) display a $G_t(N)$ graph under the PMC and symmetric comparison models, respectively.

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, of the actual fault set $\bar{F}$ which univocally generates the known input syndrome $\sigma_{\bar{F}}$. This task has non-polynomial time complexity [4] and calls for the application of heuristic and meta-heuristic methods to informedly navigate over the search space until $\bar{F}$

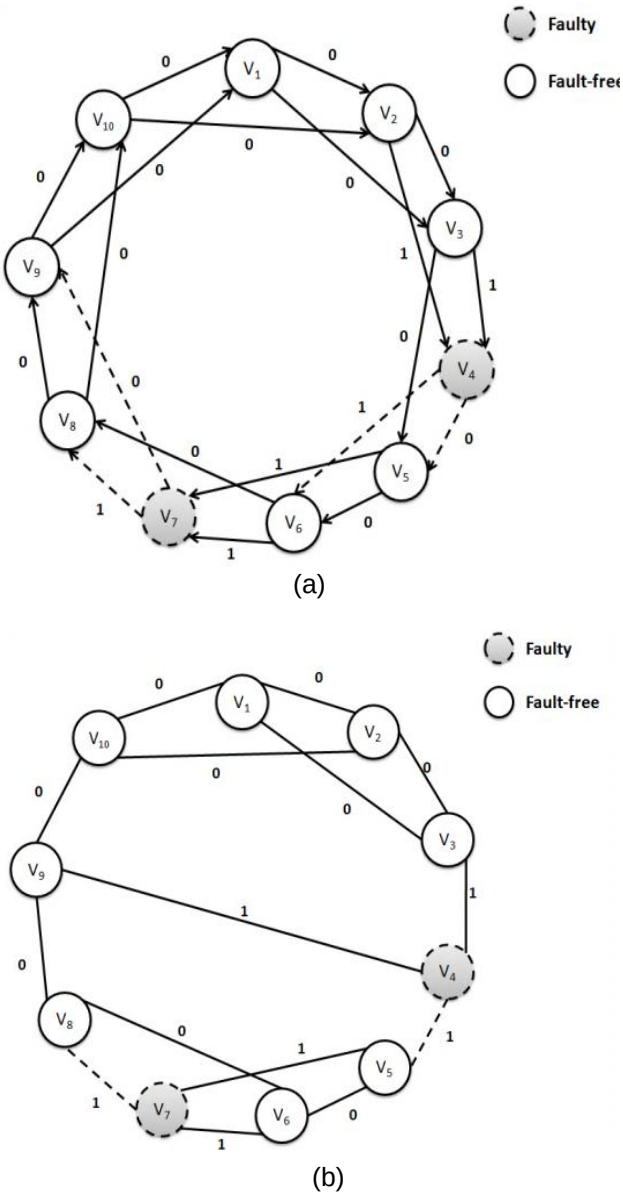


Fig. 1. (a) A $G_2(10)$ test assignment graph under the PMC model. The directed edge labels represent test outcomes; (b) A $G_2(10)$ graph under the symmetric comparison model. The undirected edge labels represent pairwise comparisons of test outcomes

is found. The next section is devoted to outline the fundamentals of one of such meta-heuristic procedures.

4 Dynamic Mesh Optimization

We first glimpse at the important concept of swarm intelligence and highlight PSO as one of its landmark representative architectures, then explain the foundations behind the canonical formulation of Dynamic Mesh Optimization.

4.1 Preliminaries

Nature-inspired algorithms are among the most powerful optimization approaches [26]. Ant colonies, bird flocks, bumble bee nests, bacterial colonies and fish schools are good illustrations of *swarm intelligence*, a term coined not long ago to refer to the *self-organizing* properties of associations of natural organisms which, despite the simplicity and limitations of their constituent individuals, are able to directly or indirectly coordinate their efforts in achieving a certain goal (foraging, mating, etc.) and whose corporate interactions exhibit a far more complex behavior.

The analogy to optimization stems from the fact that these individuals can represent potential solutions in a multimodal, nonlinear, non-differentiable, multidimensional space and by means of their social communication mechanisms, the entire swarm is able to relocate toward more promising regions of the optimization landscape as it pursues the global optimum.

Though originally conceived to target continuous optimization problems [5,14], PSO was rapidly twisted to cope with combinatorial scenarios [15]. Since its very inception, it has enjoyed a well-deserved popularity among swarm intelligence approaches owing to its intuitive formulation, computational simplicity, and fast convergence in many challenging scenarios. Each *particle* is encoded as a multidimensional vector and stands for a prospective solution to the optimization problem. It has a *position* (real-valued or discrete encoding), *velocity* (movement vector), and *fitness* value (measure of quality according to the objective function) and "flies" throughout the search space driven by its best-ever position and the fittest position reached by any neighborhood member thus far.

PSO has undergone quite a few developments [*18*] in critical aspects such as the underlying neighborhood topology, velocity, and position update rules, parameter setting, interaction protocols, and the like. The DMO algorithm we are about to unfold borrows core ideas from PSO and fosters a stronger degree of interactions among individuals in order to enhance the search diversification component.

4.2 DMO's Building Blocks

This population-based metaheuristic was put forward by Puris and Bello [21] after being inspired by the manner in which PSO particles corporately probe the solution space. Unlike PSO, the general formulation of the DMO approach embraces both discrete and continuous optimization domains. A set of nodes representing potential solutions to an optimization problem makes up a *mesh* (population) that dynamically grows and moves through the search space (evolves). This is accomplished via a *mesh expansion* process at each iteration, where new nodes are brought forth in the direction of the *local optima* (mesh nodes with the best fitness in a particular neighborhood), the *global optimum* (the best-so-far solution reached by the algorithm), and those nodes lying at the *mesh border*. Then, the *mesh shrinking* process retains the fittest nodes in the current iteration, which together form the next mesh. This process is repeated until a stop criterion is met.

The building blocks of the DMO method are outlined below and its parameters described in Table 1.

1) Generation of the initial mesh.
2) Node generation towards the local optima.
3) Node generation towards the global optimum.
4) Node generation towards the mesh border.

Bearing the above ideas in mind, DMO was enunciated as a *population-based* optimization algorithm in which a group of artificial nodes V_0 makes up a mesh $\mathcal{M}$ of size M_0. Each candidate solution (mesh node) has an associated *encoding vector* $\overrightarrow{X_i} = (X_i(1), \ldots, X_i(N)),\ i = 1, \ldots, M_0$ in the N-dimensional search space.

In the remainder of this section, we elaborate on the core steps of the DMO protocol.

Table 1. Parameters of the DMO algorithm

M_0	Number of nodes in the initial mesh
M	Maximum number of mesh nodes at any iteration. Often set as $M > 3 \cdot M_0$
k	Neighborhood size, i.e. number of neighbors per node
$maxIter$	Maximum number of iterations to be carried out

1) *Generation of the initial mesh*. At the outset of the algorithm, the $M_i$mesh nodes are created either randomly or by following an inexpensive generation procedure. During each subsequent iteration, the M_0 fittest individuals in the population are retained (out of $M > M_0$ nodes comprising the mesh as a result of the expansion process).

2) *Node generation towards the local optima.* The aim of this step is to come up with new nodes settled in the direction of the local optima found by the algorithm.

For each node $\vec{X}$, its k-nearest neighbours are computed. If none of the neighbors surpasses $\vec{X}$ in terms of fitness function value, then $\vec{X}$ is said to be a local optimum and no nodes are begotten out of it in the current iteration. Otherwise, let $\vec{X}_l$ be the fittest neighbor of $\vec{X}$. In such case, a new node $\vec{X}^*$ arises somewhere between $\vec{X}$ and $\vec{X}_l$according to the function $\vec{X}_l(k) = f\left(\vec{X}(k), \vec{X}_l(k), r_l\right)\ \forall k = 1..N$, where r_l models the proximity of the newborn individual to each ancestor by somehow relating their fitness values. Depending on the application domain DMO deals with, r_l could be interpreted as either a perturbation factor of the real-valued vector $\vec{X}_l$ or as a probabilistic threshold for componentwise selection out of the discrete vectors $\vec{X}$ and $\vec{X}_l$. Either way, $M_l = |V_l| < M_0$ individuals will be added to the mesh at this time.

3) Node generation towards the global optimum. Analogously, we allow the original mesh nodes to produce individuals in the direction of the best solution ever found by the algorithm, $\vec{X}_g$.

The encoding of the new vector $\vec{X}^*$ is determined by the mapping g as follows: $\vec{X}^*(k) = g\left(\vec{X}(k), \vec{X}_g(k), r_g\right) \forall k = 1..N$, where r_g plays a similar role to r_l in the previous step.

Notice that exactly $M_g = |V_g| = M_0 - 1$ nodes will become part of the dynamic mesh should $\vec{X}_g$ be a member of the current population, otherwise $M_g = |V_g| = M_0$.

4) Node generation towards the mesh borders. The last step in the mesh expansion process aims at intensifying the exploration around the *mesh border*. Although this definition is highly problem-dependent, the idea is that nodes lying at the farthest regions of the known search space can somehow give rise to new candidate solutions that hopefully stretch out the mesh. The mapping h governs this step: $\vec{X}^*(k) = h\left(\vec{X}_b(k), w\right) \forall k = 1..N$, where $\vec{X}_b$ is an individual located at the mesh border (picked out of the M_0 initial nodes in the current mesh) and w has a similar nature to prior variables r_l and r_g but relates to $\vec{X}_b$ alone.

In total, $M_b = |V_b| = M - M_0 - M_l - M_g$ nodes will be appended to the mesh. It is a good thing in practice to choose $\lceil M_b/2 \rceil$ population members located at the "lower border" and the rest belonging to the "upper border" as pivots for node generation.

5 Proposed Approach

We model system-level fault diagnosis as a combinatorial optimization problem undertaken by a binary DMO approach. The goal is to find, among many possible fault sets (see Section 3), the true set $\bar{F}$ that entirely matches the collection of test outcomes contained in the input syndrome $\sigma_{\bar{F}}$. The proposed protocol (Dynamic Mesh Optimization to Fault Diagnosis, DMO-FD) is executed at a centralized location (e.g., a dedicated processor in a parallel system, or a base station or cluster head node in a wireless sensor network).

5.1 Solution Encoding and Initialization

In DMO-FD, each mesh node symbolizes a candidate fault set F. Therefore, we encode its position $\vec{X}_i$ as a binary vector whose dimensionality $N = |V|$ is given by the system size (number of distributed or parallel units). The status of the k-th node $(k = 1..N)$ will be denoted by $\vec{X}_i(k) = 1$ if we guess that v_k is faulty or $\vec{X}_i(k) = 0$ otherwise.

At the algorithm outset, it is first decided at random the number of faulty units $nf \sim U(i, t)$ each mesh node will encode and then nf bits will be arbitrarily set to one in $\vec{X}_i, i = 1..M_i$. By doing so, we are trying to promote diversity in the initial mesh, since the cardinality of its encoded fault sets will vary stochastically within the feasible bounds imposed by a t-diagnosable system.

For each potential fault set F, we generate its corresponding candidate syndrome σ_F before we can actually figure out its fitness value.

5.2 Fitness Function

The quality or fitness of any individual (fault set F) in our problem is measured as the resemblance between the input syndrome $\sigma_{\bar{F}}$ and the candidate syndrome σ_F. The calculation of the latter 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 (mesh node's encoding), we follow the asymmetric model rules stated in Section 3.2 and assign a label to every edge $e_{ij} \in E$ in the test assignment graph G. The collection of these labeled edges becomes our candidate syndrome σ_F.

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

$$\sigma_F(v_i) = \sigma_{\bar{F}}(v_i) \; \forall v_i \in F$$

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 3.2. We denote by $z : \vec{X}_i \rightarrow \sigma_{F_i}$ the mapping from a candidate fault set $\vec{X}_i$ to its corresponding syndrome σ_{F_i}.

2) The Mesh Node's Fitness. A mesh node is said to have better quality (fitness) than peers if the candidate syndrome associated with its encoding (fault set F) better resembles the input syndrome. 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 two variants of the comparison model, expression (1) models the node-level similarity.

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

For the PMC model, we are to take into account that all nodes will be tested but not necessarily testers. Equations 2 to 4 respectively model the similarity from v_i's perspective as a tester node, tested node, and both.

$$f^{+1}\left(\sigma_F, \sigma_{\overline{F}}, v_i\right) = \begin{cases} 1, & \text{if } \left|\Gamma(v_i)\right| = 0 \\ \frac{\left|\sigma_F(v_i) \cap \sigma_{\overline{F}}(v_i)\right|}{\left|\Gamma(v_i)\right|}, & \text{otherwise} \end{cases} \tag{2}$$

$$f^{-1}\left(\sigma_F, \sigma_{\overline{F}}, v_i\right) = \frac{\left|\sigma_F^{-1}(v_i) \cap \sigma_{\overline{F}}^{-1}(v_i)\right|}{\left|\Gamma^{-1}(v_i)\right|} \tag{3}$$

$$f\left(\sigma_F, \sigma_{\overline{F}}, v_i\right) = \frac{f^{+1}\left(\sigma_F, \sigma_{\overline{F}}, v_i\right) + f^{-1}\left(\sigma_F, \sigma_{\overline{F}}, v_i\right)}{2} \tag{4}$$

Now we can define in (5), regardless of the diagnosis model under consideration, the overall resemblance function between the input and candidate syndromes, which actually becomes the fitness function of the DMO-FD algorithm.

$$f\left(\sigma_F, \sigma_{\overline{F}}\right) = \frac{\sum_{v_i \in V} f\left(\sigma_F, \sigma_{\overline{F}}, v_i\right)}{N} \tag{5}$$

where $N = |V|$ is the system size.

The above equation can be seen as the correctness probability of the potential fault set F being the actual fault set $\bar{F}$. Because we are working with t-diagnosable systems solely, then it is guaranteed that there is a unique fault set F that makes $\sigma_F =$. Remark that (1) and (4) take values in $[0; 1]$, which consequently defines the image of (5) to lie in the same interval.

5.3 Node Generation toward Local Optima

Whenever a new node $\vec{X}^*$ is to be created out of current node $\vec{X}$ and its local optimum $\vec{X}_l$, its components along each dimension $k = 1..N$ are generated as in (6).

$$\vec{X}^*(k) = f(\vec{X}(k), \vec{X}_l(k), r_l) = \begin{cases} \vec{X}_l(k), & \text{if rand}() \leq r_l \\ \vec{X}(k), & \text{otherwise} \end{cases} \tag{6}$$

where $rand() \sim U(0,1)$. The proximity threshold r_l is thus modeled after (7).

$$r_l = 1 - 0.5 \times \frac{\text{fitness}(\vec{X})}{\text{fitness}(\vec{X}_l)} \tag{7}$$

Notice that if the fitness values of $\vec{X}$ and $\vec{X}_l$ are slightly different, then r_l is about 0.5, which means that each component of $\vec{X}^*$ can be taken out of either vector ($\vec{X}$ or $\vec{X}_l$) with roughly the same probability. On the other hand, if $\vec{X}$ is quite inferior to $\vec{X}_l$ in fitness, then $r_l \approx 1$ and $\vec{X}^*$ will be biased towards $\vec{X}_l$ to a large extent.

5.4 Node Generation toward Global Optimum

An alike rationale is applied for giving rise to mesh nodes in the direction of the best-performing individual. The mapping g and its associated proximity threshold r_g are calculated by means of (8) and (9).

$$\vec{X}^*(k) = g(\vec{X}(k), \vec{X}_g(k), r_g) = \begin{cases} \vec{X}_g(k), & \text{if rand}() \leq r_g \\ \vec{X}(k), & \text{otherwise} \end{cases} \quad (8)$$

$$r_g = 1 - 0.5 \times \frac{\text{fitness}(\vec{X})}{\text{fitness}(\vec{X}_g)} \quad (9)$$

5.5 Node Generation toward Mesh Border

At this point, we try to "enlarge" the artificial mesh, i.e., to intensify the exploration around its border. We can determine whether a node lies at the mesh border or not by computing the norm of the vector it encodes.

Since we work with binary solutions, the norm is simply the number of components set to 1. We denote by *lower border* (*upper border*) the region of the search space containing those vectors having minimal (maximal) norm. Notice that, in the context of system-level fault diagnosis, the norm of any feasible vector ranges between 1 and t (the number of faulty units in the network), so lower-border nodes are those having norm 1, 2, etc., and upper-border nodes will have norm t, $t-1$, $t-2$, etc.

The number of nodes generated during this step is $M_b = |V_b| = M - M_0 - M_l - M_g$. Hence, we seek $\lceil M_b/2 \rceil$ nodes lying at the lower border and the same number of individuals in the upper border. In our implementation, mesh border representatives shall be drawn from the set $V_o \cup V_l \cup V_g$, i.e., all nodes in the current iteration, as a vehicle to better capture the dynamics of the population.

Let $\vec{X}_{lower}$ and $\vec{X}_{uppwer}$ be two representatives of the corresponding classes. Then, the dual character of the h mapping $\forall k = 1..N$ is captured by (10) and (11), respectively.

$$\vec{X}^*(k) = h(\vec{X}_{lower}(k), w) = \begin{cases} 0, & \text{if rand}() \leq w \\ \vec{X}_{lower}(k), & \text{otherwise} \end{cases} \quad (10)$$

$$\vec{X}^*(k) = h(\vec{X}_{upper}(k), w) = \begin{cases} 1, & \text{if rand}() \leq w \\ \vec{X}_{upper}(k), & \text{otherwise} \end{cases} \quad (11)$$

with w calculated as shown below:

$$w = (w_0 - w_f) \times \frac{\text{maxIter} - \text{iter}}{\text{maxIter}} + w_f \quad (12)$$

where "iter" is the current iteration number, "maxIter" is the maximum number of iterations, and w_0, w_f are the initial and final values for w, respectively ($w_0 > w_f$). Remark that w has a monotonically decreasing nature, i.e. more exploration towards the mesh borders is encouraged at the beginning of the algorithm and decreases over time.

5.6 Handling Infeasible Solutions

The randomness that permeates the mesh expansion stage in (6), (8), (10), and (11) could bring forth an *infeasible solution*, defined as a particular configuration (binary vector) in which either all units are fault-free or there are more than w 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 (1) or (4) depending on the diagnosis model under consideration.

2) If the mesh node's encoding is the zero vector, generate a random number $b \sim U(1, t)$ and flip (turn to 1) the b bits with the lowest fitness values. Otherwise, let k be the number of 1's in the vector (notice that $k > t$). Then generate a random number $b \sim U(k-t, k-1)$ and flip (turn to 0) the b bits set to 1 with the lowest fitness values.

The above procedure rapidly turns infeasible individuals into feasible ones by inverting the guesses on the status of those nodes with

poorest local similarity to the input syndrome, in an attempt to accelerate the convergence of the algorithm towards the actual fault set.

5.7 Stop Criterion

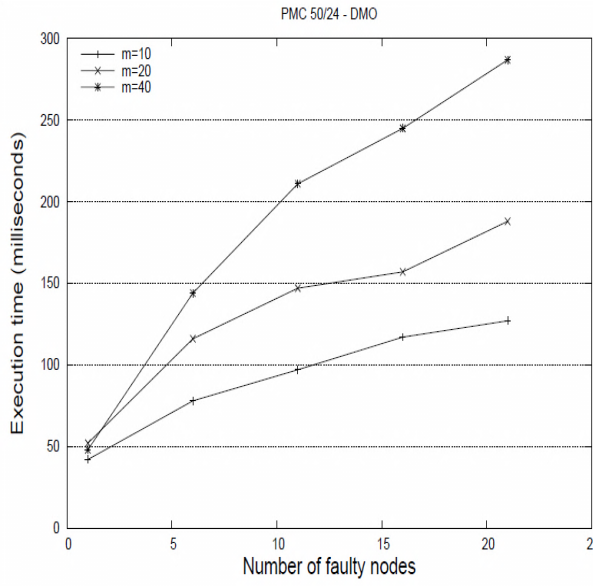
DMO-FD stops when it either reaches an upper bound of the running time (usually expressed as a function of the number of iterations) or discovers the actual fault set $\bar{F}$, i.e., a node encoding F with fitness$(\sigma_F, \sigma_{\bar{F}}) = 1$. Such F is guaranteed to be unique in t-diagnosable systems [Pelc, 1991; Hakimi *et al*, 1974].

6 Simulation Results

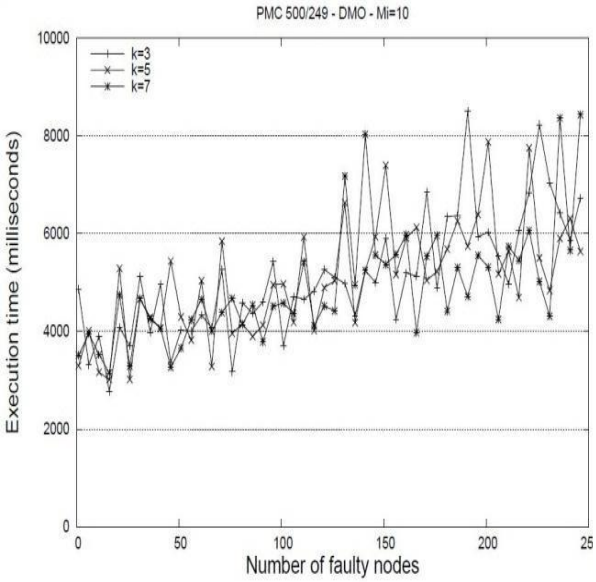
We have empirically validated the performance of the DMO-FD protocol under *t*-diagnosable systems of various sizes for both the invalidation and comparison test models. Our algorithm has been contrasted to the AIS in Yang *et al.* [26] and the PSO in Falcon *et al.* [10]. The experiments were conducted in Java using JDK 1.6 on an Intel Core2 Duo E4500 2.2GHz with 2 GB of RAM under Windows Vista.

Each algorithm was executed until either the true fault set $\bar{F}$ was encountered or a maximum allotted running time was reached. The spatio-temporal demands of the competing approaches were measured by the following performance indicators: (1) total number of candidate solutions probed and (2) CPU time consumed. To assess the methods' scalability, t-diagnosable systems of different sizes were tried, viz $G_{24}(50)$, $G_{49}(100)$, $G_{249}(500)$, and $G_{499}(1000)$.

For each system above, the number of faults *x* ranged from 1 to *t* and for every value of *x*, 100 arbitrary fault sets of that cardinality were generated. The average-case and worst-case behaviors of every algorithm with the two previous performance metrics have been recorded. The 95% confidence intervals of the average case are not shown in every plot for the sake of clarity. Rather, they are displayed mainly when highlighting differences between PSO and DMO-FD, provided either succeeded in discovering $\bar{F}$.



(a)



(b)

Fig. 2. (a) DMO-FD in a $G_{24}(50)$, small M gives a higher performance; (b) DMO-FD in a $G_{249}(500)$, no meaningful differences among the three tested *k* values are spotted for most of the trials across tested algorithms, diagnosis models, and system sizes

6.1 Parameters of Experiments

We adopted the AIS configuration outlined in X.S.Yang [29], where the number of elite antibodies undergoing cloning is equal to the population size and a quarter of the individuals is replaced after every iteration. For PSO, the recommended values in competitive implementations [5] were respected, i.e., $c_1 = c_2 = 2.05$, and the constriction factor $\chi = 0.7298$ is included in the velocity update rule. PSO works with the "complete" or "global best" neighborhood layout while DMO-FD's topological structure is controlled by the parameter *k*.

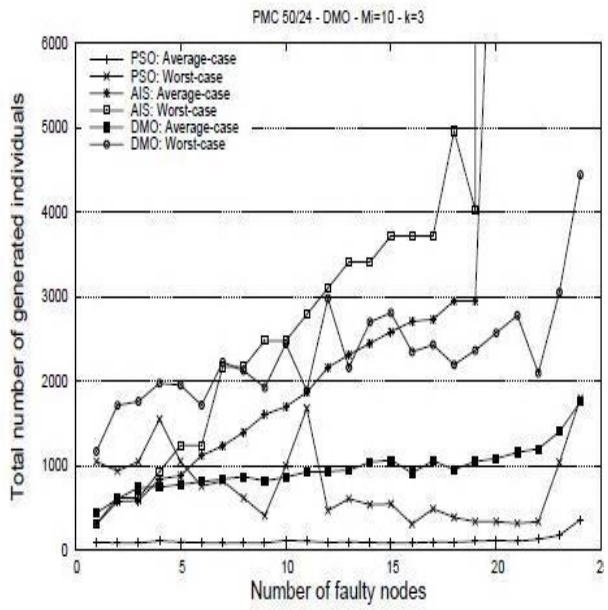
To guarantee a fair comparison baseline, the three metaheuristic schemes share the same population size (10 individuals). We set this value as such because we are interested in gauging the algorithmic performance in resource-scant environments, i.e. assuming the device in charge of running the fault diagnosis scheme has limited computing and memory capabilities.

A further justification to the small population size is the demonstrated fact Falcon *el al.* [11] that a modest number of search agents manages to explore reasonably well the binary *N*-dimensional space. Fig. 2(a) confirms the higher performance when using M=10 than higher values.

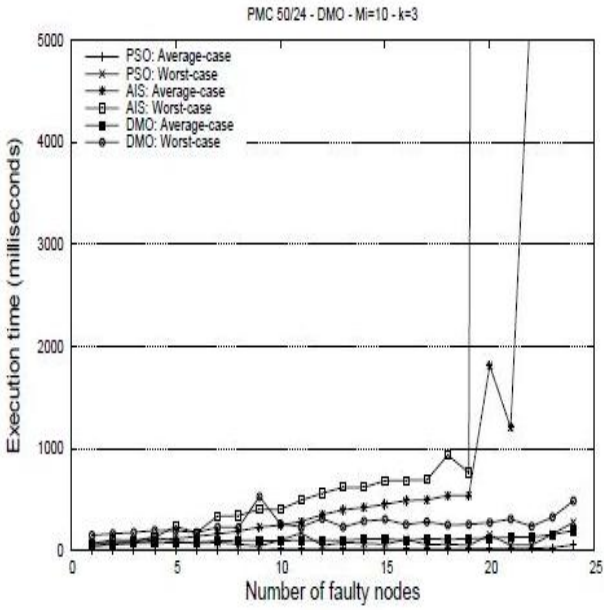
The selection of the neighborhood size (*k*) in DMO followed a more careful analysis. Fig. 2(b) reveals that no significant differences arise among all tested values $k \in \{3,5,7\}$. This is true for the vast majority of the configurations tested and the two empirical metrics and diagnosis models under consideration. Hence, setting $k = 3$ contributes to diminish the computational complexity of the algorithm.

6.2 PMC Model, Small Graph

Fig. 3(a) displays the cumulative number of candidate solutions probed along each algorithm's lifetime in presence of the invalidation model and a small system $G_{24}(50)$. The two steep AIS curves are due to the heavy effect of cloning, a vital mechanism to guarantee diversification in AIS along with the ensuing mutation operator. Because at every AIS iteration the highest ranked antibodies are cloned as many times as the



(a) Exploratory capabilities of the three algorithms



(b) Convergence speed of the three algorithms

Fig. 3. Performance metrics for AIS, PSO and DMO-FD with the PMC model and $G_{24}(50)$ graph

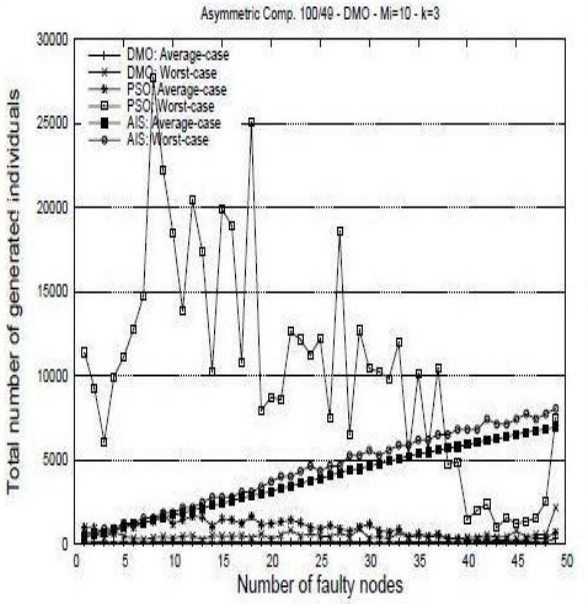
population size itself, the entailed memory requirements of such protocol may be prohibitive if we consider running it on a limited device like an ordinary sensor acting as cluster head in a wireless sensor network. DMO-FD's worst-case curve is quite erratic and strongly contrasts with its roughly steady average-case trend. PSO's performance is indeed encouraging (on average) as the number of presumably damaged units grows. Its worst-case scenario is preferable to DMO's average case once 12 or more faulty nodes are identified in the network. PSO's tangible superiority over DMO-FD in small graphs could be ascribed to the negligible bearing of stretching the mesh toward its borders when the search space is quite narrow. In such cases, it seems that the stochastic flight of the PSO particles produces a much more efficient exploration than the node generation toward poorly guessed local optima and global optimum in DMO. Should this conjecture be right, we will behold a dramatic change in DMO's demeanor as the search space dimensionality rises.

Concerning running time, Fig. 3(b) helps us realize that only AIS fails to identify 20 or more damaged units in less than five seconds. The large number of individuals spawned in AIS has an immediate impact on its computational complexity, as represented by the two curves with the sharpest growth rates. Conversely, the response time of the two remaining metaheuristic approaches stays nearly unaltered as more faulty nodes are added, which means that having the global best population member and a locally optimal solution guide the search of the rest of the agents is profitable for scalability purposes. PSO still beats DMO-FD and finds the true fault set quicker.

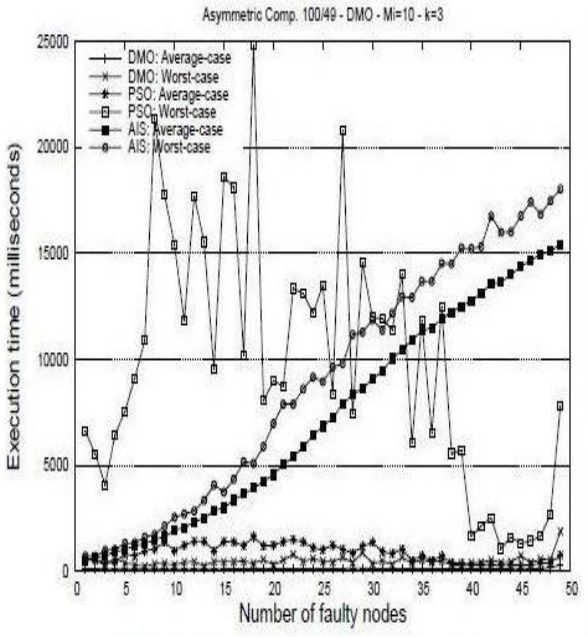
Similar results are obtained for the $G_{24}(50)$ in the comparison model, which are therefore left out of the discussion.

6.3 Asymmetric Comparison Model, Medium-Size Graph

When the three algorithms are tested with a medium-size distributed system (100 nodes) following the asymmetric comparison model, there is a significant variation in their performance w.r.t. the previous case. For instance, Fig. 4(a)



(a) Exploratory capabilities of the three algorithms



(b) Convergence speed of the three algorithms

Fig. 4. Performance metrics for PSO, AIS and DMO-FD with the Asymmetric Comparison model and $G_{49}(100)$ graph

reveals that PSO shows an irregular worst-case behavior in the number of particles required to find the global optimum. Interestingly, the stagnation phenomenon slowly fades out as the dimension of the search space augments. One could argue that the presence of a single informer (swarm's best) in a particular region of the search space could be drawing other particles strongly enough so as to lead them into an infeasible encoding. Particles can more easily escape stagnation and be redirected towards any stochastic, feasible encoding as the size of the search space increases. Fortunately, the joint influence of manifold informers together with the exploration around the mesh border prevents DMO-FD agents from getting caught in local optima, as proved by the fact that it consumes a nearly constant and negligible number of candidate solutions to converge. The evolutionary search undertaken by AIS yields almost identical results for the average and worst cases.

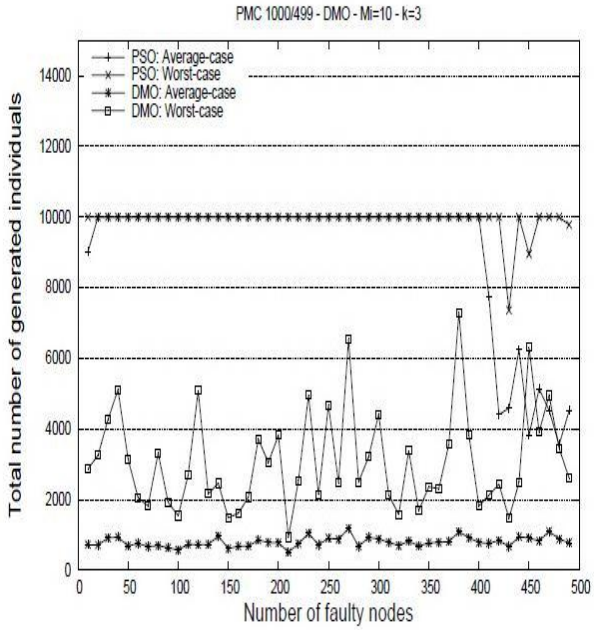
Dynamic-mesh-based optimization outperforms peer methods in terms of running time as well. From Fig. 4(b) it can be noticed how the insufficient guidance in PSO still produces a significant delay in convergence speed. Although PSO and DMO-FD do very well in the average case, for the former it takes over ten times longer to detect the faulty set than for the latter in the worst scenario. Such delay could be unacceptable in critical environments like aeronautics, underwater oil drilling, etc.

Nevertheless, most of the times both approaches efficiently reach the global optima in less than 2 seconds. The reliability of the average time prediction for the proposed method is schematically verified by the short length of the majority of the confidence intervals, with still quite some margin for PSO improvement.

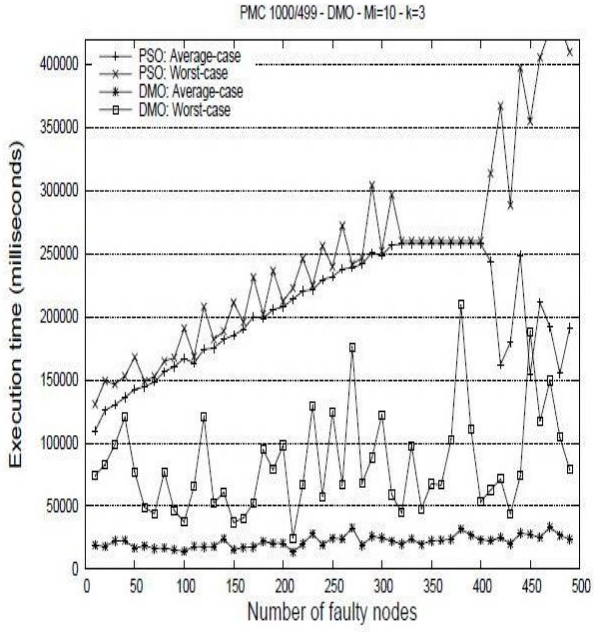
6.4 PMC Model, Large Graph

Let us shed light on the empirical performance between DMO-FD and PSO in presence of a large, complex system $G_{249}(500)$. We have excluded AIS from this trial since its behavior is the poorest and quite impractical.

Fig. 5(a) bears witness to the inability of PSO in pointing out the actual fault set for most of the values of $t \in \{1..200\}$. This fact is depicted by the



(a) Exploratory capabilities of the two algorithms



(b) Convergence speed of the two algorithms

Fig. 5. Performance metrics for PSO and DMO-FD with the PMC model and $G_{249}(500)$ graph

straight line topping the maximum number of individuals unfolded along 1,000 iterations. Its vanishing effect can be explained by the same rationale in Section 6.3. While DMO-FD never failed to encounter the true ensemble of damaged nodes, PSO fell short in 78% of the cases. Remark that both methods are contrasted with a small population size (*M=10*), which makes DMO-FD a strong candidate for tackling challenging fault detection problems with low resource requirements. Even DMO-FD's worst-case setting is far more preferable than PSO's average-case.

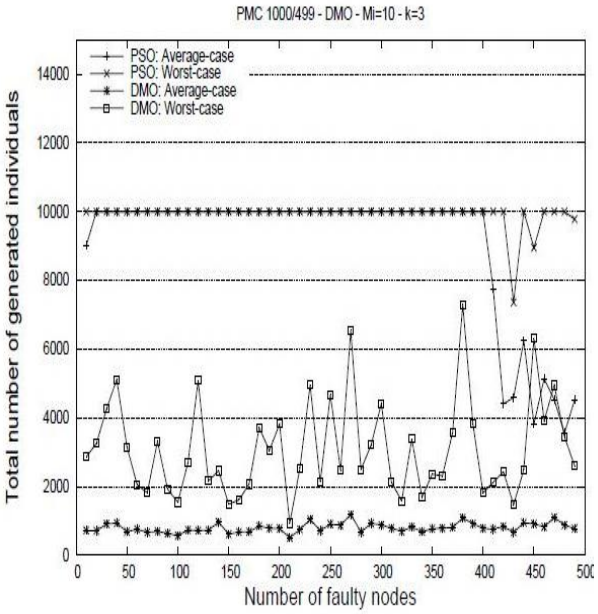
After inspecting Fig. 5(b), a similar trend to Fig. 5(a) surfaces, except that the running time for PSO increases nearly monotonically with the expected number of faulty units until a large enough search space is reached for $t > 200$. In such cases, the presence of numerous feasible candidate sets helps attenuate the undesirable effect of having a single informer particle draw all remaining population members towards a reduced group of infeasible encodings. Because of PSO's high failure rate, its average-case behavior approaches the worst-case performance. On the other hand, observe how DMO-FD responds with the true fault set within 10 seconds on average and always within 22 seconds, irrespective of the values of *t*. Its responsiveness can be ascribed to the enhanced exploration around the mesh border and the presence of multiple local minima guiding the search.

6.5 PMC Model, Large Graph – 1000 nodes

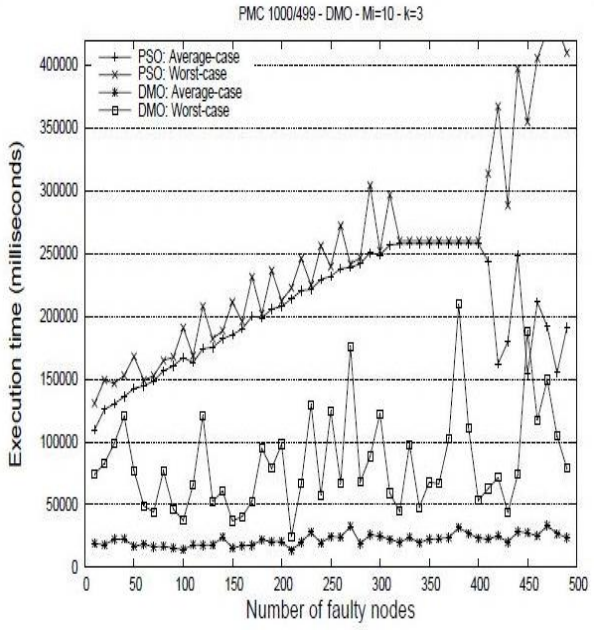
Finally using a system $G_{499}(1000)$ the DMO-FD is able to find solutions within good average time for any number of faulty nodes, as depicted in Fig. 6 (a) and (b). The PSO is unable to find solutions until a large enough search space is reached for $t > 400$. DMO-FD responds with the true fault set within 20 seconds on average and the most cases within 100 seconds, irrespective of the values of *t*. So, with the reported results we can confirm the scalability of the DMO-FD for complex systems.

7 Conclusions and Future Work

In this study, we have cast the relevant problem of system-level fault diagnosis into the



(a) Exploratory capabilities of the two algorithms



(b) Convergence speed of the two algorithms

Fig. 6. Performance metrics for PSO and DMO-FD with the PMC model and $G_{499}(1000)$ graph

combinatorial optimization world and applied a discrete version of Dynamic Mesh Optimization, a novel population-based optimization method, to quickly and affordably identify the set of faulty units in the system. The proposed DMO-FD protocol utilizes a binary encoding of the potential solutions (mesh nodes) and dynamically modifies the mesh topology via node generation toward the local optima, global optimum, and mesh border. Whenever a population member drifts into the infeasible region, the assumption on the status of the least promising nodes is inverted, thus causing its candidate syndrome to better resemble the input syndrome.

Though PSO still remains as the best optimization alternative for small graphs (less than 50 nodes), simulation results amply argue for the introduction of DMO-based fault detection in medium and large-size networks in presence of the invalidation and comparison test models. In such scenarios, which are increasingly prolific in today's societies, DMO-FD displays a robust and more stable behavior in terms of convergence rate and amount of memory required. This speaks highly of the scalability properties of our algorithm.

As future work, we plan to incorporate other fault diagnosis models and performance metrics into the experimental setting and bring forth further enhancements through the hybridization with other population-based meta-heuristics or the application of local search mechanisms that intensify the exploitation of the corporately induced potential fault sets.

References

1. **Ahlswede, R. & Aydinian, H. (2008).** On Diagnosability of Large Multiprocessor Networks. *Discrete Applied Mathematics*, 156(18), 3464–3474.
2. **Ayeb, B. (1999).** Fault Identification Algorithmic: A New Formal Approach. *29th Annual International Symposium on Fault-Tolerant Computing*, Madison, Wisconsin, USA, 138–145.
3. **Bello, R., Puris, A., Falcon, R., & Gómez, Y. (2008).** Feature Selection through Dynamic Mesh Optimization. *Progress in Pattern Recognition, Image Analysis and Applications, Lecture Notes in Computer Science*, 5197, 348–355.
4. **Blough, D.M. & Pelc, A. (1992).** Complexity of Fault Diagnosis in Comparison Models. *IEEE Transactions on Computers*, 41(3), 318–324.
5. **Bratton, D. & Kennedy, J. (2007).** Defining a Standard for Particle Swarm Optimization. *2007 IEEE Swarm Intelligence Symposium (SIS 2007),* Honolulu, HI, USA, 120–127.
6. **Chwa, K.Y. & Hakimi, S.L. (1981).** Schemes for Fault Tolerant Computing: a Comparison of Modularly Redundant and t-Diagnosable Systems. *Information & Control*, 49(3), 212–238.
7. **Elhadef, M. & Ayeb, B. (2000).** An Evolutionary Algorithm for Identifying Faults in t-Diagnosable Systems. *19th IEEE Symposium on Reliable Distributed Systems (SRDS-2000)*, Nurnberg, Germany, 74–83.
8. **Elhadef, M., Das, S., & Nayak, A. (2005).** A Parallel Genetic Algorithm for Identifying Faults in Large Diagnosable Systems. *The International Journal of Parallel, Emergent and Distributed Systems,* 20(2), 113–125.
9. **Elhadef, M., Nayak, A., & Zeng, N. (2007).** An Ant-based Fault Identification Algorithm for Distributed and Parallel Systems. *10th World Conference on Integrated Design & Process Technology (IDPT-2007)*, Antalya, Turkey, 1–6.
10. **Falcon, R., Almeida, M., & Nayak, A. (2010).** A Binary Particle Swarm Optimization Approach to Fault Diagnosis in Parallel and Distributed Systems*. 2010 IEEE Congress on Evolutionary Computation (CEC)*, Barcelona, Spain, 1–8.
11. **Falcon, R., Li, X., Nayak, A., & Stojmenovic, I. (2010).** The One-Commodity Traveling Salesman Problem with Selective Pickup and Delivery: an Ant Colony Approach. *2010 IEEE Congress on Evolutionary Computation (CEC)*, Barcelona, Spain, 1–8.
12. **Hakimi, S.L. & Amin, A.T. (1974).** Characterization of the Connection Assignment of Diagnosable Systems. *IEEE Transactions on Computers*, C-23(1), 86–88.
13. **Kameda, T., Toida, S., & Allan, F.J. (1975).** A Diagnosis Algorithm for Networks. *Information & Control*, 29(2), 141–148.
14. **Kennedy, J. & Eberhart, R. (1995).** Particle Swarm Optimization. *IEEE International Conference on Neural Networks*, Perth, Australia, 1942–1948.
15. **Kennedy, J. & Eberhart, R.C. (1997).** A Discrete Binary Version of the Particle Swarm Algorithm. *1997 IEEE International Conference on Systems, Man, and Cybernetics*, Orlando, Florida, USA, 5, 4104–4108.

16. **Madden, R.F.** On Fault-Set Identification in Some System-Level Diagnostic Models", *Proc. Int'l Symposium on Fault-Tolerant Computing*, Jun. 1977.
17. **Maeng, J. & Malek, M. (1981).** A Comparison Connection Assignment for Self-Diagnosis of Multiprocessor Systems, *Proc. 11th International Symposium on Fault-Tolerant Computing*, New York, USA, 1981, pp. 173–175
18. **Montes de Oca, M.A., Stutzle, T., Birattari, M., & Dorigo, M. (2009).** Frankestein's PSO: A Composite Particle Swarm Optimization Algorithm. *IEEE Transactions on Evolutionary Computation*, 13(5), 1120–1132.
19. **Pelc, A. (1991).** Undirected Graph Models for System-Level Fault Diagnosis. *IEEE Transactions on Electronic Computers*, 40(11), 1271–1276.
20. **Preparata, F.P., Metze, G., & Chien, R.T. (1967).** On the Connection Assignment Problem of Diagnosable Systems. *IEEE Transactions on Electronic Computers,* EC-16(6), 848–854.
21. **Puris, A. & Bello, R. (2009).** Optimización basada en Mallas Dinámicas. Su Aplicación en la Solución de Problemas de Optimización Continuos. *VI Congreso Español Sobre Metaheurísticas, Algoritmos Evolutivos y Bioinspirados (MAEB'09),* Málaga, Spain, 441–448.
22. **Sullivan, G.F. (1988).** An $O(t^3+|E|)$ Fault Identification Algorithm for Diagnosable Systems. *IEEE Transactions on Computers*, 37(4), 388–397.
23. **Tzu-Liang, K., Hsing-Chung, C., & Tan, J.J.M. (2010).** On the Faulty Sensor Identification Algorithm of Wireless Sensor Networks under the PMC Diagnosis Model. *6th International Conference on Networked Computing and Advanced Information Management (NCM),* Seoul, Korea*,* 657–661.
24. **Vaquero L.M., Rodero-Merino, L., Caceres, J., & Lindner, M. (2009).** A Break in the Clouds: Towards a Cloud Definition. *ACM SIGCOMM Computer Communication Review*, 39(1), 50–55.
25. **Verdone, R., Dardari, D., Mazzini, G., & Conti, A. (2008).** *Wireless Sensor and Actuator Networks: Technologies, Analysis and Design*, London: Academic Press.
26. **Yang, H., Elhadef, M., Nayak, A., & Yang, X. (2008).** Network Fault Diagnosis: An Artificial Immune System Approach. *14th IEEE International Conference on Parallel and Distributed Systems*, Melbourne, Australia, 463–469.
27. **Yang, H., Yang, X., & Nayak, A. (2010).** A $(4n-9)/3$ Diagnosis Algorithm for Generalised Cube Networks. *International Journal of Parallel Emergent and Distributed Systems*, 25(3), 171–182.
28. **Yang, X., Megson, G.M., & Evans, D.J. (2005).** A Comparison-based Diagnosis Algorithm Tailored for Crossed Cube Multiprocessor Systems. *Microprocessors and Microsystems,* 29(4), 169–175.
29. **Yang, X.S. (2008).** *Nature-Inspired Metaheuristic Algorithms*. Cambridge: Luniver Press.

Rafael Falcon received a Ph.D. degree from the University of Ottawa, Canada in 2012, and M.S. and B.S. (with highest honors) degrees from Universidad Central de Las Villas, Cuba in 2003 and 2006, respectively, all in computer science. He was a Visiting Scholar for Hasselt Universiteit, Belgium, and the Universidad de Granada, Spain. He has co-edited two Springer volumes on fuzzy and rough set theories and served as reviewer for top-tier scientific journals. His current research interests embrace wireless sensor and robot networks, distributed computing, bio-inspired optimization, rough sets, fuzzy logic and knowledge-based clustering. He is a member of the International Rough Set Society and the IEEE Computational Intelligence Society.

Marcio Almeida received his B.S. degree in Computer Science and M.S. degree in Computer Networks from Universidade Salvador, Brazil in 2000 and 2003, respectively, and Ph.D. in Computer Science from Universidade de São Paulo, Brazil in 2009. Currently, he is a postdoctoral fellow at University of Ottawa, Canada. His research interests embrace wireless sensor and robot networks, distributed computing, bio-inspired optimization, data mining, and information visualization.

Amiya Nayak received his B.Math. degree in Computer Science and Combinatorics & Optimization from University of Waterloo in 1981, and Ph.D. in Systems and Computer Engineering from Carleton University in 1991. He has over 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 & 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 a Full Professor at the School of Electrical Engineering and Computer Science at the University of Ottawa. His research interests are in the area of fault tolerance, distributed systems/algorithms, and mobile ad hoc networks.

Rafael Esteban Bello Pérez received his Bachelor degree in Mathematics and Computer Science (1982) at Universidad Central de Las Villas (UCLV), Santa Clara, Cuba and his Ph.D. in Mathematics at UCLV in 1988. He has been a visiting scholar of several universities in Spain, Germany, and Belgium. He is a Full Professor at Computer Science Department, UCLV**,** Cuba, and exhibits a long record of academic exchange with many institutions in Latin America and Europe. Dr. Bello has coordinated Master programs in Applied Computing and Computer Science, and is currently the coordinator of the Ph.D. program in Informatics at UCLV. He was the Chair of the Computer Science Department, Dean of the Computer Science Faculty, and vice-president of UCLV. Dr. Bello has taught over 45 undergraduate and graduate courses and published over 150 papers in conference proceedings and scientific journals. He has authored/edited 9 books and supervised over 30 Bachelor, Master and Ph.D. theses. He has deserved prestigious awards from the Cuban Academy of Sciences (CAS) and other renowned scientific societies. Dr. Bello is the Director of the Center of Studies on Informatics at UCLV and became a Member of AAAI in 2004. He runs the academic collaboration project between the Flemish Interuniversity Council and UCLV. He has been a CAS member since 2002, and also the vice president of the National Board for Ph.D. Degree in Mathematics, Computing and Automation of Cuba. His research interests comprise meta-heuristics, soft computing (rough and fuzzy set theories), machine learning techniques, and decision making.

Article received on 12/02/2011; accepted on 27/10/2011.