PERIODICA POLYTECHNICA SER. EL. ENG. VOL. 2, NO. 2, PP. 127–149 (2000)

PROBABILISTIC FAULT DIAGNOSIS IN LARGE, HETEROGENEOUS COMPUTING SYSTEMS

Tamás BARTHA* and Endre SELÉNYI**

* Computer and Automation Research Institute,
Hungarian Academy of Sciences
H-1111 Budapest, Kende u. 13-17, Hungary
Phone/Fax: (+361) 466-7483, E-mail: bartha@sztaki.hu
** Department of Measurement and Information System,
Budapest University of Technology and Economics
H–1521 Budapest, Műegyetem rkp. 9, R/113, Hungary
Phone: (+361) 463-2057, Fax: (+361) 463-4112, E-mail: selenyi@mit.bme.hu

Received: Dec. 20, 1999

Abstract

Probabilistic diagnosis aims at making the system-level fault diagnostic problem both easier to solve and the resulting algorithms more generally applicable. The price to pay for these advantages is that the diagnostic result is no longer guaranteed to be correct and complete in every fault situation. This paper presents a novel approach, called *local information diagnosis* (LID), and applies this methodology to create a family of probabilistic diagnostic algorithms. The developed algorithms can be divided into three main classes: *limited inference*, *limited information*, and *scalar* algorithms. All of the LID algorithms are composed of three main phases: an *inference extraction* phase, an *inference propagation* phase, and a *fault classification* phase. The paper introduces four algorithms based on the LID concept. These algorithms differ mainly in the inference propagation and fault classification phases, representing a trade-off between performance and diagnostic accuracy. The quality of the heuristic rules employed in the fault classification phase significantly affects the accuracy of diagnosis. Three heuristic methods of fault classification are defined, and the diagnostic performance provided by these heuristics are compared using measurement results.

Keywords: multiprocessor systems, system-level fault diagnosis, probabilistic diagnostic algorithms, generalized test invalidation, fault classification heuristics.

1. Introduction

The decreasing trend of the price to performance ratio of microprocessor components advances the development of *massively parallel* (MP) computers, that deliver significantly more computing power than single processor systems. Moreover, low cost supercomputing solutions emerge on the basis of *workstation clusters*, networked commercial off-the-shelf microcomputers controlled by a UNIX-compliant operating system such as Linux. These systems are built up of a large amount of functionally identical *processing elements* (PEs). PEs execute a part of the user application in parallel, and cooperate using a communication medium to unify the partial result in a complete solution. Still, the scientific problems solved by these computing environments are so complex, that the typical application run times fall

in the range of several days. Although modern electronic circuits have a very low permanent fault rate, the probability of an error occurrence during application execution in an MP system is significant due to the large number of components and the long continuous time of operation. Therefore, keeping the delivered system service uninterrupted by tolerating the effects of occurring errors is very important for parallel systems. This aim can be achieved by a fault tolerant architecture.

Automated fault diagnosis is an integral part of multiprocessor fault tolerance. Its task is to locate the faulty units in the system. Identified faulty units are stopped and physically or logically excluded from the set of available resources, and the computer is reconfigured to use only the fault-free system devices. This strategy is well applicable in massively parallel computers due to their significant inherent redundancy.

Parallel computers are too complex to be modelled on the lowest, physical level. In *system-level fault diagnosis* only the processing elements and their interactions are taken into consideration. The diagnostic procedure consists of the two main steps: (1) PEs execute tests on each other to detect possible errors, (2) the diagnostic algorithm determines the fault state of each PE by analyzing the collection of test results (called the *syndrome*). The tests are performed according to a predefined test arrangement (tester – tested unit relationships) and have a binary (pass/fail) outcome. The problem of syndrome analysis lies in the fact, that faulty processors may behave in an arbitrary manner. Thus, the results of tests performed by faulty units can be affected by the error and become unreliable. This effect is known as the *test invalidation*.

Existing methods for system-level fault diagnosis can be categorized into *deterministic* and *probabilistic* methods. Deterministic diagnosis algorithms guarantee the correct and complete identification of the fault set, provided that certain a priori requirements on the structure of the test arrangement and on the behaviour of the faulty units are satisfied. These requirements are usually strict and often impractical. The resulting deterministic algorithms are too complex and not efficient enough to handle large systems. Probabilistic diagnostic algorithms only attempt to provide correct diagnosis with high probability. This implies that the created diagnostic image can be either *incorrect* (fault-free processors are misdiagnosed as faulty, or vice versa) or *incomplete* (the fault state of certain processors cannot be classified). The benefits of the probabilistic approach are simpler, faster algorithms witout restrictive assumptions on the test arrangement or on the fault sets.

2. System-Level Fault Diagnosis

System-level fault diagnosis uses a simplified fault model composed of units, communication links, test connections, and test results. The system is built of a set of $u_i \in U$ units: $U = \{u_1, u_2, \ldots, u_n\}$, connected by a set of $(u_i, u_j) \in C$ communication links, where $u_i, u_j \in U$ but $i \neq j$. The units and links form the *system graph* $S = (U, C)$. The fault state f_i of the u_i unit can either be *fault-free* (denoted

by f_i^0) or *faulty* (f_i^1). Each unit may test one or more other fault-free or faulty units, but (without loss of generality) we require that only the communication links can be used for testing purposes. The test assignment is expressed in the form of a digraph $T = (U, E)$, where $E \subseteq C$ defines the set of t_{ij} test connections between units u_i and u_j. Two sets can be associated with each u_i unit:

- the set of units tested by u_i, $\Gamma(u_i) = \{u_j : t_{ij} \in E\}$, and
- the set of testers of u_i, $\Gamma^{-1}(u_i) = \{u_j : t_{ji} \in E\}$.

The union of tested and tester units is the set of neighbours $N(u_i) = \Gamma(u_i) \cup \Gamma^{-1}(u_i)$. The set of units, that are reacheable from u_i via directed edge sequences consisting of at most k edges are called the k-neighbours: $N_k(u_i)$. The cardinalities of these sets are denoted by $\nu(u_i)$ and $\nu_k(u_i)$, respectively. Edges of the T digraph or *testing graph* are labelled by the $a_{ij} \in A$ test results. Tests are simple GO/NO GO tests, they may either pass (a_{ij} takes the value 0) or fail (a_{ij} evaluates to 1). The A collection of test results is the *syndrome*.

The syndrome can be interpreted according to various *test invalidation models*. Test invalidation is the effect of the behavior of a faulty unit on a test result. For example, a faulty tester unit may produce a nondeterministic pass/fail test result, independent of the state of the tested unit. This test invalidation scheme is called the *symmetric* invalidation or PMC model (PREPARATA – METZE – CHIEN, 1967). Other test invalidation schemes are also possible. The *asymmetric* invalidation or BGM model (BARSI – GRANDONI – MAESTRINI, 1976) assumes that the probability of two identical unit failures is negligible, and the testing procedure is thorough enough to detect the difference of two fault modes. Therefore, a test on a faulty unit will always fail even if it is performed by a faulty tester processor. In both invalidation models a test result of a fault-free processor indicates the exact state of the tested unit, i.e., tests are assumed to be *complete*.

2.1. Generalized Test Invalidation

In *heterogeneous* systems consisting of various functional units, test invalidation will likely be heterogeneous as well. The *generalized test invalidation* scheme provides a unified framework to handle the differences of the invalidation models of system components (SELÉNYI, 1984). The model is described in *Table 1*. Due to the complete test assumption fault-free units always test other units correctly. Test results generated by faulty tester units can have three outcomes: always pass, always fail, or arbitrarily pass/fail independent of the fault state of the tested unit. These results correspond to the constants 0, 1, and X. Nine possible test invalidation models are encompassed by the generalized scheme, a particular model is determined by the respective $C \in \{0, 1, X\}$ and $D \in \{0, 1, X\}$ values. For example, symmetric invalidation is referred to as the T_{XX}, and asymmetric invalidation as the T_{X1} test invalidation model.

Table 1. Generalized test invalidation

Tester unit	Tested unit	Test result
fault-free	fault-free	pass
fault-free	faulty	fail
faulty	fault-free	$C \in$ {pass, fail, or arbitrary}
faulty	faulty	$D \in$ {pass, fail, or arbitrary}

The relationship between tester and tested units encapsulated by generalized invalidation can be used to derive *parameterized one-step implication rules*. One-step implications have the form of 'fault state a of unit u_i implies the fault state b of unit u_j' (denoted by $f_i^a \rightarrow f_j^b$). An implication rule is affected by three main parameters: (1) the test invalidation of the tester unit, (2) the (supposed) fault state of the tester/tested unit, and (3) the actual test outcome. The complete set of parameterized one-step implication rules derived from the general test invalidation model is shown in *Table 2*.

Table 2. One-step implication rules

Type	Tester unit		Tested unit	Test	Implication
	fault state	invalidation	fault state	result	
$\hookleftarrow$	fault-free				$f_i^0 \rightarrow f_i^0$
$\rightarrow$	fault-free			pass	$f_i^0 \rightarrow f_j^0$
$\leftarrow$		T_{1D}	fault-free	pass	$f_j^0 \rightarrow f_i^0$
$\rightarrow$	fault-free			fail	$f_i^0 \rightarrow f_j^1$
$\leftarrow$		T_{1D}, T_{XD}	fault-free	fail	$f_j^0 \rightarrow f_i^1$
$\rightsquigarrow$		T_{0D}	fault-free	fail	$f_j^0 \rightarrow f_j^1$
$\leftarrow$		T_{C0}	faulty	fail	$f_j^1 \rightarrow f_i^0$
$\rightarrow$	faulty	T_{10}, T_{X0}		fail	$f_i^1 \rightarrow f_j^0$
$\rightarrow$	faulty	T_{01}, T_{X1}		pass	$f_i^1 \rightarrow f_j^0$
$\rightsquigarrow$		T_{C0}	faulty	pass	$f_j^1 \rightarrow f_j^0$
$\rightsquigarrow$	faulty	T_{00}		fail	$f_i^1 \rightarrow f_i^0$
$\rightsquigarrow$	faulty	T_{11}		pass	$f_i^1 \rightarrow f_i^0$
$\hookleftarrow$	faulty				$f_i^1 \rightarrow f_i^1$
$\leftarrow$		T_{C0}, T_{CX}	faulty	pass	$f_j^1 \rightarrow f_i^1$
$\rightarrow$	faulty	T_{00}, T_{0X}		fail	$f_i^1 \rightarrow f_j^1$
$\rightarrow$	faulty	T_{10}, T_{1X}		pass	$f_i^1 \rightarrow f_j^1$

Type: $\hookleftarrow$ tautology, $\rightarrow$ forward, $\leftarrow$ backward, $\rightsquigarrow$ contradiction

Four types of one-step implication rules exist: tautology, forward implication, backward implication, and contradiction. A *contradiction* provides a sure implication: it expresses that either the fault-free or the faulty state of a certain unit is incompatible with the syndrome: $f_i^a \rightarrow f_i^{\neg a}$. Two one-step implications can be combined into a two-step implication using the transitive property: if $f_i^a \rightarrow f_j^b$ and $f_j^b \rightarrow f_k^c$ are two valid one-step implications, then they imply $f_i^a \rightarrow f_k^c$. The set of all one-step and multiple-step implications obtained by repeated application of the transitive property is the *transitive closure*. It contains all information that can be extracted from the syndrome. In the following section we describe how the transitive closure can be utilized in the diagnostic procedure.

Diagnostic implications can be drawn in a digraph form. The *inference graph* $I = (U, \mathcal{F}, \mathcal{P})$ is composed of the U set of units, the set of $f_i^{\{0,1\}} \in \mathcal{F}$ possible fault states and the set of $p_{ij} \in \mathcal{P}$ one-step implications, derived from the actual syndrome. In the graphical representation units, states and implications correspond to boxes, nodes and directed edges (see *Fig. 1*).

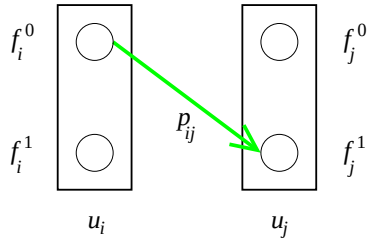


Fig. 1. Components of the inference graph

2.2. Local Information Diagnosis

The transitive closure is obtained using the implication rules derived from the generalized test invalidation model, and so it is the complete source of topology and fault set independent diagnostic information. A diagnostic algorithm based on the transitive closure has the following structure:

1. One-step diagnostic implications are extracted using the parameterized implication rules and the actual syndrome (see an example in *Fig. 2*).
2. Multiple-step implications are obtained by transitively combining one-step implications. Inference propagation may continue until all possible implication chains are expanded in full length, that is, the transitive closure is created.
3. All units involved in contradictions found in the transitive closure can be surely classified as fault-free or faulty.

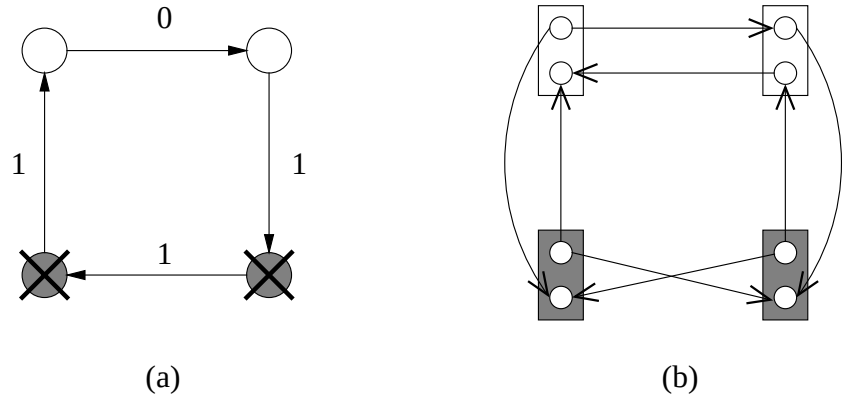


Fig. 2. Inference graph creation: (a) Example testing graph with syndrome, (b) Corresponding inference graph

4. Other units are diagnosed by a deterministic or probabilistic fault classification method.

For units whose fault state cannot be surely classified there are two possible diagnostic approaches. Deterministic algorithms assume that certain predefined conditions on the possible fault sets and on the testing assignment hold. The restrictive conditions allow the diagnostic algorithm to eliminate diagnostic uncertainty by leaving certain fault sets out of consideration. This way a one-to-one correspondence is created between the fault sets and the resulting syndromes, and the generated diagnostic image is guaranteed to be correct and complete when the requirements are justified. However, there is no information on the behaviour of the deterministic algorithms outside the valid range of their assumptions; they may produce arbitrary results. Probabilistic methods give a possibility to achieve a satisfactory diagnostic result even where the deterministic approach is useless. They employ fault classification heuristics to determine the fault state of system units. The aim of these heuristics is to estimate the most likely fault state, while they must remain simple and computationally efficient at the same time.

There are two main performance bottlenecks of the above outlined procedure. First and foremost, generating the transitive closure of a large inference graph is a computation-intensive task. The underlying idea of *local information diagnosis* (LID) is that a probabilistic algorithm can achieve high probability of diagnostic correctness without expanding the implication chains in full length. The informal explanation of this claim requires us to examine the possible fault configurations.

Two main types of fault patterns can occur in an MP system: (1) the faults are scattered throughout the system, separated from each other, and (2) the faults are located close to each other forming a group. In most practical cases both situations can be handled using just a portion of the diagnostic information (BLOUGH

– SULLIVAN – MASSON, 1989). When faults are separated, the failed test results appear locally in the syndrome. The faulty units are surrounded by fault-free tester processors, there is diagnostic information enough available to identify the faulty units. In the second case, however, diagnostic uncertainty caused by faulty units 'blocks' the propagation of the inferences. In other words, the faults constituting the group border isolate the inside of the group from the rest of the system. The implication chains do not lead into the core of the group. For this reason, any classification method can only attempt to identify the units on the fault group borders. These peripheral units are surrounded by fault-free testers similarly to separated faults, and can be reliably identified even if only a partial diagnostic information is extracted from the syndrome. The diagnosis of the fault group core will improve only little when the implication chains are calculated in full length.

The other performance bottleneck originates in the classification of those units which are not involved in a contradiction and whose fault state cannot be surely identified. Deterministic algorithms require complex methods for this task, since they must guarantee a correct and complete diagnosis (if only in a restricted set of cases). A typical requirement employed by many traditional diagnostic algorithms is a static upper bound on the number of tolerated faulty units. This diagnostic t-limit is a very serious restriction in large systems, because a significant amount of fault sets consisting of more than t faulty units are unambiguously diagnosable (SOMANI – AGARWAL – AVIS, 1987). Moreover, the amount of unambiguously diagnosable faults does not remain static (as the diagnostic t-limit suggests) but increases proportionally to the system size.

Along these guidelines the authors developed a family of probabilistic diagnostic algorithms based on the local information diagnosis methodology (BARTHA – SELÉNYI, 1996). These simple and efficient algorithms use the *generalized test invalidation* principle making them able to handle a class of heterogeneous systems. They are significantly faster than deterministic algorithms as they analyze only a portion of diagnostic information contained in the transitive closure. Several of them exploit the regular structure and low local complexity of MP systems by propagating implications only among the neighbour units. The fault classification step uses a simple heuristic rule, solely based on the collected one- and multiple-step implications independent of the number of faulty units. This further reduces the time complexity of the algorithms, and provides good diagnostic accuracy even for fault sets significantly larger than the t-limit.

3. Diagnostic Algorithms

In this section we present four local information diagnostic algorithms. They can be divided into the following three categories:

Limited inference algorithms: These algorithms use a binary matrix representation of the one- and multiple-step implications derived from the syndrome. This binary matrix (called the *inference matrix* and denoted by **M**) stores

every possible implication, i.e., the complete diagnostic information about the system. The repeated multiplication of the inference matrix transitively propagates the stored implication chains. The underlying idea of limited inference methods is to compute implication chains only in a limited, predetermined length. This way only a subset of the transitive closure is obtained. Units are classified on the basis of this incomplete diagnostic information.

Limited information algorithms: Another method of reducing the diagnostic complexity is to limit the amount of information taken into account during inference propagation. The concept of this approach is to 'cut out' the local environment of the unit under diagnosis, and perform a full transitive closure in this restricted area. Thus, the computation-intensive transitive closure computation is performed n times, but only for a small, constant size $\mathbf{M}_k(u_i)$ reduced inference matrix.

Scalar algorithms: Scalar algorithms compute and utilize only the quantity of implications supporting a given fault hypothesis. They do not keep record of the implication chains connecting the fault states. The time and space complexity of this class of algorithms is quite low, while they provide considerably good diagnostic accuracy. On the other hand, the scalar representation obviously results in the loss of diagnostic information. The relationship of non-neighbour units cannot be determined and multiple-step contradictions remain undetected. A further consequence of this information loss is that the fault classification heuristics presented in this paper are not applicable to scalar algorithms. The heuristic classification rule in these methods is included in the form of a specific weight function used in the implication sum calculation.

3.1. Limited Inference Algorithms

Limited inference algorithms process the complete set of diagnostic inferences, but propagate the implication chains only in a limited, predetermined length and classify the units on the basis of this 'partial transitive closure'.

Limited Multiplication of Inference Matrix (LMIM) algorithm. The LMIM algorithm is a simplified variant of the Selényi algorithm described in (SELÉNYI, 1984). One-step implications are collected and stored in the $2n \times 2n$ $\mathbf{M}$ inference hypermatrix. The $\mathbf{M}$ matrix consists of four $n \times n$ binary minor matrices: $\mathbf{M}^{00}$, $\mathbf{M}^{01}$, $\mathbf{M}^{10}$ and $\mathbf{M}^{11}$. The $m^{xy}[i, j]$ element of the $\mathbf{M}^{xy}$ minor matrix ($x, y \in \{0, 1\}$) equals 1 if there exists an $f_i^x \rightarrow f_j^y$ one-step implication between units u_i and u_j, otherwise it is 0. All elements in the main diagonal of the $\mathbf{M}^{00}$ and $\mathbf{M}^{11}$ minor matrices are 1, representing the tautology implications. The structure of the inference matrix is shown in *Fig. 3*.

Transitive closure can be computed by the logical closure of the $\mathbf{M}$ matrix. This is achieved by the repeated application of the $\mathbf{M}^{(k+1)} \leftarrow \mathbf{M}^{(k)} \cdot \mathbf{M}^{(k)}$ (the

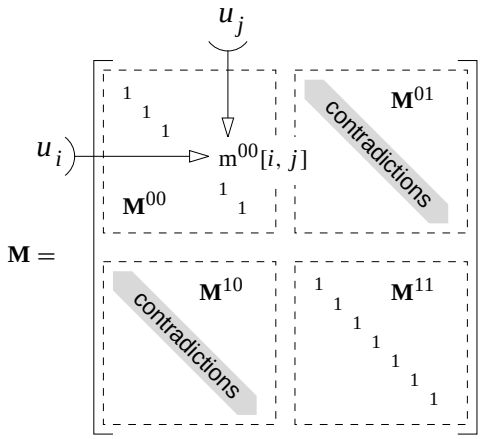


Fig. 3. Structure of the inference matrix

contents of **M** in the $(k+1)$-th step is the square of **M** in the k-th step) iteration until no new implications appear in the subsequent steps. However, in the LMIM algorithm the **M** matrix is raised only to a small, constant power (hence the name of the method), i.e., the iteration is executed only a few, constant times. Thus, the matrix will contain only a subset of the diagnostic inferences included in the transitive closure. Non-zero elements in the main diagonal of the $\mathbf{M}^{01}$ and $\mathbf{M}^{10}$ minor matrices signify contradictions. For example, if $m^{01}[i, i]$ equals to 1, then the $f_i^0 \rightarrow f_i^1$ implication holds, that is unit u_i is surely faulty. Similarly, all u_j units corresponding to the non-zero $m^{01}[j, j]$ and $m^{10}[j, j]$ elements can be surely classified. For other units a heuristic fault classification rule, like those described in Section 4, must be used to determine their fault state.

Distribution of Inference Lists (DIL) algorithm. The DIL algorithm uses the same representation as the LMIM method. However, it has a better time complexity, because it exploits the properties of regular topologies. These topologies have a constant, relatively low connectivity of nodes, i.e., each processor has only a few neighbours. In such systems, the matrix multiplication used to compute the transitive closure performs a lot of redundant operations, especially in the first few iterations. For example, the $m^{00}[i, j]$ element of the $\mathbf{M}^{00}$ minor matrix is computed as

$$m^{00}[i, j] \leftarrow \sum_{k=1}^{n} m^{00}[i, k] \cdot m^{00}[k, j] + m^{01}[i, k] \cdot m^{10}[k, j].$$

In the first iteration only one-step implications exist and these can be combined into two-step implications only at neighbour nodes, therefore each multiplication where $u_k \notin N(u_i) \cap N(u_j)$ is surplus. The idea of the DIL algorithm is to avoid these redundant operations by propagating inference chains only among neighbouring

nodes.

For every u_i unit four binary vectors: $\mathbf{m}^{00}[i]$, $\mathbf{m}^{01}[i]$, $\mathbf{m}^{10}[i]$, and $\mathbf{m}^{11}[i]$ contain the set of one-step implications according to the local view of u_i (these are the row vectors of the respective minor matrices in $\mathbf{M}$). In every iteration the set of implications is propagated locally at each u_i unit by unifying the appropriate row vectors of u_i with the row vectors of its neighbours (see *Fig. 4*). At the end of the process, sure classification is indicated by the i-th components of the $\mathbf{m}^{01}[i]$ and $\mathbf{m}^{10}[i]$ vectors. Unclassified processors are diagnosed with the help of heuristic fault classification rules, similarly to the LMIM algorithm.

Algorithm DIL
{ initialization }
for each $u_i \in U$ **do**
  $\mathbf{m}^{00}[i] \Leftarrow \{u_j : \exists p_{ji},\ f_j^0 \rightarrow f_i^0\}$
  $\mathbf{m}^{01}[i] \Leftarrow \{u_j : \exists p_{ji},\ f_j^0 \rightarrow f_i^1\}$
  $\mathbf{m}^{10}[i] \Leftarrow \{u_j : \exists p_{ji},\ f_j^1 \rightarrow f_i^0\}$
  $\mathbf{m}^{11}[i] \Leftarrow \{u_j : \exists p_{ji},\ f_j^1 \rightarrow f_i^1\}$
end for
{ update $\mathbf{m}^{00}, \mathbf{m}^{01}, \mathbf{m}^{10}, \mathbf{m}^{11}$ vectors }
for each iteration **do**
  for each $u_i \in U$ **do**
    for each $p_{ij} \in P[i]$ **do**
      if $p_{ij} : f_i^0 \rightarrow f_j^0$ **then**
        $\mathbf{m}^{00}[j] \Leftarrow \mathbf{m}^{00}[j] \cup \mathbf{m}^{00}[i]$
        $\mathbf{m}^{10}[j] \Leftarrow \mathbf{m}^{10}[j] \cup \mathbf{m}^{10}[i]$
      else if $p_{ij} : f_i^0 \rightarrow f_j^1$ **then**
        $\mathbf{m}^{01}[j] \Leftarrow \mathbf{m}^{01}[j] \cup \mathbf{m}^{00}[i]$
        $\mathbf{m}^{11}[j] \Leftarrow \mathbf{m}^{11}[j] \cup \mathbf{m}^{10}[i]$
      else if $p_{ij} : f_i^1 \rightarrow f_j^0$ **then**
        $\mathbf{m}^{00}[j] \Leftarrow \mathbf{m}^{00}[j] \cup \mathbf{m}^{01}[i]$
        $\mathbf{m}^{10}[j] \Leftarrow \mathbf{m}^{10}[j] \cup \mathbf{m}^{11}[i]$
      else if $p_{ij} : f_i^1 \rightarrow f_j^1$ **then**
        $\mathbf{m}^{01}[j] \Leftarrow \mathbf{m}^{01}[j] \cup \mathbf{m}^{01}[i]$
        $\mathbf{m}^{11}[j] \Leftarrow \mathbf{m}^{11}[j] \cup \mathbf{m}^{11}[i]$
    end for
  end for
end for

Fig. 4. Pseudo code of DIL algorithm

3.2. Limited information

The limited information approach uses a different concept to utilize regularity. Instead of limiting the length of implication chains, this approach limits the *area* of inference propagation.

Local Transitive Closure (LTC) algorithm. The LTC algorithm calculates a complete transitive closure, but only in a small local environment of the diagnosed units. The diagnosis of every u_i unit begins with the creation of an additional $2v_k \times 2v_k$ hypermatrix. This reduced $\mathbf{M}_k(u_i)$ matrix includes only the u_i processor and its k-neighbours, and the one-step implications among them. In the second step the transitive closure of the reduced matrix is computed, and the resulting implications are copied back to the $\mathbf{M}$ matrix. Since each $\mathbf{M}_k(u_i)$ matrix contains paths of at most k length, the final $\mathbf{M}$ matrix includes implications of at most k-step length. Space and time complexity is reduced due to the smaller size of the reduced matrix. The resulting $\mathbf{M}$ matrix is used for sure and heuristic fault classification identically to the LMIM and DIL algorithms.

3.3. Scalar Algorithms

The decision factor in the case of scalar algorithms is the number of implications supporting a given fault hypothesis. These algorithms achieve a significant time and space complexity reduction by not representing the implication chains themselves. This is advantageous from the efficiency viewpoint, but obviously results in further loss of diagnostic information. As a consequence, the relationship of non-neighbour units cannot be determined and higher order contradictions remain undetected. To compensate for this effect additional information – like a weight function – must be included in these methods.

Count Inference Paths (CIP) algorithms. The CIP algorithm estimates the likelihood of a fault hypothesis as the number of implications supporting it. For this purpose the algorithm maintains two counters $\Sigma^0[i]$ and $\Sigma^1[i]$ at each u_i unit, corresponding to the weighted number of edges in implication chains ending in the fault-free state f_i^0 and the faulty state f_i^1 of u_i in the inference graph. The $\Sigma^0[i]$ and $\Sigma^1[i]$ numbers are calculated by an iterative algorithm. The initial value of the counters is set using the one-step implications collected from the syndrome. One-step contradictions are detected during the initialization process, and they are used to surely classify the affected units. For processors without sure classification the number of the implications supporting both fault states of the unit are counted. The algorithm is outlined in *Fig. 5*.

In the $(k+1)$-th iteration step the counters are increased appropriately by the number of paths added to the neighbour units' counters in the k-th step. This *counter update* mechanism is described in *Fig. 6*. The value of the added paths is multiplied

```
Algorithm CIP
{ initialization }
for each u_i ∈ U do
   for each p_ij ∈ P, u_j ∈ N(u_i) do
     if p_ij is contradiction then
       surely classify u_i
     else P[i] ⇐ P[i] ∪ p_ij
   end for
end for
{ count inference paths }
for each iteration do
   for each u_i ∈ U do
     for each p_ij ∈ P[i] do
       if u_i is surely classified then
         surely classify u_j
       else Update Σ^0[i], Σ^1[i]
     end for
   end for
end for
```

Fig. 5. Pseudo code of CIP algorithm

by the $W(p_{ij})$ weight of the p_{ij} implication connecting the unit and its neighbour. The W weight function is set to compensate for the effect of incorrect implications corresponding to faulty units. In the subsequent iteration steps all implication chains of length $2, 3, \ldots, k$ are added to the calculation. The set of surely classified units is also extended using the implications drawn from the already surely classified fault states. After the given number of iterations the remaining unclassified units are diagnosed as faulty if $\Sigma^1[i] > \Sigma^0[i]$, otherwise they are assumed to be fault-free.

There are two variants of the CIP algorithm. The above outlined method is referred to as CIP-2, because it uses two counters to calculate the weighted number of edges in the $f_j^x \rightarrow f_i^0$ and $f_j^y \rightarrow f_i^1$ (where $x, y \in \{0, 1\}$ and $j = 1, 2, \ldots, n$) type of implication chains. The refined CIP-4 variant maintains four counters called $\Sigma^{00}[i]$, $\Sigma^{01}[i]$, $\Sigma^{10}[i]$, and $\Sigma^{11}[i]$. Now it is possible to separately calculate the $f_j^0 \rightarrow f_i^0$, $f_j^0 \rightarrow f_i^1$, $f_j^1 \rightarrow f_i^0$, and $f_j^1 \rightarrow f_i^1$ type of implication chains. This reduces the number of loops in the considered inference paths, which results in a more exact estimation of the implication number. The CIP-4 algorithm is completely identical to the CIP-2, only the Update routine must be modified to handle the four counters. The fault classification step is also different: the remaining (not surely classified) units are diagnosed as faulty if $\Sigma^{01}[i] > \Sigma^{00}[i]$, other units are fault-free. Note, that in the case of both scalar algorithms the fault classification heuristics introduced in Section 4 are not applicable, 'fine-tuning' of these methods can be achieved by modifying the elements of the W weight function.

Algorithm Update $\Sigma^0[i]$, $\Sigma^1[i]$
{ initialization }
for each $u_i \in U$ **do**
  $\Sigma^0[i] \Leftarrow 0, \mathbf{s}^0[i] \Leftarrow 0, \mathrm{t}^0[i] \Leftarrow -1$
  $\Sigma^1[i] \Leftarrow 0, \mathbf{s}^1[i] \Leftarrow 0, \mathrm{t}^1[i] \Leftarrow -1$
end for
{ update $\Sigma^0[i]$, $\Sigma^1[i]$ counters }
for each iteration **do**
  for each $u_i \in U$ **do**
    for each $p_{ij} \in P[i]$ **do**
      if $p_{ij} : f_i^0 \rightarrow f_j^0$ **then**
        $\Sigma^0[j] \Leftarrow \Sigma^0[j] + W(p_{ij})(s^0[i] - \mathrm{t}^0[i])$
      else if $p_{ij} : f_i^0 \rightarrow f_j^1$ **then**
        $\Sigma^1[j] \Leftarrow \Sigma^1[j] + W(p_{ij})(s^0[i] - \mathrm{t}^0[i])$
      else if $p_{ij} : f_i^1 \rightarrow f_j^0$ **then**
        $\Sigma^0[j] \Leftarrow \Sigma^0[j] + W(p_{ij})(s^1[i] - \mathrm{t}^1[i])$
      else if $p_{ij} : f_i^1 \rightarrow f_j^1$ **then**
        $\Sigma^1[j] \Leftarrow \Sigma^1[j] + W(p_{ij})(s^1[i] - \mathrm{t}^1[i])$
    end for
  end for
  for each $u_i \in U$ do
    $\mathrm{t}^0[i] \Leftarrow s^0[i], s^0[i] \Leftarrow \Sigma^0[i]$
    $\mathrm{t}^1[i] \Leftarrow s^1[i], s^1[i] \Leftarrow \Sigma^1[i]$
  end for
end for

Fig. 6. Counter update mechanism of CIP algorithm

4. Fault Classification Heuristics

The limited inference type of LID algorithms: LMIM, DIL, and LTC methods described in (BARTHA – SELÉNYI, 1996) are all algorithmically different, but diagnostically equivalent methods of generating the partial implication set, which is used for fault classification. However, the evaluation of the partial implication set is also a complex problem. In probabilistic methods, such as the LID algorithms subject to this paper, use heuristic fault classification rules to transform the implications into a system-wide diagnostic image. Our previous paper (BARTHA – SELÉNYI, 1996) used only one of the many possible heuristic rules. As the quality of the employed fault classification heuristic significantly affects diagnostic accuracy, without comparison it is hard to value the performance of probabilistic algorithms.

This paper defines three new fault classification heuristics developed on the

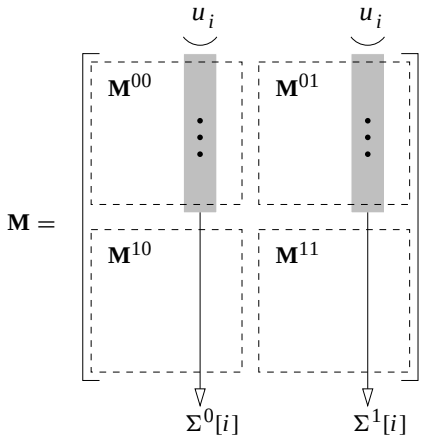


Fig. 7. Calculation of the $\Sigma^0[i]$ and $\Sigma^1[i]$ values

basis of successful existing methods. The developed heuristic methods are called *Majority* (this was the heuristic used in our previous papers), *Election*, and *Clique*. They are all based on the assumption that the number of faulty units does not exceed the number of fault-free units in the system. However, each heuristic uses this assumption differently. The section presents the description of the heuristic fault classification methods, then the diagnostic performances of these methods are compared using measurement results.

Majority heuristic. The idea of Majority heuristics is simple: since only the fault-free units produce reliable test results, only the implications from the fault-free states (stored in the $\mathbf{M}^{00}$ and $\mathbf{M}^{01}$ minor matrices) should be considered. The $f_j^0 \to f_i^0$ and $f_j^0 \to f_i^1$ implications ($j = 1, 2, \ldots, n$) can be interpreted as votes for the fault-free and faulty state of the u_i unit, respectively. The fault classification can be made as a majority decision between the votes for the fault-free/faulty state. The sum of votes, i.e., the number of $f_j^0 \to f_i^0$ and $f_j^0 \to f_i^1$ implications can be calculated by counting the non-zero elements stored in the i-th column of the $\mathbf{M}^{00}$ and $\mathbf{M}^{01}$ matrices (see *Fig. 7*). Comparing the two sums $\Sigma^0[i] = \sum_j m^{01}[j, i]$ and $\Sigma^1[i] = \sum_j m^{00}[j, i]$, the unit is diagnosed as faulty if $\Sigma^0[i] < \Sigma^1[i]$, otherwise it is fault-free. (In a system with more fault-free than faulty units and a completely connected testing graph the Majority heuristic would always correctly classify the fault state of all units.)

Election heuristic. The Election heuristic applies the mechanism of the *Count Failed Tests* (CFT) algorithm developed by the authors in (BARTHA – SELÉNYI, 1997) and the Dahbura et al algorithm (DAHBURA – SABNANI – KING, 1987) to limited inference methods. The idea is to identify the faulty units sequentially one-by-one. Units are ranked according to the likelihood of being faulty for the

purpose of selection, and in each identification step the unit with the highest ranking is diagnosed as faulty. The diagnostic uncertainty is decreased by removing the useless and confusing implications originating in the actually located faulty unit. Naturally, ranks must be recomputed each time the set of diagnostic implications is changed. The procedure is outlined in *Fig. 8.*

```
Election heuristic
{ initialization }
for each u_i ∈ U do
    LF[i] ⇐ Σ^1[i] − Σ^0[i]
    NLF[i] ⇐ Σ_j LF[j], u_j ∈ Γ^{-1}(u_i)
end for
{ election }
ϒ ⇐ U, Φ ⇐ ∅
while ∃j, k, m^{01}[j, k] ≠ 0 do
    find u_m with:
        maximum LF[m], and
        minimum NLF[m]
    Φ ⇐ Φ ∪ u_m
    ϒ ⇐ ϒ − u_m
    ∀u_i ∈ U, m^{01}[m, i] ⇐ 0
    recalculate LF[i] and NLF[i]
end while { classification }
∀u_i ∈ Φ, u_i is faulty
∀u_j ∈ ϒ, u_j is fault-free
```

Fig. 8. Pseudo code of the Election heuristic

The likelihood $LF[i]$ of the faulty state of unit u_i is estimated as $LF[i] = \Sigma^1[i] - \Sigma^0[i]$. For ranking units with identical LF values, the likelihood $NLF[i]$ of the faulty state of units testing u_i is also counted: $NLF[i] = \sum_j LF[j]$ for each $u_j \in \Gamma^{-1}(u_i)$. The units are sorted to find the unit u_m most likely to be faulty with the most reliable testers, i.e., having the maximum $LF[m]$ and the minimum $NLF[m]$ values. The u_m unit is then added to the Φ set of faulty units. The unit and its $f_m^0 \rightarrow f_i^1$ implications are removed from the **M** inference matrix, and the entire selection procedure starts again. When there are no more implications in the $\mathbf{M}^{01}$ minor matrix the remaining units are classified as fault-free.

Clique heuristic. The Clique heuristic is based on the diagnostic algorithm by Maestrini et al. (MAESTRINI – SANTI, 1995). The concept is similar to the Majority heuristic: if some fault-free units could be located, then their test results could reliably identify the fault state of other units. However, instead of comparing the feasibility of the fault-free/faulty states individually, the algorithm tries to group the units into two separate cliques. Since in the worst case each clique can contain only one element, clique generation must be done on a per unit basis. The *friendly*

clique $\mathbf{C}^0[i]$ of unit u_i contains units with a fault state identical to u_i (they are either all fault-free or faulty), while the *foe* clique $\mathbf{C}^1[i]$ groups units with a fault state opposite to u_i (if u_i is fault-free, then they can only be faulty, and vice versa). Obviously, the clique sets of neighbour fault-free units are identical.

Clique heuristic
{ initialization }
for each $u_i \in U$ **do**
  $\mathbf{C}^0[i] \Leftarrow \mathbf{C}^0[i] \cup u_j$, if $\mathbf{m}^{00}[i, j] \neq 0$
  $\mathbf{C}^1[i] \Leftarrow \mathbf{C}^1[i] \cup u_j$, if $\mathbf{m}^{01}[i, j] \neq 0$
end for
{ clique closure }
for each $u_i \in U$ **do**
  for each $u_j \in \mathbf{C}^0[i]$ **do**
    $\mathbf{C}^0[i] \Leftarrow \mathbf{C}^0[i] \cup \mathbf{C}^0[j]$
    $\mathbf{C}^1[i] \Leftarrow \mathbf{C}^1[i] \cup \mathbf{C}^1[j]$
  end for
end for
{ classification }
find u_m with:
  maximum $|\mathbf{C}^0[m]|$, and
  minimum $|\mathbf{C}^1[m]|$
$\forall u_i \in \mathbf{C}^0[m]$, u_i is fault-free
$\forall u_j \in \mathbf{C}^1[m]$, u_j is faulty
other units are unknown

Fig. 9. Pseudo code of the Clique heuristic

Cliques are initialized using the implications in the $\mathbf{M}^{00}$ and $\mathbf{M}^{01}$ minor matrices. Clique membership is then extended using the following two rules: (1) 'my friend's friend is my friend', and (2) 'my friends foe is my foe'. The other two possible rules: (3) 'my foe's friend is my foe', and (4) 'my foe's foe is my friend' are not used, since they could lead to inconsistent cliques due to faulty units. Then, the algorithm searches for the u_m unit with a maximum cardinality $\mathbf{C}^0[m]$ set and minimum cardinality $\mathbf{C}^1[m]$ set. The units belonging to the $\mathbf{C}^0[m]$ set are called the *Fault-Free Core*, they are classified as fault-free. Units in the $\mathbf{C}^1[m]$ set are diagnosed as faulty. Since some parts of the system can be separated by faulty units, there can be units neither contained in the $\mathbf{C}^0[m]$ set nor in the $\mathbf{C}^1[m]$ set. These units get the *unknown* classification, i.e., the Clique heuristic may lead to an *incomplete* diagnostic image.

5. Performance

The presented methods were compared using measurements in a dedicated simulation environment. The simulation examined many characteristics of the algorithms, including the effect of fault percentage, type of fault patterns, number of iterations, and system topology on diagnostic performance. The simulated system had a 2-dimensional toroidal mesh topology containing 12×12 processing elements. Random fault patterns of various fault probabilities were injected in the system, and after executing the diagnostic algorithm in 2 iterations statistical data of the diagnostic accuracy were collected in 512 subsequent simulation rounds. Although several homogeneous and heterogeneous invalidation schemes were involved in the simulation, here we can present only the results for most common symmetric (PMC) test invalidation model due to space constraints.

Table 3 summarizes the main characteristics of the presented algorithms:

Time complexity: The CIP and DIL algorithms process the information locally, their complexity depends on the $\nu = \max_i \nu(u_i)$ number of neighbouring units. The complexity of the LTC algorithm depends only on the $\nu_k = \max_i \nu_k(u_i)$ number of k-neighbors examined. Note, that the ν and ν_k values are constant in the function of system size, so the CIP, DIL, and LTC algorithms have essentially linear time complexity. The only exception is the LMIM algorithm, its relatively low performance results from the redundant matrix operations which make it topology-independent.

Space complexity: The values reflect the amount of data manipulated for diagnostic purpose, not including the syndrome. For large systems, and in case of small numbers of considered k-neighbours the LTC algorithm has the best space complexity.

Number of diagnostic implications: This row of the table shows the size of the transitively propagated implication set after k iteration steps. In this respect LMIM is far more effective than the other algorithms, since it increases the length of considered implication chains exponentially, and yet it has linear time complexity respective to the number of iterations. The LTC algorithm is the worst among LID methods in this respect: it can only linearly increase the amount of diagnostic information at the cost of cubical time complexity.

Table 3. Characteristics of the presented algorithms

Algorithm Type	CIP scalar	LMIM limited inference	DIL limited inference	LTC limited information
Time complexity	$O(n\nu)$	$O(n^3)$	$O(n\nu)$	$O(n\nu_k^3)$
Space complexity	$O(n)$	$O(n^2)$	$O(n^2)$	$O(\nu_k^2)$
Implication set	$O(k)$	$O(2^k)$	$O(k)$	$O(k)$
Propagation	local	global	local	local

The effect of the fault set size on diagnosis accuracy is shown in *Fig. 10*. The figure shows the average number of incorrectly diagnosed units for various fault probabilities, in the percentage of the system size. The LMIM, DIL, and LTC algorithms are not drawn separately, since they produce identical results (as expected) using the same fault classification heuristic. Therefore, only the two CIP variants and the three developed heuristic methods are presented. It can be seen, that diagnostic accuracy strongly drops as the ratio of faulty units reaches near 50 percent of the system size. However, the accuracy is very good (less than 0.2 percent of the total 144 processors) in the practically relevant 0–25 fault percent range.

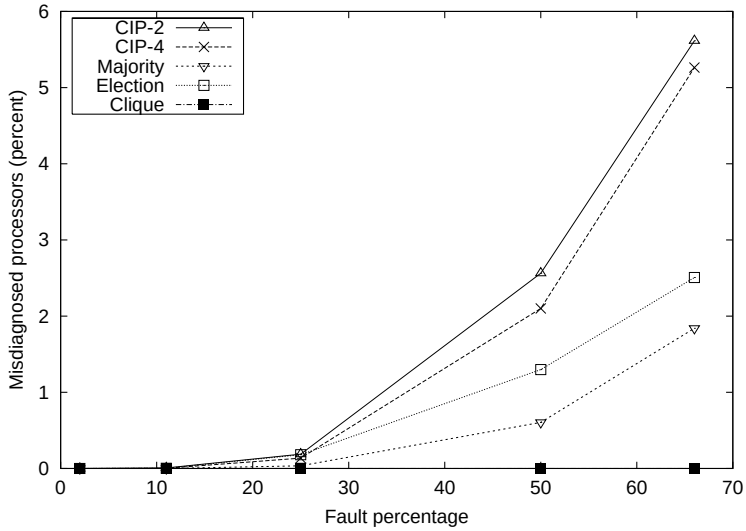


Fig. 10. Average misdiagnosed processors versus fault percentage

The worst case diagnostic measures of the three fault classification heuristics are summarized separately in *Table 4*. The first two columns contain the number of faults injected in the system and the percentage of faulty units. For the Majority and Election heuristics the number of simulation rounds with incorrect diagnosis (MR), and the maximum number of incorrectly classified fault-free (MM0), and faulty (MM1) units per round are presented. According to the results the Majority heuristic gives a better overall performance than the Election heuristic, although the latter is less prone to misdiagnose a fault-free unit as a faulty unit. The Clique heuristic did not make any diagnostic mistakes, therefore the number of simulation rounds with incomplete diagnosis (IR) and the maximum number of unknown fault-free units (UM0), and faulty (UM1) units per round is given instead. Clearly, the number of unknown units considerably exceeds the total amount of units misdiagnosed by the

other two methods, this is the price of the accurate diagnostic performance of the Clique heuristic.

Table 4. Diagnosis accuracy versus number of faults

Faults		Majority MR	Majority MM0 / MM1	Election MR	Election MM0 / MM1	Clique IR	Clique UM0 / UM1
4	(2.7%)	0	0 / 0	0	0 / 0	0	0 / 0
16	(11%)	0	0 / 0	0	0 / 0	10	1 / 0
36	(25%)	13	1 / 1	106	0 / 2	171	10 / 3
72	(50%)	222	5 / 6	463	0 / 8	510	47 / 11
96	(66%)	454	6 / 8	507	2 / 12	512	43 / 20

To illustrate the tendency of diagnostic accuracy as the system size increases, we varied the number of units in the simulated two-dimensional toroidal mesh topology. Five systems were measured, having respectively 6×6, 8×8, 12×12, 16×16, and 18×18 units. In each system 50 percent of the processors were faulty. The results, shown in *Fig. 11*, represent the average number of incorrectly diagnosed units in the percentage of the system size. The figure well indicates the expected convergence of the diagnostic accuracy to a constant value.

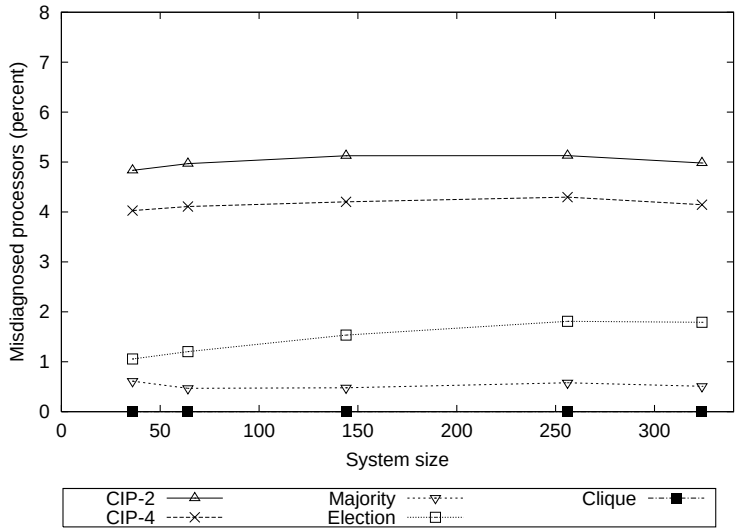
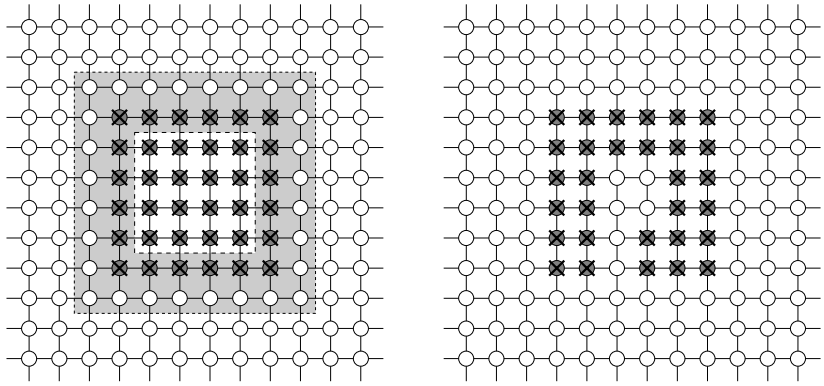


Fig. 11. Average misdiagnosed processors versus system size

For analyzing group faults we used the fault patterns shown in *Fig. 12.* The measurement results can be seen in *Table 5.* Diagnostic mistakes occurred only

for isolated faulty units. To prove this, in the case of Pattern (a) we also examined selectively the border region indicated by grey background. None of the units in the border area have been misdiagnosed. Also note, that the maximum number of incorrectly diagnosed faulty units is surprisingly small compared to the fault set size. This is due to the many sure classifications provided by the detected contradictions.



(a) (b)

Fig. 12. Fault patterns to examine diagnosis on fault group borders

Table 5. Diagnosis accuracy in group fault patterns

Pattern		**Majority** MR	MM0 / MM1	**Election** MR	MM0 / MM1	**Clique** IR	UM0 / UM1
(a)	1-step	478	0 / 9	444	0 / 7	406	0 / 6
	2-step	376	0 / 6	415	0 / 6	365	0 / 6
	4-step	358	0 / 5	395	0 / 6	361	0 / 6
(b)	1-step	408	0 / 8	355	0 / 5	139	0 / 2
	2-step	117	0 / 2	325	0 / 4	84	0 / 2
	4-step	78	0 / 2	259	0 / 4	84	0 / 2

As *Table 5* shows, inference propagation (increasing the length of implication chains) improves the diagnostic performance. *Fig. 13* presents this effect in the case of randomly injected fault patterns consisting of 36 faulty units. The number of simulation rounds with incorrect diagnosis is shown in the function of inference propagation iterations. Recall, that the length of implication chains doubles in each iteration, i.e., numbers 1, 2, 3, and 4 correspond to one-, two-, four-, and eight-step implications. The results justify our assumption: in the simulated system for random fault sets subsequent iterations improve diagnostic accuracy less and less.

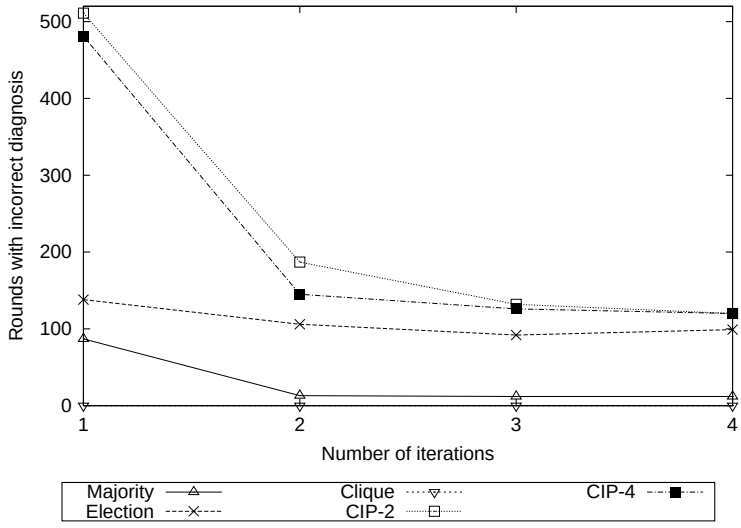


Fig. 13. The effect of inference propagation

We also examined the effect of system topology on diagnostic accuracy. Three regular interconnection topologies were simulated as illustrated in *Fig. 14*: (a) hexagonal toroidal grid with three connections, (b) 2-dimensional toroidal mesh with four connections, and (c) triangular toroidal grid with six connections. *Fig. 15* plots the number of simulation rounds with incorrect diagnosis in the function of connectivity. Random fault patterns consisting of 36 and 72 faulty units were injected in the system. As it can be seen, all of the heuristics perform better regardless of fault set size as the number of connections increases. In a completely connected system each heuristic would provide a correct and complete classification.

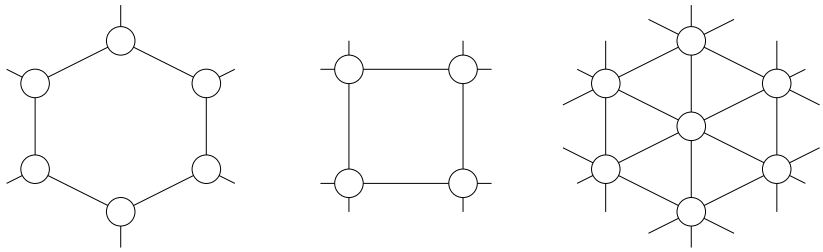


Fig. 14. Simulated system topologies

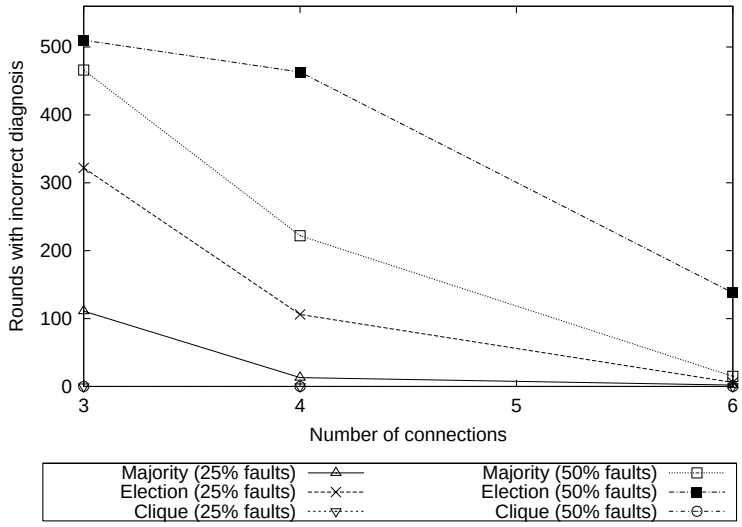


Fig. 15. The effect of system topology

6. Conclusions

This paper describes the concept of local information diagnosis, a novel approach to efficient probabilistic system-level fault diagnosis of massively parallel systems. The LID method uses a generalized test invalidation model to comply with heterogeneous structures and converts the syndrome into a topology and invalidation independent implication set. The paper demonstrated the legitimacy of the limited inference approach: it is possible to achieve high diagnostic accuracy even using only a portion of diagnostic information contained in the transitive closure by evaluating the implication chains in the inference graph only in a limited length.

Three different fault classification heuristics were presented. These heuristics adopt ideas from existing successful algorithms to the LID framework. The main characteristics of the heuristics were compared by simulation and measurements. The Majority and Election heuristics have similar diagnostic performance and complexity. They provide a complete diagnostic image, but make a low amount of diagnostic mistakes in the case of large fault sets. The Clique heuristic produces correct diagnosis even for many faulty units, but as a disadvantage more units remain unknown than misdiagnosed by the other two methods. We are currently working on further measurements on other characteristics of the LID algorithms to get more detailed information about their performance and limitations; as well as on the theoretical analysis of the practical and asymptotic behaviour of the algorithms.

References

[1] BARSI, F. – GRANDONI, F. – MAESTRINI, P.: A Theory of Diagnosability of Digital Systems, *IEEE Trans. Computers,* **C-25(6)**, (1976) pp. 585–593.

[2] BARTHA, T. – SELÉNYI, E.: Efficient Algorithms for System-Level Diagnosis of Multiprocessors Using Local Information. *Proc. of the DAPSYS '96 Workshop on Distributed and Parallel Systems,* Miskolc, (1996) pp. 183–190.

[3] BARTHA, T. – SELÉNYI, E. . Probabilistic System-Level Fault Diagnostic Algorithms for Multiprocessors. *Parallel Computing,* **22**,(1997) pp. 1807–1821.

[4] BLOUGH, D. – SULLIVAN, G. – MASSON, G.. Fault Diagnosis for Sparsely Interconnected Multiprocessor Systems, *19th Int. IEEE Symp. on Fault-Tolerant Computing,* IEEE Computer Society, (1989) pp. 62–69.

[5] DAHBURA, A. – SABNANI, K. – KING, L.: The Comparison Approach to Multiprocessor Fault Diagnosis. *IEEE Trans. Computers,* **C-36(3)**, (1987) pp. 373–378.

[6] MAESTRINI, P. – SANTI, P.: Self Diagnosis of Processor Arrays Using a Comparison Model. *Symposium on Reliable Distributed Systems (SRDS '95),* IEEE Computer Society Press, Los Alamitos, Ca., USA, (1995) pp. 218–228.

[7] PREPARATA, F. – METZE, G. – CHIEN, R.: On the Connection Assignment Problem of Diagnosable Systems, *IEEE Trans. Electronic Computers,* **EC-16(6)**, (1967) pp. 848–854.

[8] SELÉNYI, E.: Generalization of System-Level Diagnosis. D. Sc. thesis, Hungarian Academy of Sciences, Budapest (1984).

[9] SOMANI, A. K. – AGARWAL, V. K. – AVIS, D.: Generalized Theory for System Level Diagnosis, *IEEE Trans. Computers,* **C-36(5)**, (1987) pp. 538–546.