2-SATISFIABILITY AND DIAGNOSING FAULTY PROCESSORS IN MASSIVELY PARALLEL COMPUTING SYSTEMS

ANSUMAN BAGCHI, BRIGITTE SERVATIUS, AND WEIGENG SHI

ABSTRACT. A fault diagnosis model for multiprocessor computers is proposed. Under normal operating mode each processor executes its own data. When an error occurs, the system is switched to the diagnostic mode. Previous input data for each processor is shifted to a different unit, to obtain a set of comparison results. We show that analysis of the test data to diagnose or locate faulty processors is equivalent to a 2-satisfiability problem. Under the assumption that discrepancy in a comparison result occurs if and only if at least one of the processors (being compared) is faulty, we prove that all the faulty processors can be diagnosed in $O(n^2)$ time, where n denotes the number of processors in the system.

1. INTRODUCTION

In a massively parallel computing system, such as CM-2, thousands of processors work on a problem simultaneously in teraflop speeds and some problems may require hours of computing. Since reliability is a critical issue, error detection and correction mechanisms should become mandatory, and fault-diagnostic system designs and redundancy checking should be used more often.

In multi-processor computers there are two different system modes, a *normal operation mode* and a *diagnostic mode*. Usually, the system runs in a normal mode and every processor executes its own data assigned by the system or sent from other processors. If an error occurs, the system is switched to a diagnostic mode, where the computation may be restarted from a checkpointed state, i.e., a state for which the output for each processor is known to be correct. In a diagnostic mode, diagnostic tests are performed to locate (diagnose) the failed processors. Upon replacement of these faulty processors the system restarts in its normal operation mode. The entire process is known as *system-level fault diagnosis* [2, 5, 6, 7, 8, 10]. Instead of the checkpoint, this approach can also be used constantly at every step of a computation to provide redundant computation. Such a procedure can be implemented by either software or hardware.

A diagnosis mechanism, introduced by Preparata, Metze and Chien [10], compares one processor against another. Distinct processors are forced to carry out an identical set of computations, at the end of which the results are compared. Thus a fault diagnosis approach is completely characterized by the (i) *testing procedure*: which includes the selection of processor-pairs for comparison, and by the (ii) *analysis process*: the method of interpreting the comparison results to locate all (or some of) the faulty processors. Certain system diagnosis methods rely on the

Disc. Appl. Math. 60, 25–37, (1995).

specific architecture of the system, where pairwise comparisons are allowed only for processors *directly* connected by hardware links (see [5], for example). In this work, we assume that each processor may be compared with any other. This may be achieved in systems where there is a central *master* processor which acts as the *comparator* [6] and performs the entire process of system diagnosis. For example, the "front-end computer" in the CM-2 machine directs and controls each processor within the parallel processing unit [3]; so it can compare arbitrarily chosen pairs of processors. In the so called "decentralized" environments such as *distributed* multiprocessor systems [5], our assumption is valid if each processor is capable of comparing itself with any other processor.

In this paper we present a new system diagnosis model (introduced in [11]), where the emphasis will be on specifying the choice of processor-pairs to be compared and on devising a simple algorithm to analyze the data for diagnosis of the entire set of faulty processors. In addition to diagnosing the failed processors, the method can also be used to detect errors in the entire system (thereby prompting it to switch to the diagnostic mode).

2. Diagnosing Faulty Processors

2.1. **The Diagnostic Model.** We will use an n element array, called *processor array* P, to describe n processors in parallel, where p_i denotes the ith processor of the system:

$$P = \{p_0,\ p_1,\ p_2,\ \dots\ p_i,\ \dots\ p_{n-1}\}.$$

In our model, under the diagnostic mode all data scheduled to be sent to each processor p_i in the normal mode will be re-directed to the processor p_{i+m}, where m is an integer such that $1 \le m \le n-1$. Here and in the following '+' ('$-$') denotes addition (subtraction) modulo n. So, the system or a processor shifts (or re-assigns) the previous data which ran on p_i in the normal mode as current data for the processor p_{i+m} in the diagnostic mode. The actual choice of m is largely user dependent, however a systematic choice of m leading to exhaustive fault-diagnosis will be discussed later.

The process described above is called a *single m-shift fault diagnostic approach.* A system using such an approach is called a *single m-shift system.*

Let r_i be the output of processor p_i. Then, we call

$$R = \{r_0,\ r_1,\ r_2,\ \dots\ r_i,\ \dots\ r_{n-1}\},$$

the *output array* corresponding to P.

When the input data are shifted from processor p_i to processor p_{i+m}, we obtain a different output array, R^m. R^m is called the *m-shift output array* corresponding to P :

$$R^m = \{r_0^m,\ r_1^m,\ r_2^m,\ \dots\ r_i^m,\ \dots\ r_{n-1}^m\}.$$

By definition, $R = R^0$. If there is no error, r_{i+m}^m and r_i are equal. We may assume that the same failure occurs in two different processors with very low probability. That is, if $r_i = r_{i+m}^m$, then with very high probability there is no error in r_i and r_{i+m}^m; if $r_i \neq r_{i+m}^m$ then either r_i or r_{i+m}^m is incorrect or both of them are incorrect. Following this assumption, the comparison elements can now be computed. Define the *comparison array* C as:

$$C = \{c_0,\ c_1,\ c_2,\ \dots\ c_i,\ \dots\ c_{n-1}\},$$

where $c_i = 0$, if $r_i = r^m_{i+m}$ and $c_i = 1$, if $r_i \neq r^m_{i+m}$.

We will use x_i ("0" or "1") to indicate the status of a processor p_i:

: $x_i = 0$, means that processor p_i is fault-free;
: $x_i = 1$, means that processor p_i is faulty.

The relationships among x_i , x_{i+m}, and c_i are shown in Table 1:

Table 1

x_i	x_{i+m}	c_i
0	0	0
0	1	1
1	0	1
1	1	1

We note that, the above comparison assignments are similar to the *asymmetric comparison model* of Malek [7] :

$c_i = 0$, means that both processors p_i and p_{i+m} are fault-free and both outputs r_i and r^m_{i+m} are correct;
$c_i = 1$, means that at least one of processors (or both) are faulty, but we don't know which one is faulty.

The relationships between x_i, x_{i+m}, and c_i can also be described by boolean equations:

$$c_i = x_i \vee x_{i+m} \tag{1}$$

$$\overline{c}_i = \overline{x}_i \cdot \overline{x}_{i+m} \tag{2}$$

where '$\vee$' and '$\cdot$' are logical 'OR' and logical 'AND' respectively.

Next we will show that detection of faulty processors in a single m-shift system may be achieved by a careful analysis of the boolean product

$$B = \prod_{c_i=1} c_i \cdot \prod_{c_j=0} \overline{c}_j. \tag{3}$$

Expand and simplify the product B as a boolean sum of products of x_i's and $\overline{x}_i$'s (note that any term in this expansion containing *both* x_i and $\overline{x}_i$, for some i, is zero). Then, faulty processors can be *diagnosed* if the following boolean identity, rewritten in terms of the x_i's

$$B = \prod_{c_i=1} (x_i \vee x_{i+m}) \cdot \prod_{c_j=0} (\overline{x}_j \cdot \overline{x}_{j+m}) \equiv 1 \tag{4}$$

has a UNIQUE solution.

EXAMPLE 1. Let us look at a 6 processor system:

$$P = \{p_0, p_1, p_2, p_3, p_4, p_5\},$$

operating under 1-shift diagnostic mode (i.e. $m = 1$).

Suppose there are two failed processors p_1 and p_2. The results under a normal operation are:

$$R = \{r_0, r_1, r_2, r_3, r_4, r_5\}$$

When an error is detected among the r_i's, the system has to go back to the last checkpoint and do a recalculation to provide a 1-shift output array by shifting each processor input to the next :

$$R^1 = \{r^1_0, r^1_1, r^1_2, r^1_3, r^1_4, r^1_5\}.$$

Comparing the previous result R with the current result R^1, we have a comparison array

$$C = \{1,\ 1,\ 1,\ 0,\ 0,\ 0\}.$$

Using Equations (1)-(3), the failed processors can be located:

$$\begin{aligned} 1 &= c_0 \cdot c_1 \cdot c_2 \cdot \overline{c}_3 \cdot \overline{c}_4 \cdot \overline{c}_5 = \\ &= (x_0 \vee x_1) \cdot (x_1 \vee x_2) \cdot (x_2 \vee x_3) \cdot (\overline{x}_3 \cdot \overline{x}_4) \cdot (\overline{x}_4 \cdot \overline{x}_5) \cdot (\overline{x}_5 \cdot \overline{x}_0) \\ &= \overline{x}_0 \cdot x_1 \cdot x_2 \cdot \overline{x}_3 \cdot \overline{x}_4 \cdot \overline{x}_5 \cdot (x_1 \vee x_2). \end{aligned}$$

Each factor in the last product must have value 1, which means that p_0, p_3, p_4, and p_5 are fault-free, and processors p_1 and p_2 are faulty. $\diamond$

2.2. **The Algorithm.**

In this section we present a linear time algorithm to analyze the boolean identity (4). It is a variant of an algorithm for the 2-satisfiability problem, presented in [4].

Let $X = \{x_1, x_2, \dots, x_n\}$ be a set of n 0-1 variables. The 2-satisfiability (2-SAT) problem is to find a solution to a quadratic boolean equation of the form

$$T_1 \cdot T_2 \cdots T_n = 1, \qquad (5)$$

where each term T_k is a *disjunction* or a *conjunction* of two literals. 2-SAT problems have been studied extensively in the literature, including Aspvall, Plass and Tarjan [1], Petreschi and Simeone [9], Hansen and Jaumard [4], among others. In [4], an $O(n)$ algorithm was presented to check uniqueness of the solution to (5). In what follows, we will present a simplified version of this linear algorithm to check unicity in a 2-SAT problem of the special form (4).

Clearly, diagnosability of the entire set of faulty processors is equivalent to unicity of (5), where each term T_k is equal to either $(x_i \vee x_j)$ or $(\overline{x}_i \cdot \overline{x}_j)$.

With (5), we associate the *implication digraph* $G(V \cup \overline{V},\ A)$ defined as follows [1, 4] : the set of vertices is the union of

$$V = \{i : x_i \in X\}, \quad \text{and} \quad \overline{V} = \{\overline{\imath} : x_i \in X\};$$

and A contains two arcs (in fact at most 2, since we do not duplicate arcs) corresponding to each term T_k in (5) :

- if $T_k = (x_i \vee x_j)$, include arcs $(\overline{\imath}, j)$ and $(\overline{\jmath}, i)$ in A
- if $T_k = (\overline{x}_i \cdot \overline{x}_j)$ include arcs $(i, \overline{\imath})$ and $(j, \overline{\jmath})$ in A.

Note that, (a) every arc directed from V to $\overline{V}$ is of type $(r, \overline{r})$; (b) presence of an arc $(r, \overline{r})$ implies that the processor p_r is fault-free; (c) if $(\overline{r}, s) \in A$, then $r \neq s$; and (d) presence of an arc $(\overline{r}, s)$ in A corresponds to the fact that AT LEAST one of the processors p_r or p_s is faulty. These observations lead us to the following result.

Proposition 1. *G is bipartite, and each vertex of G is a strongly connected component of G.*

Proof : Clearly, $V \cup \overline{V}$ provides a bipartition of the vertices in G, since by definition A does not contain arcs of the form (i, j) or $(\overline{\imath}, \overline{\jmath})$.

Suppose that a strongly connected component S of G contains more than one vertex. Following our observations above, we can always find a pair of distinct vertices i and j in S lying on a circle of the form

$$i \longrightarrow \overline{\imath} \longrightarrow \cdots \longrightarrow j \longrightarrow \overline{\jmath} \longrightarrow i.$$

Now, the existence of the arcs $(i, \bar{\imath})$ and $(j, \bar{\jmath})$ imply that *both* p_i and p_j are fault-free processors; however, presence of the arc $(\bar{\jmath}, i)$ indicates that *at least one* of these processors *must* be faulty, a contradiction. ■

The above result provides the most significant modification to the algorithm of Hansen and Jaumard [4]. The original procedure requires identification of all strongly connected components of the implication digraph G, which in our case is a triviality. The simplified version of this labeling algorithm is presented below :

INPUT DATA. Number of processors n, and the set of terms in (5), $\langle T_k \rangle_{k=1}^{n}$.

STEP 1. Form the implication digraph G. Initially all nodes are unlabeled.

STEP 2. For each arc of type $(i, \bar{\imath}) \in A$, label

(i) nodes i, $\bar{\imath}$;

(ii) *all* descendants of $\bar{\imath}$; and

(iii) *all* ancestors of i.

STEP 3. **If** any node remains unlabeled at the end of Step 2, STOP, (4) has no unique solution. **Else** compute the unique solution to (4) as

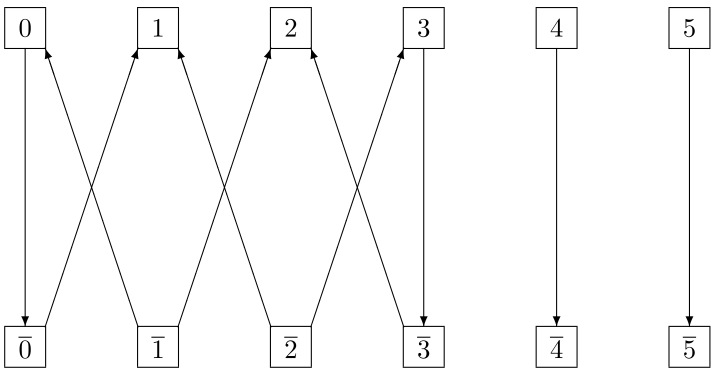
$$x_i = \begin{cases} 0 & \text{if } (i, \bar{\imath}) \in A \\ 1 & \text{otherwise} \end{cases} \tag{6}$$

for all $0 \le i \le n - 1$, STOP.

Note that in Step 2, (ii) and (iii) can be carried out simultaneously, as j is a descendant of $\bar{\imath}$ implies $\bar{\jmath}$ is an ancestor of i. Since, $|V \cup \overline{V}| = 2n \geq |A|$, the above algorithm is clearly $\mathrm{O}(n)$. Proof of correctness is similar to one in [4], and so is omitted.

Example 1 (contd). The implication digraph corresponding to example 1 is shown in Figure 1 :

Figure 1. Implication digraph for Example 1



An application of the algorithm, discussed before, would result in a labeling of all vertices. From (6), the corresponding unique solution will then be

$$x_0 = 0 = x_3 = x_4 = x_5, \text{ and } x_1 = 1 = x_2. \diamond$$

Interchanging i and $\bar{\imath}$ and reversing orientations in the implication digraph is the identity operation. Therefore the (unoriented) graph obtained from the implication digraph by identifying vertices i and $\bar{\imath}$, replacing a pair of parallel arcs of opposing directions by an edge and directed loops by loops, yields a graph, the *implication graph*, carrying the same information as the implication digraph. The implication graph can be directly defined as follows.

The implication graph associated with the boolean product

$$B = \prod_{c_i=1} c_i \cdot \prod_{c_j=0} \bar{c}_j = \prod_{c_i=1} (x_i \vee x_{i+m}) \cdot \prod_{c_j=0} (\overline{x}_j \cdot \overline{x}_{j+m})$$

has vertex set $\{0, 1, \dots, n-1\}$, contains loops at nodes i and $i+m$ iff $c_i = 0$ and nodes i and $i+m$ are joined by an edge iff $c_i = 1$.

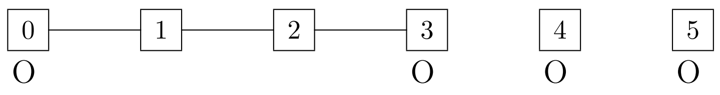
The problem of diagnosing the set of faulty processors has the following attractive reformulation in terms of the properties of the implication graph.

Proposition 2. (4) *has a unique solution if and only if the subset* $\mathcal{O} = \{j,\, j+m : c_j = 0\}$ *of vertices of the implication graph is dominating and stable.*

Proof : If $\mathcal{O}$ is not dominating, (4) admits multiple solutions, since $x_i = 0$ or $x_i = 1$ both satisfy (4) if node i is not adjacent to a node in $\mathcal{O}$. If $\mathcal{O}$ is not stable, (4) has no solution. If $\mathcal{O}$ is *both* dominating and stable, then (4) has the unique solution : $x_j = x_{j+m} = 0$, if $c_j = 0$, and $x_i = 1$ otherwise. ■

EXAMPLE 1 (CONTD). The implication graph corresponding to example 1 is shown in Figure 2 :

Figure 2. Implication *graph* for Example 1



Also note that $\mathcal{O} = \{0, 3, 4, 5\}$ is stable, dominating and contains all fault-free processors, i.e., all nodes incident on loops. ◇

Proposition 2 leads to a simple algorithm for checking unicity of (4) : record $\mathcal{O}$; stop if $\mathcal{O}$ is empty; otherwise check if $\mathcal{O}$ is dominating (stability of $\mathcal{O}$ follows from our assumptions about the processors). Since every vertex in $\mathcal{O}$ is incident on at most two edges, the running time for this algorithm is also linear in n. However, our first algorithm is applicable to more general situations.

2.3. **Diagnosing *all* faulty processors.**

In this section we will characterize situations where a single m-shift system *will* locate all faulty processors. Furthermore, we will present a generalized version of the labeling algorithm which is *guaranteed* to diagnose all faulty processors. First, consider the following definitions.

DEFINITION 1. A *faulty processor set* (FPS) is a set F of processors which contains all the failed processors in the system. ■

DEFINITION 2. An FPS is said to be *diagnosable* in a single m-shift system if all the failed processors in it can be located by the single m-shift operation. ■

DEFINITION 3. Given a processor p_i in a single m-shift system, the processor p_{i+m} is called the *forward neighbor* of p_i and the processor p_{i-m} is called the *backward neighbor* of p_i. ■

The following result provides a way to determine whether a faulty set is diagnosable.

Proposition 3. *A faulty processor set F is diagnosable in a single m-shift system if and only if for any processor p_i (faulty or fault-free) in the system there is at least one fault-free neighbor processor (either the forward neighbor p_{i+m} or the backward neighbor p_{i-m}, or both).* ■

Proof : Suppose there exists a processor p_i, such that BOTH p_{i-m} and p_{i+m} are faulty. Note that information about p_i is available only through the comparison elements c_{i-m} and c_i.

It is easy to see that $c_{i-m} = 1$, and $c_i = 1$, irrespective of the fact that p_i is faulty or not. Hence it is impossible to decide whether or not p_i is a member of F, contradicting its diagnosability.

Conversely, let p_i be such that p_{i+m} is fault-free (the case when p_{i-m} is not faulty can be treated similarly).

Case 1 : p_i is not faulty. Then outputs r_i and r^m_{i+m} are identical, i.e. $c_i = 0$. Hence, under the m-shift system, the i-th element of the comparison array appearing in the expansion of B (equation (3)) is

$$\overline{c}_i = \overline{x}_i \cdot \overline{x}_{i+m},$$

implying that p_i can be diagnosed as a fault-free processor.

Case 2 : Suppose $p_i \in F$. By assumption p_{i+2m} (as a forward neighbor of p_{i+m}) must be fault-free. Comparing the outputs r_i, r^m_{i+m} and r^m_{i+2m}, we have $c_i = 1$ and $c_{i+m} = 0$. Then from equations (1) and (2)

$$1 = c_i \cdot \overline{c}_{i+m} = (x_i \vee x_{i+m}) \cdot \overline{x}_{i+m} \cdot \overline{x}_{i+2m} = x_i \cdot \overline{x}_{i+m} \cdot \overline{x}_{i+2m},$$

which identifies p_i as a member of F. ■

It follows from the above result that a single m-shift operation (for a specific value of m) need not lead us to a 2-SAT problem (see equation (4)) with a unique solution.

Example 2. Consider a 6 processor system: $P = \{p_0, p_1, p_2, p_3, p_4, p_5\}$, where processors p_1 and p_3 are faulty. A single 1-shift operation leads to the implication digraph shown in Figure 3. It is easy to see that the labeling algorithm will fail to label nodes 2 and $\overline{2}$. So, the status of processor p_2 will not be determined. One may also check that performing a 2-shift operation leads to a similar inconclusive result. Also, in the implication graph, shown in Figure 4, the set $\mathcal{O} = \{0, 4, 5\}$ is not dominating as it fails to dominate node 2. ◇

Figure 3. Implication digraph for Example 2

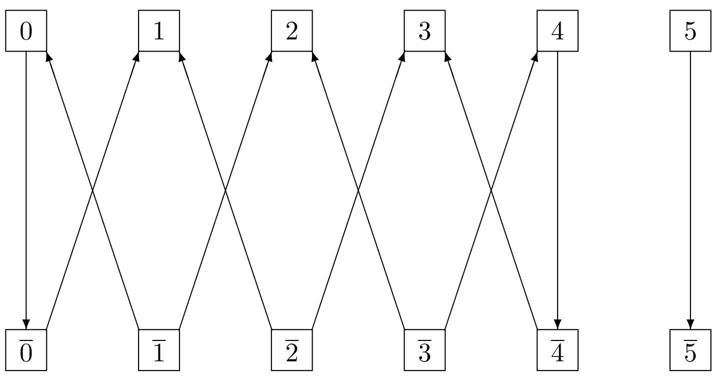
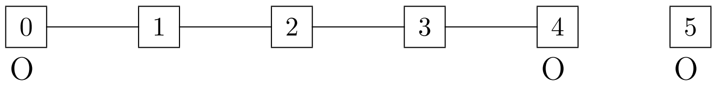


Figure 4. Implication graph for Example 2



To overcome this problem, consider *repetitions* of single m-shift operations for distinct values of m. Combining all comparison arrays resulting from respective m-shift output arrays we obtain a 2-SAT problem of type (4), where the number of terms is a *multiple* of n (instead of being *equal* to n). The corresponding implication digraph still has the same vertex set and, although the edge set is larger, conclusions of Proposition 3 remain valid, in fact the same proof holds. The possible advantage in this approach is that a larger edge set results in a denser digraph, thereby increasing the chances of all nodes being labeled. Naturally, it is of interest to investigate the number of single shift operations one needs to perform which guarantees a unique solution in (4). In the following, this scheme of repeating single shift operations will be termed as *multiple shift operation.*

Let k denote the number of faulty processors in the system. Clearly, if $k = n$ or $n - 1$, none of the processors can be diagnosed in a single shift operation or in multiple shift operations. In the following result we will show that only O(k) repetitions of single shift operations are needed to identify *all* faulty processors.

Proposition 4. *If* $k \leq n-2$, *all faulty processors can be diagnosed by performing at most* $r = \left[\frac{k}{2}\right] + 1$ *single shift operations.*

Proof : Perform m-shift operations for r values of $m = 1, 2, \ldots, r$. Then, for each processor p_i, $(1 \leq i \leq n,)$ we would have $2r$ comparison elements between p_i and processors p_j for $j \in I$, where

$$I = \{i \pm t : t = 1, 2, \ldots, r\}.$$

Let us denote $l = |I| + |\{p_i\}|$.

If possible, suppose that the status of p_i cannot be diagnosed. This is true *if and only if* all processors p_j, for $j \in I$, are faulty (Proposition 3). We have two cases :

CASE 1. $l < n$.

Then, all processors p_j, for $j \in I$, are *distinct* and are faulty. This however implies that the number of faulty processors is at least

$$|I| = 2r = \begin{cases} k+2 & \text{if } k \text{ is even} \\ k+1 & \text{otherwise,} \end{cases}$$

contradicting the fact that there are only k faulty processors.

CASE 2. $l \geq n$.

In this case, I contains all processor indices except that of p_i. Thus, we must have $I = \{1, 2, \ldots, i-1, i+1, \ldots, n\}$, implying that $k \geq |I| = n - 1$, which is again a contradiction. ■

EXAMPLE 2 (CONTD). Let us now consider the earlier example, where a single 1-shift or 2-shift operation failed to diagnose all faulty processors. Combining results from these single m-shift operations (for $m = 1$ and 2) we get the 2-satisfiability problem :

$$\begin{aligned}1 \;=\; &(x_0 \;\vee\; x_1)\cdot(x_1 \;\vee\; x_2)\cdot(x_2 \;\vee\; x_3)\cdot(x_3 \;\vee\; x_4)\cdot(\overline{x}_4 \;\cdot\; \overline{x}_5)\cdot(\overline{x}_5 \;\cdot\; \overline{x}_0)\\ &\cdot(\overline{x}_0 \;\cdot\; \overline{x}_2)\cdot(x_1 \;\vee\; x_3)\cdot(\overline{x}_2 \;\cdot\; \overline{x}_4)\cdot(x_3 \;\vee\; x_5)\cdot(\overline{x}_4 \;\cdot\; \overline{x}_0)\cdot(x_5 \;\vee\; x_1).\end{aligned}$$

The labeling algorithm identifies the unique solution

$$x_0 \;=\; 0 \;=\; x_2 \;=\; x_4 \;=\; x_5, \text{ and } x_1 \;=\; 1 \;=\; x_3. \diamond$$

A straightforward implementation of multiple shift operations would thus be as follows : for each value of $m = 1, 2, 3, \ldots$ combine all results from the m comparison arrays obtained so far; form the implication digraph corresponding to the mth 2-SAT problem; and continue till all nodes get a label. As proved in Proposition 4, this process terminates after at most $O(k)$ iterations and is guaranteed to identify all faulty processors. A careful analysis, however, reveals certain redundancies. At iteration m the implication digraph has $O(mn)$ edges, i.e. the edge set continues to grow in size. Moreover, any information regarding the status of a processor obtained within an intermediate iteration is left unused in all future calculations. Next we consider a generalized labeling algorithm, which implements the multiple shift operation scheme more *efficiently.*

Step 0. Given the number of processors n, generate all terms $\langle T_k\rangle_{k=1}^n$ corresponding to $m = 1$. Form the implication digraph $G(V \cup \overline{V}, E)$. Initially, *all* vertices are unlabeled.

Step 1. For each unlabeled node $i \in V$ such that arc $(i, \overline{\imath}) \in A$, label (i) nodes i, $\overline{\imath}$; (ii) *all* descendants of $\overline{\imath}$; and (iii) *all* ancestors of i.
If all vertices are labeled, compute the unique solution as in (6) and STOP.
Else, go to Step 2.

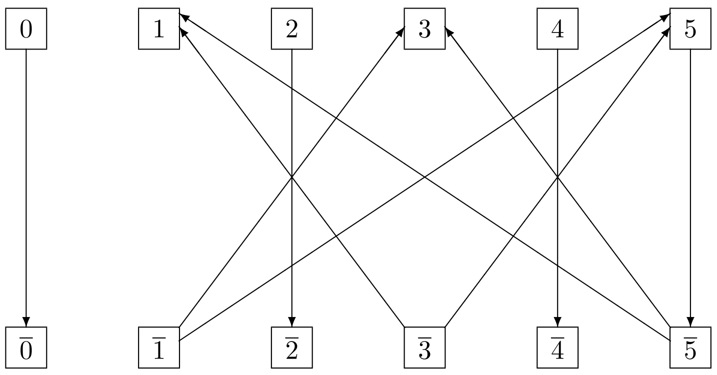
Step 2. Set $m = m + 1$, and generate $\langle T_k\rangle_{k=1}^n$ corresponding to this new value of m. *Remove* all edges in E of type $(\overline{r}, s)$ and then *add* new edges corresponding to the updated T_k elements. Go to Step 1.

Note that the generalized algorithm stops only when it identifies the status of all processors (i.e. when all nodes get labeled). Every vertex is labeled exactly once before termination, and the labeling is never erased. In Step 2, we add at most $O(n)$ edges to the *reduced* edge set E containing at most n edges (number of edges deleted is again $O(n)$). Note also that we do not add edges of type (i, j) or $(\overline{\imath}, \overline{\jmath})$. So, at every iteration, the underlying modified digraph is bipartite, contains at most $O(n)$ edges and every vertex remains to be a strongly connected component. Furthermore, Proposition 4 implies that the iteration count never exceeds $\left[\frac{k}{2}\right] + 1$. Thus we have the following result :

Proposition 5. *If $k \leq n - 2$, the multiple shift operation, as implemented by the generalized labeling algorithm, identifies all faulty processors in $O(nk) \approx O(n^2)$ time.* ■

Example 2 (contd). Consider now applying the generalized labeling algorithm to the previous problem. The first step is identical to the single 1-shift operation, and the implication digraph is same as in Figure 3. Since node 2 (or $\overline{2}$) does not get labeled, the algorithm moves on to Step 2. All edges except $(0, \overline{0})$, $(4, \overline{4})$ and $(5, \overline{5})$ are removed. Addition of new edges (corresponding to the single 2-shift operation) gives the following modified implication digraph :

Figure 5. Modified implication digraph for Example 2



The only unlabeled nodes 2 and $\overline{2}$ are labeled immediately in the next iteration, identifying processors p_1 and p_3 as faulty. $\diamond$

We may point out that in almost all practical situations the number k of faulty processors is relatively small compared to n, and so in most cases the multiple shift operation scheme behaves as a linear time algorithm. Further, in such cases, the set $\mathcal{O}$ defined for the implication graph (Section 2.2) would be nonempty for all choices of m. If $\mathcal{O}$ is not dominating, one can use the implication graph to identify additional faulty processors by properly choosing future values of m.

3. Concluding Remarks

In this paper we have presented simple labeling algorithms to efficiently diagnose some or all faulty processors in an n processor system. The number of comparisons performed before locating the faulty processor set is *at most* $n\left([k/2]+1\right)$, which is *optimal* (as defined in [8]). Classical diagnosis models [2, 7, 10] are based on a pre-specified choice of an upper bound t on the number of faulty processors. In the present work we relax this by exhibiting that as many as $n-2$ faulty processors can be diagnosed.

Although the necessary and sufficient conditions for diagnosability (Proposition 3) for a single shift system is dependent on the system design, the diagnostic analysis procedure (using the 2-satisfiability formulation) is applicable to any collection of comparison test results. In particular, we will consider diagnosability of faulty processors within a distributed system in a future project.

We note that diagnosis of faulty processors in a system may be considered as a precursor to the process of *replacing* incorrect output data with correct results. In a related work [12], we address this topic and characterize situations when an extended diagnosis algorithm would allow us to replace faulty data without actually replacing the faulty processors.

Acknowledgments : The authors would like to thank Prof. Brigitte Jaumard for bringing the reference [4] to their attention, and an anonymous referee for suggesting Proposition 2.

References

[1] Aspvall, B., M.F. Plass and R.E. Tarjan, "A linear time algorithm for testing the truth of certain quantified boolean formulas", *Inform. Process. Lett.* **8** (1979), 121-123.

[2] Chwa, K-Y. and S.L. Hakimi, "Schemes for Fault-Tolerant Computing: A Comparison of Modularly Redundant and t-Diagnosable Systems", *Information and Control* **49** (1981) 212-238.

[3] "Connection Machine : Model CM-2 Technical Summary", The Connection Machine System Technical Report, Thinking Machines Corporation, Cambridge, Massachusetts (1989).

[4] Hansen, P. and B. Jaumard, "Uniquely solvable quadratic boolean equations", *Discrete Applied Mathematics* 12 (1985), 147-154.

[5] Kuhl, J.G. and S.M. Reddy, "Distributed Fault-Tolerance for Large Multiprocessor Systems", *Proc. IEEE Symp. Comp. Architecture* (1980) 23-30.

[6] Maeng, J. and M. Malek, "A Comparison Connection Assignment for Self-Diagnosis of Multiprocessor Systems", *Proc. IEEE FTCS* **11** (1981) 173-175.

[7] Malek, M., "A Comparison Connection Assignment for Diagnosis of Multiprocessor Systems", *Proc. IEEE Symp. Comp. Architecture* (1980) 31-36.

[8] Pelc, A., "Undirected Graph Models for System-Level Fault Diagnosis", *IEEE Trans. Comp.* **40** (1991) 1271-1276.

[9] Petreschi, R. and B. Simeone, "A switching algorithm for the solution of quadratic boolean equations", *Inform. Process. Lett.* **?** (1980), 193-198.

[10] Preparata, F.P., G. Metze and R.T. Chien, "On the Connection Assignment Problem of Diagnosable Systems", *IEEE Trans. Electr. Comp.* **EC-16** (1967) 848-854.

[11] Shi, W., B. Servatius and A. Bagchi, "Fault Tolerance in Massively Parallel Computers", Technical Report, Thinking Machines Corporation, Cambridge, MA (1992).

[12] Shi, W., B. Servatius and A. Bagchi, "A Fault Tolerant Computing Approach for Massively Parallel Computing Systems", Technical Report, Department of Mathematical Sciences, WPI, Worcester, MA (1992).

Mathematical Sciences Department, Worcester Polytechnic Institute, 100 Institute Road, Worcester, MA 01609

Thinking Machines Corporation, 245 First Street, Cambridge, MA 02142