# Optimal Reconfiguration Algorithms for Real-Time Fault-Tolerant Processor Arrays

Ran Libeskind-Hadas, *Member, IEEE,* Nimish Shrivastava, Rami G. Melhem, and C. L. Liu, *Fellow, IEEE*

***Abstract*—In this paper we consider the problem of reconfiguring processor arrays subject to computational loads that alternate between two modes. A *strict* mode is characterized by a heavy computational load and severe constraints on response time while a *relaxed mode* is characterized by a relatively light computational load and relaxed constraints on response time. In the strict mode, reconfiguration is performed by a distributed local algorithm in order to achieve fast recovery from faults. In the relaxed mode, a global reconfiguration algorithm is used to restore the system to a state that maximizes the probability that future faults occurring in subsequent strict modes will be repairable.**

**Several new results are given for this problem. Efficient reconfiguration algorithms are described for a number of general classes of architectures. These general algorithms obviate the need for architecture-specific algorithms for architectures in these classes. We show that it is unlikely that similar algorithms can be obtained for related classes of architectures since the reconfiguration problem for these classes is NP-complete. Finally, a general approximation algorithm is described that can be used for any architecture. Experimental results are given, suggesting that our algorithms are very effective.**

## I. Introduction

ADVANCES in VLSI and WSI technologies allow increasingly larger processor arrays to be fabricated on a single chip or wafer. As the number of processors in an array increases, the problem of reconfiguring the array to replace faults occurring at run-time becomes increasingly important. One common way of providing fault tolerance in processor arrays is to augment the array with a set of spare processors that can replace primary processors that become faulty. This approach has been proposed for a number of architectures [2], [10], [13], [18], [19], [21] and a variety of reconfiguration algorithms for these reconfigurable systems have been studied.

It has been observed that in many real-time applications, systems are subject to computational loads that alternate between a *strict* mode in which the computational load is heavy and severe constraints are imposed on response time, and a *relaxed* mode in which the computational load is light and the constraints on response time are relaxed substantially [8], [11]. In order to achieve fast response time, it is desirable that reconfiguration in the strict mode be performed in a distributed fashion so as not to incur the overhead of communication with a central host processor. Moreover, in order to minimize interruption in service in the strict mode, it is important that the replacement of a faulty processor causes only a minimal number of changes to the existing system interconnections. Therefore, during strict mode reconfiguration, a processor uses only very local knowledge about its immediate neighbors. These objectives are met by using a distributed local algorithm in the strict mode in which a faulty processor finds a replacement by selecting an available spare processor to which it is directly connected [11].

Although local reconfiguration allows fast replacement of faulty processors, repeated application of the local reconfiguration algorithm may quickly degrade the reliability of the system since spare elements are often not used in the most effective way. Consequently, in the relaxed mode we wish to use a global reconfiguration algorithm to restore the system to a more reliable state. Specifically, the goal of reconfiguration in the relaxed mode is to assign faulty elements to spare elements maximizing the probability that any processor becoming faulty in the next strict mode will be repairable by the local reconfiguration algorithm. Such an assignment is called an *optimal assignment.* This approach of using both local and global reconfiguration algorithms, depending on the state of the system, is called *bilevel reconfiguration.* It was shown in [11] that bilevel reconfiguration can substantially improve the expected lifetime of a system and this approach was illustrated for the case of an augmented mesh architecture. Subsequently, Chen *et al.* [3] and Shrivastava and Melhem [17] independently discovered efficient optimal assignment algorithms for this particular architecture.

In this paper we investigate the problem of finding optimal assignments for several broad classes of architectures. In Section II we give a formal description of the optimal assignment problem and show that the problem is, in the general case, NP-complete. In Section III we give several basic results that are used throughout the paper. In Sections IV, V, and VI we show that optimal assignments can be found in polynomial time for several classes of architectures, including a number of well-known architectures that have been proposed in the literature. Moreover, in Section VII we give an efficient approximation algorithm that can be used for any architecture. Experimental results are given at the end of each of Sections IV, V, VI, and VII to illustrate the effectiveness of these algorithms. Conclusions are given in Section VIII.

Manuscript received January 18, 1993; revised May 16, 1994. This work was supported in part by a GTE graduate fellowship, University of Illinois, Urbana-Champaign and by the National Science Foundation under Grants MIP 89-06932 and MIP 89-11303.

R. Libeskind-Hadas is with the Department of Computer Science, Harvey Mudd College, Claremont, CA 91711 USA.

N. Shrivastava and R. G. Melhem are with the Department of Computer Science, University of Pittsburgh, PA USA.

C. L. Liu is with the Department of Computer Science, University of Illinois, Urbana-Champaign, IL 61801 USA.

IEEE Log Number 9409870.

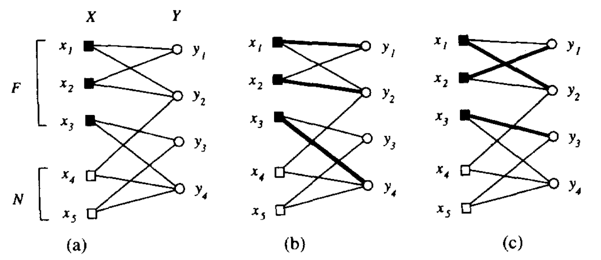


Fig. 1. (a) A bipartite graph. (b) A matching with one unrepairable vertex, $x_4$. (c) A safe matching.

## II. Optimal Reconfiguration

In this section we formally define the reconfiguration problems studied in this paper. We begin by introducing some notation and definitions that will simplify our discussion.

A processor array can be represented by a bipartite graph $G = (X \cup Y, E)$ where $X$ is the set of vertices representing primary processors and $Y$ is the set of vertices representing nonfaulty spare processors.[1] There is an edge $\{x, y\} \in E$ iff the primary processor corresponding to $x \in X$ can be replaced by the spare processor corresponding to $y \in Y$. The connection allowing a spare processor to replace a primary processor may be due to a physical link between the two processors or some other mechanism such as a crossbar switch [15]. The vertices in $X$ are denoted *primary vertices* and the vertices in $Y$ are denoted *spare vertices*. We let $F \subseteq X$ represent the set of faulty primary processors and let $N = X - F$ represent the set of nonfaulty primary processors. The vertices in $F$ are denoted *faulty primary vertices* or simply *faulty vertices* and the vertices in $N$ are denoted *nonfaulty primary vertices* or simply *nonfaulty vertices*.

A matching, $M$, of $F$ into $Y$ represents an assignment of faulty primary processors to nonfaulty spare processors. A vertex $x \in N$ is said to be *repairable* with respect to $M$ if it is adjacent to an unmatched vertex in $Y$. Thus, a repairable vertex corresponds to a nonfaulty primary processor that can be replaced using the local reconfiguration algorithm. A vertex $x \in N$ that is adjacent only to matched vertices in $Y$ is said to be *unrepairable* since the corresponding primary processor cannot be replaced by the local reconfiguration algorithm. Our objective is to find a matching for $F$ such that every vertex in $N$ is repairable and thus any fault occurring in the next strict phase will be successfully replaced by the local reconfiguration algorithm. Such a matching is called a *safe matching*. Fig. 1 illustrates a matching and a safe matching in a bipartite graph with three faulty vertices. (Throughout this paper we will use the convention that primary vertices are denoted by empty squares, faulty primary vertices are denoted by darkened squares, and spare processors are denoted by empty circles.)

For some patterns of faults, a safe matching may not exist. In this case, we wish to find a matching for $F$ such that the number of repairable vertices in $N$ is maximized, thus maximizing the probability that a fault occurring in the next strict phase will be successfully replaced by the local reconfiguration algorithm. Such a matching corresponds to an optimal assignment and is therefore called an *optimal matching*. In order to formally study the complexity of these problems, we define the decision problems corresponding to optimal and safe matchings as follows.

*1) Safe Matching Decision Problem (SMDP):*

INSTANCE: Bipartite graph $G = (X \cup Y, E)$ and $F \subseteq X$.
QUESTION: Does there exist a matching $M$ of $F$ into $Y$ such that all vertices in $N = X - F$ are repairable with respect to $M$?

*2) Optimal Matching Decision Problem (OMDP)*

INSTANCE: Bipartite graph $G = (X \cup Y, E)$, $F \subseteq X$, and positive integer $K$.
QUESTION: Does there exist a matching $M$ of $F$ into $Y$ such that at least $K$ vertices in $N = X - F$ are unrepairable with respect to $M$?

Note that since the safe matching problem is a special case of the optimal matching problem, a polynomial time algorithm for the optimal matching problem implies a polynomial time algorithm for the safe matching problem.

*Theorem 1:* SMDP is NP-complete.

*Proof:* SMDP is clearly in NP since we can guess a matching for $F$ and verify in polynomial time that all vertices in $N$ are repairable with respect to this matching. To show that SMDP is NP-complete, we reduce the NP-complete 3-Satisfiability Problem (3SAT) to this problem. 3SAT is defined as follows [6]. Given a set $U$ of variables and a collection $C$ of clauses over $U$ such that each clause $c \in C$ has $|c| = 3$, is there a truth assignment for $U$ that simultaneously satisfies all of the clauses in $C$?

We transform a given instance of 3SAT to an instance of SMDP as follows. For each variable $u \in U$ there is a vertex $x_u \in F$, vertices $y_{u,T}, y_{u,F} \in Y$ and edges $\{x_u, y_{u,T}\}$, $\{x_u, y_{u,F}\} \in E$. For each clause $c \in C$, where $c$ contains literals $s$, $t$, and $r$, there is a vertex $x_c \in N$. Let $v$ denote the variable for literal $s$. If $s$ is the unnegated variable $v$ then there is an edge $\{x_c, y_{v,F}\} \in E$ and if $s$ is $\neg v$ then there is an edge $\{x_c, y_{v,T}\} \in E$. Similarly, a single edge is added for each literal $t$ and $r$.

We claim that there is a truth assignment that satisfies all of the clauses in $C$ iff there is a safe matching in the constructed graph. Assume that there is a truth assignment that satisfies all of the clauses in $C$. Let $s_i$, $t_i$, and $r_i$ denote the literals in clause $c_i$. For each variable $u \in U$ we include edge $\{x_u, y_{u,T}\}$ in matching $M$ if $u$ is **true** in the assignment and include the edge $\{x_u, y_{u,F}\}$ if $u$ is **false**. This construction results in a matching for $F$. Moreover, for each $c_i \in C$, the vertex $x_{c_i} \in N$ is repairable with respect to $M$. To see this, observe that since $c_i$ is satisfied by the truth assignment, at least one of the literals $s_i$, $t_i$, or $r_i$ is **true**. Without loss of generality, assume $s_i$ is **true** and let $v$ denote the variable for $s_i$. If $s_i$ is $v$ then $v$ is **true**, $\{x_v, y_{v,T}\} \in M$ by construction of $M$, and thus $y_{v,F}$ is unmatched. Since $x_{c_i}$ is adjacent to $y_{v,F}$ by construction, $x_{c_i}$ is repairable. Similarly, if $s_i$ is $\neg v$ then $v$ is **false**, $\{x_v, y_{v,F}\} \in M$, by construction of $M$, and thus $y_{v,T}$ is unmatched. In this case, $x_{c_i}$ is adjacent to $y_{v,T}$ and $x_{c_i}$ is repairable.

[1]Faulty spare processors may be disregarded since they cannot be used to replace primary processors.

Conversely, assume that there exists a matching $M$ for $F$ such that all vertices in $N$ are repairable. For each variable $u \in U$, if $x_u$ is matched to $y_{u,T}$ then set $u = \mathbf{true}$ and, otherwise, set $u = \mathbf{false}$. Since $x_{c_i}$ is repairable, at least one of the three adjacent vertices is unmatched. If $x_{c_i}$ is adjacent to unmatched vertex $y_{v,T}$ then variable $v$ was set to **false** and, by construction, $c_i$ contains the literal $\neg v$ and is therefore satisfied. Similarly, if $x_{c_i}$ is adjacent to unmatched vertex $y_{v,F}$ then variable $v$ was set to **true** and, by construction, $c_i$ contains the literal $v$ and is therefore also satisfied. Therefore, all clauses in $C$ are satisfied by this assignment. □

*Corollary 1:* OMDP is NP-complete.

Although Theorem 1 and its corollary tell us that the safe matching and optimal matching problems are NP-complete in general, some architectures have properties that allow us to find efficient algorithms for these problems. In Section IV we show that the optimal matching problem (and therefore the safe matching problem) can be solved in linear time for acyclic graphs.[2] This implies that optimal assignments can be found in a number of tree architectures [7], [18], [19] and interstitial redundancy schemes [19]. This algorithm can also be applied to any architecture to find an optimal solution when the fault pattern in the array induces an acyclic graph.

A primary processor can generally be connected to only a small fixed number of spare processors, and similarly, a spare processor can often only be connected to a small number of primary processors. This is due to the fixed number of ports on a processor and, in VLSI implementations, layout considerations that tightly restrict the number of neighbors to which a processor may be connected. We show that for certain bounds on processor degrees, safe matchings and optimal matchings can be found by efficient algorithms.

In Section V we consider architectures in which the primary processors have constant degree. We show that when each primary processor is adjacent to at most two spares, safe matchings can be found in polynomial time. This implies that safe matchings can be found for a number of well-known architectures. For example, this result applies to the fault tolerant binary tree architecture proposed by Hassan and Agarwal [7] and Raghavendra *et al.* [14], the interstitial redundancy scheme used in the Hughes 3-D Computer [21], and the 2-D augmented mesh proposed in [11], among others. We also show that this result is tight in the sense that the safe matching problem is NP-complete when each processor is adjacent to at most three spares. Moreover, the optimal matching problem remains NP-complete even if each processor is adjacent to at most two spares. However, we show that for arrays that satisfy some additional properties, the optimal matching problem can be solved efficiently even if each processor is adjacent to two or more spares.

In Section VI we consider architectures in which the spare processors have constant degree. We show that when each spare processor is adjacent to at most two primary processors, optimal matchings (and therefore safe matchings) can be found in linear time. This result implies that optimal assignments can

[2]Recall that this means that the bipartite graph is acyclic. That is, there is no cycle in the processor array that alternates between primary processors and spare processors. The processor array may, however, contain other cycles.

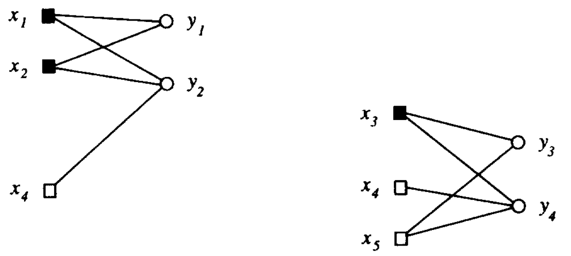


Fig. 2. The regions corresponding to the bipartite graph in Fig. 1(a).

be found efficiently in a number of well-known architectures, such as the fault tolerant augmented hypercube proposed by Banerjee [1] and a number of interstitial array schemes [19]. However, the problem of finding even safe matchings becomes NP-complete when each spare is adjacent to at most 3 primary processors.

In Section VII we show how the NP-complete cases can be handled effectively. We give an efficient approximation algorithm that finds solutions that are within a constant factor of optimal for any architecture.

## III. Definitions and Lemmas

In this section we introduce notation and state and prove several key lemmas that will be used throughout this paper.

Let $G = (X \cup Y, E)$ be a bipartite graph and let $F \subseteq X$ be the set of faulty vertices. A subgraph of $G$ induced by a single spare vertex and all its adjacent primary vertices is called a *1-fault-tolerant set* (or *1-FT set*). Observe that if a 1-FT set contains no faulty vertices then all of the primary vertices in this 1-FT set will be repairable with respect to any matching of $F$ into $Y$. Consequently, all the vertices and incident edges in such a 1-FT set can be removed from $G$. Similarly, if a faulty primary vertex $x$ is in only one 1-FT set, then $x$ must be replaced by the unique spare $y$ in that 1-FT set. Consequently, vertices $x$ and $y$ and all incident edges may be removed from $G$. A graph $G$ containing a 1-FT set with no faulty vertices, or a faulty primary vertex in only one 1-FT set, is said to be *reducible* and is said to be *irreducible* otherwise. Without loss of generality, in the remainder of this paper we assume that all graphs under consideration are irreducible.

Let $\mathcal{S}$ denote the set of all 1-FT sets in $G$. (By the irreducibility assumption, each of these 1-FT sets contains at least one faulty primary vertex). Let $T$ be a binary relation on $\mathcal{S}$ defined as follows. For $s_i, s_j \in \mathcal{S}$, $(s_i, s_j) \in T$ iff $s_i$ and $s_j$ contain a common faulty primary vertex. Since $T$ is both reflexive and symmetric, the transitive closure of $T$, denoted $T'$, is an equivalence relation of $\mathcal{S}$. Each equivalence class of $T'$ is a collection of 1-FT sets. The subgraph of $G$ induced by the 1-FT sets in an equivalence class of $T'$ is called a *region* of $G$. Fig. 2 shows an example of regions for the bipartite graph in Fig. 1(a). The following lemma characterizes four important properties of regions.

*Lemma 1:* Let $G = (X \cup Y, E)$ be a bipartite graph and let $F \subseteq X$ be the set of faulty vertices.

1) Each region is a connected graph.
2) Each region remains connected upon removal of non-faulty vertices and their incident edges.

3) Every faulty vertex is in exactly one region.
4) A faulty vertex in region $R$ is not adjacent to any spare vertex that is not in region $R$.

*Proof:* Follows immediately from the definition of regions. □

We now give three properties of bipartite graphs that are exploited in several of the algorithms presented in subsequent sections. Let $H = (A \cup B, E)$ be a bipartite graph[3] and let $S \subseteq A \cup B$. We define $D(S)$ to be the maximum degree among all vertices in $S$.

*Lemma 2:* Let $H = (A \cup B, E)$ be a connected bipartite graph.

1) If $D(A) \leq 2$ then $|A| \geq |B| - 1$.
2) If $H$ is acyclic and every vertex in $A$ has degree exactly 2 then $|A| = |B| - 1$.

*Proof:* Let $\ell = |B|$ and assume that $|A| = \ell - k$ where $k \geq 2$. Since $H$ is connected, it has at least

$$\ell + (\ell - k) - 1 = 2\ell - k - 1$$

edges. Therefore, there exists a vertex in $A$ with degree at least

$$\frac{2\ell - k - 1}{\ell - k} = \frac{2(\ell - k) + k - 1}{\ell - k} > 2.$$

However, this contradicts our assumption that $D(A) \leq 2$.

Next, assume that $H$ is acyclic and every vertex in $A$ has degree exactly 2. If $|B| \leq |A|$ then there are $|A| + |B| - 1 \leq 2|A| - 1$ edges in the tree, contradicting the assumption that every vertex in $A$ has degree exactly 2. If $|B| > |A| + 1$ then there are $|A| + |B| - 1 > 2|A|$ edges in the graph, again contradicting the assumption that every vertex in $A$ has degree exactly 2. □

*Lemma 3:* Let $H = (A \cup B, E)$ be a connected bipartite graph. If $D(A) \leq 2$ and $|A| = |B|$ then there exists a perfect matching of $A$ into $B$ and this matching can be found in linear time.

*Proof:* Let $T$ be a spanning tree of $H$. Tree $T$ can be found in linear time using breadth-first search. We prove the stronger results that there exists a perfect matching in $T$ from $A$ to $B$. We show that there exists a perfect matching in $T$ by induction on $n = |A|$. For $n = 1$ there is clearly a perfect matching. Assume that the assertion is true for $n = k - 1$ and consider $n = k$. There are $2k$ vertices and thus $2k - 1$ edges in $T$. Therefore, there exists some vertex $b \in B$ with degree 1. Vertex $b$ is matched to its neighbor $a$. Since the degree of $a$ is at most 2, tree $T$ remains connected when $a$ and $b$ and all incident edges are removed from the graph. By the induction hypothesis, this subtree has a perfect matching. This matching, augmented with edge $\{a, b\}$ comprises a perfect matching for $T$ and thus for $H$. This argument indicates that a perfect matching can be found in linear time by repeatedly selecting a spare vertex of degree 1 and matching it to its neighbor. □

[3]We denote the graph in this way to distinguish it from the graph $G = (X \cup Y, E)$ which we reserve to represent a processor array.

*Lemma 4:* Let $H = (A \cup B, E)$ be a connected bipartite graph. If $H$ is acyclic and every vertex in $A$ has degree exactly 2, then for every $b \in B$ there exists a perfect matching of $A$ into $B - \{b\}$ and this matching can be found in linear time.

*Proof:* By Lemma 2, $|B| = |A| + 1$. Let $b$ be an arbitrary vertex in $B$ and let $C_1, \ldots, C_k$ be the connected components induced by the removal of $b$ from $H$. Let $A_i$ and $B_i$ denote the subsets of $A$ and $B$, respectively, in $C_i$, $1 \leq i \leq k$. We claim that $|A_i| = |B_i|$, $1 \leq i \leq k$. Assume otherwise. Then $|A_j| \neq |B_j|$ for some $j$, $1 \leq j \leq k$. By construction of the components, exactly one vertex in $A_i$ has degree 1 and all remaining vertices in $A_i$ have degree 2, for all $1 \leq i \leq k$. Thus, $C_i$ has exactly $2|A_j| - 1$ edges, $1 \leq i \leq k$, and therefore $C_j$ has exactly $2|A_j| - 1$ edges. If $|A_j| > |B_j|$ then $C_j$ has at most $2|A_j| - 1$ vertices and, since $C_j$ is acyclic, it has at most $2|A_j| - 2$ edges, a contradiction. Similarly, if $|A_j| < |B_j|$ then $C_j$ has at least $2|A_j| + 1$ vertices and thus at least $2|A_j|$ edges, again a contradiction. Therefore, $|A_i| = |B_i|$, for $1 \leq i \leq k$, and by Lemma 3 a matching in $C_i$ can be found in linear time. Consequently, a perfect matching can be found in $H$ in linear time. □

## IV. Acyclic Graphs

In this section we describe a linear time algorithm to find optimal matchings in acyclic bipartite graphs. Experimental results for this algorithm are given in Section IV–B.

### A. The Optimal Matching Algorithm

Let $G$ be an acyclic bipartite graph and let the root $r$ of $G$ be an arbitrary nonfaulty primary vertex. (If all primary vertices are faulty then any matching from $F$ to $Y$ is an optimal matching and matchings in acyclic graphs can be found in linear time by well-known algorithms [16].) Without loss of generality, we assume that $G$ is irreducible and connected, and therefore a tree. (If $G$ is not connected then each connected component can be treated individually.)

An example of an architecture with an acyclic bipartite graph is shown in Fig. 3(a). This is the fault tolerant modular binary tree architecture proposed by Hassan and Agarwal [7]. The corresponding irreducible bipartite graph $G$ is shown in Fig. 3(b).[4] The vertex $x_1$ has been arbitrarily selected as the root of $G$. Note that the 1-FT set induced by $y_6$ and its neighbors contains no nonfaulty vertices and therefore these vertices and their incident edges do not appear in the irreducible graph $G$.

Since a region is a connected subgraph of $G$, it is also a tree. The root of region $R$ is denoted by $\rho(R)$. For example, the two regions $R_1$ and $R_2$ for the graph $G$ in Fig. 3(b) are shown in Fig. 3(c). The roots of these regions are $x_1$ and $x_2$, respectively. The following lemma states two basic properties of regions in trees.

*Lemma 5:* Let $R$ be a region in tree $G$.

1) $\rho(R)$ is a nonfaulty primary vertex.
2) $\rho(R)$ has exactly one child in $R$.

[4]Note that although $D(X) = 2$ in this example, the algorithm presented here applies to acyclic graphs with arbitrary vertex degrees.

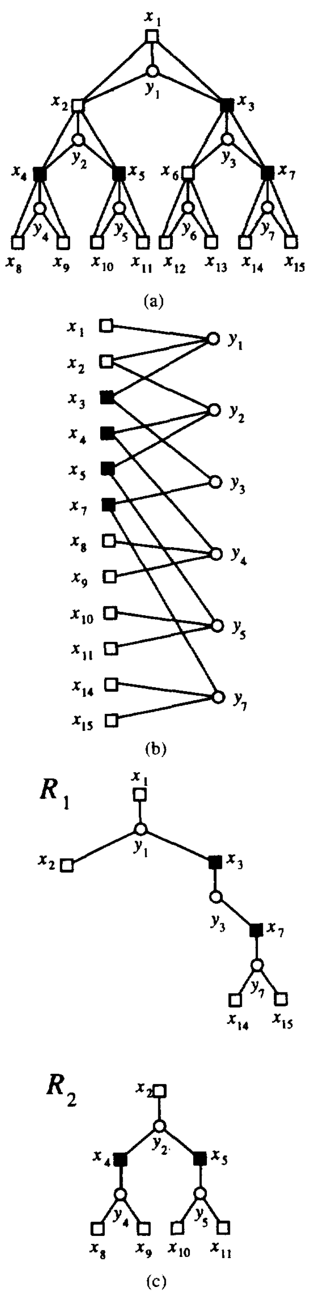


Fig. 3. (a) A fault tolerant modular binary tree. (b) The reduced bipartite graph. (c) The two regions.

*Proof:* First, assume that $\rho(R)$ is a spare vertex. By the definition of regions, every neighbor of a spare vertex is in $R$. In particular, the parent of $\rho(R)$ is in $R$, a contradiction. Next, assume that $\rho(R)$ is faulty. By the definition of regions, every spare vertex adjacent to a faulty vertex is in the same region as the faulty vertex. In particular, the parent of $\rho(R)$ is also in $R$, and thus $\rho(R)$ is not the root of $R$, a contradiction. Finally, assume that $\rho(R)$ has two children, $y_1$ and $y_2$. Since $G$ is a tree, $\rho(R)$ is the only common neighbor of spares $y_1$ and $y_2$. Since $\rho(R)$ is nonfaulty, $y_1$ and $y_2$ cannot belong to the same region. □

The optimal matching algorithm considers regions in a "bottom-up" fashion. To this end, we define the notion of one region being above or below another region. Formally, let $\mathcal{R}$ denote the set of regions of $G$ and let $\preceq_{\mathcal{R}}$ be a binary relation on $\mathcal{R}$ defined as follows. For $R_i, R_j \in \mathcal{R}$, $R_i \preceq_{\mathcal{R}} R_j$ iff $R_i = R_j$ or the path from $r$, the root of $G$, to $\rho(R_i)$ passes through some vertex of $R_j$ other than $\rho(R_j)$. If $R_i \prec_{\mathcal{R}} R_j$ then $R_i$ is said to be *below* $R_j$ and $R_j$ is said to be *above* $R_i$. For example, in Fig. 3(c), region $R_1$ is above region $R_2$ because the path from $x_1$ to $\rho(R_2) = x_2$ passes through vertex $y_1$. The relation $\preceq_{\mathcal{R}}$ is easily verified to be a partial ordering relation on $\mathcal{R}$.

Let $R \in \mathcal{R}$ be any minimal element of the partial order. That is, there exists no region that is below $R$. Such a region is called a *minimal region*. Each spare vertex $y$ in $R$ is assigned an ordered pair cost defined by $\text{cost}(y) = (a, b)$ where $a$ is the number of nonfaulty primary vertices adjacent to $y$ but not adjacent to any other spare vertices in $G$, and $b$ is the number of nonfaulty primary vertices adjacent to $y$ and adjacent to at least one other spare vertex in $G$. Intuitively, the cost of vertex $y$ tells us that if vertex $y$ is matched to a faulty vertex then $a$ nonfaulty vertices in $G$ will certainly become unrepairable and $b$ additional vertices in $G$ may become unrepairable, depending on the status of the other spare vertices. For example, in Fig. 3, $R_2$ is a minimal region. In $R_2$, $\text{cost}(y_2) = (0,1)$, $\text{cost}(y_4) = (2,0)$, and $\text{cost}(y_5) = (2,0)$.

Let $\text{cost}(y_i) = (a_i, b_i)$. We note that if the property $b_i = 0$ holds for every spare vertex $y_i$, then the number of unrepairable vertices with respect to a given matching is simply $\sum_{y_i \in S} a_i$ where $S$ denotes the set of matched spare vertices. Unfortunately, this property is generally not satisfied. However, the following lemma suggests that in a minimal region, this property is "almost" satisfied.

*Lemma 6:* Let $R$ be a minimal region and let s be the unique child of $\rho(R)$ in $R$. For each spare processor $y_i$ in $R$, let $\text{cost}(y_i) = (a_i, b_i)$. If $y_i \neq s$ then $b_i = 0$. If $y_i = s$ then $b_i = 1$ if $\rho(R)$ contains a neighbor other than $s$ in $G$ and $b_i = 0$ otherwise.

*Proof:* If $y_i \neq s$, then by the definition of regions, the parent of $y_i$ must be faulty. Assume that $b_i > 0$. Then some nonfaulty child of $y_i$, $x$, must have at least one child. Observe that $x$ cannot have any faulty descendants, since otherwise $R$ would not be minimal. On the other hand, $x$ cannot have only nonfaulty descendants, since otherwise $G$ is not irreducible, a contradiction. if $y_i = s$, all nonfaulty children of $y_i$ must again be adjacent to no other spares. Thus, if $\rho(R)$ is adjacent to a spare vertex other than $s$, $b_i = 1$ and $b_i = 0$ otherwise. □

We define addition on costs by $(a_1, b_1) + (a_2, b_2) = (a_1 + a_2, b_1 + b_2)$. We let $\preceq_{\text{lex}}$ be the lexicographic ordering on costs. That is, $(a_1, b_1) \preceq_{\text{lex}} (a_2, b_2)$ iff $a_1 < a_2$ or $a_1 = a_2$ and $b_1 \leq b_2$. Let $M_R$ be a matching of all faulty primary vertices in $R$ to spare vertices in $R$. The *cost* of $M_R$ is defined to be $(a_1, b_1) + \cdots + (a_n, b_n)$ where $y_1, \ldots, y_n$ are the spare vertices matched in $M_R$ and $(a_i, b_i)$ is the cost of spare vertex $y_i$. Observe that $\preceq_{\text{lex}}$ is a total ordering on the costs of matchings. Thus, a *minimum cost matching* in region $R$ is a matching of

```
Input: Acyclic bipartite graph $G = (X \cup Y, E)$ and set $F$.
Output: Optimal matching for $F$ in $G$.

begin
  $M_{\text{opt}} = \emptyset$.
  Find all regions using breadth-first search.
  Use topological sort to order the regions.
  while regions remain do
    Select a minimum region $R$.
    Compute costs for each spare vertex in $R$.
    Find a minimum cost matching $M$ in $R$.
    $M_{\text{opt}} = M_{\text{opt}} \cup M$.
    Remove $R$ from $G$.
  endwhile
  return($M_{\text{opt}}$).
end
```

Fig. 4. Optimal matching algorithm for acyclic graphs.

all faulty primary vertices to spare vertices in $R$ of minimum cost with respect to $\preceq_{\text{lex}}$.

The optimal matching algorithm is based on the following property of minimum cost matchings: Any minimum cost matching in a minimal region of a tree is a subset of an optimal matching. This property implies that an optimal matching may be constructed by repeatedly selecting a minimal region and finding a minimum cost matching in the region, until all faulty primary vertices have been matched to spare vertices.

*Theorem 2:* Let $R$ be a minimal region in tree $G$ and let $M_{R,\min}$ be any minimum cost matching in $R$. The matching $M_{R,\min}$ is a subset of an optimal matching for $G$.

*Proof:* Let $M$ be an optimal matching for $G$ and let $M_R$ be the subset of $M$ restricted to $R$. Let $(\alpha_1, \beta_1)$ and $(\alpha_2, \beta_2)$ be the costs of $M_R$ and $M_{R,\min}$, respectively. Let $u$ denote the number of unrepairable vertices outside of $R$ with respect to $M$. Let $M'$ be the matching obtained from $M$ by replacing the edges in $M_R$ by the edges in $M_{R,\min}$. We claim that the number of unrepairable vertices with respect to $M'$ is no larger than the number of unrepairable vertices with respect to $M$. Assume that $\rho(R)$ is adjacent to un unmatched spare with respect to $M$ outside of $R$. Then $\rho(R)$ is repairable with respect to both $M$ and $M'$. From Lemma 6, it follows that the number of unrepairable vertices with respect to $M$ and $M'$ is $u + \alpha_1$ and $u + \alpha_2$, respectively. Since $\alpha_2 \leq \alpha_1$, we conclude that $M'$ is an optimal matching. Next, assume that $\rho(R)$ is adjacent to at least one spare vertex outside $R$ but all such spares are matched with respect to $M$. Then $\rho(R)$ is repairable only if its child in $R$ is unmatched. Thus, the number of unrepairable vertices in $M$ and $M'$ is $u + \alpha_1 + \beta_1$ and $u + \alpha_2 + \beta_2$, respectively. By Lemma 6, $0 \leq \beta_1, \beta_2 \leq 1$, and thus $\alpha_2 + \beta_2 \leq \alpha_1 + \beta_1$. We again conclude that $M'$ is an optimal matching. Finally, assume that $\rho(R)$ has degree 1 in $G$. It follows from Lemma 6, that there are $u + \alpha_1$ and $u + \alpha_2$ unrepairable vertices with respect to $M$ and $M'$, respectively. Since $\alpha_2 \leq \alpha_1$, $M'$ is an optimal matching. □

Theorem 2 suggests the following algorithm for finding optimal matchings in acyclic graphs. First, all the regions in the graph are found using breadth-first search. A minimal region is then selected, the costs of spare vertices in the region are determined, and a minimum cost matching is found in this region. This step is repeated until no more regions remain. The algorithm is summarized in Fig. 4.

We now show that a minimum cost matching in a minimal region can be found by an efficient algorithm. By the irreducibility assumption, every faulty vertex $x$ in minimal region $R$ has at least one child. Moreover, by Lemma 5, $\rho(R)$ is nonfaulty and thus every faulty vertex in $R$ has a parent in $R$. The following theorem is the basis for finding minimum cost matchings in minimal regions in linear time.

*Theorem 3:* Let $x$ be a faulty primary vertex in $R$ in which all faulty descendants of $x$ have exactly one child. Let $y_1, y_2, \ldots, y_k$ denote the children of $x$ and let $c_i$ denote the maximum cost among $y_i$ and all of its descendants where $c_i \preceq_{\text{lex}} c_{i+1}$, for $1 \leq i \leq k-1$. If there exists a matching in $R$ then there exists a minimum cost matching in $R$ in which $x$ is not matched to any of $y_2, \ldots, y_k$.

*Proof:* Assume that in every minimum cost matching in $R$, $x$ is matched to one of $y_2, \ldots, y_k$. Since each faulty descendant of $y_i$ has exactly one child and one parent, by Lemma 2 the subtree of $R$ rooted at $y_i$, for each $1 \leq i \leq k$, has one more spare vertex than faulty primary vertices. Let $M_{R,\min}$ be a minimum cost matching for $R$ in which $x$ is matched to $y_j$, for some $2 \leq j \leq k$. From the observation above, every spare vertex in the subtree rooted at $y_j$ is matched in $M_{R,\min}$. Let $y'$ denote the spare vertex in subtree rooted at $y_j$ with cost $c_j$. Since $x$ is not matched to $y_1$, there is one unused spare vertex, $y''$, in the subtree rooted at $y_1$. Let $c''$ denote the cost of $y''$. Let $P$ denote the unique path in $R$ from $y'$ to $y''$. Since $P$ is an alternating path, a new matching $M_{R,\text{new}}$ is obtained by making each matched edge in $P$ unmatched and making each unmatched edge a matched edge. Since $y''$ is in the subtree rooted at $y_1$, vertex $x$ is matched to $y_1$ in $M_{R,\text{new}}$. Moreover, the cost of $M_{R,\text{new}}$ is no larger than the cost of $M_{R,\min}$ since $c'' \preceq_{\text{lex}} c_1 \preceq_{\text{lex}} c_j$, contradicting the assumption that no minimum cost matching in $R$ matches $x$ to $y_1$. □

Let $x$ be a faulty primary vertex in $R$ in which all faulty descendants have one child. Such a vertex always exists, since any faulty vertex in $R$ with no faulty descendants has this property. Let $y_1, \ldots, y_k$ be the children of $x$ as in Theorem 3. Let $T_i$ denote the subtree rooted at $y_i$ and let $T$ denote the subtree obtained by removing subtrees $T_2, \ldots, T_k$ from $R$. Each faulty primary vertex in $T_2, \ldots, T_k$ has degree 2. By Theorem 3, if a matching in $R$ exists then a minimum cost matching can be obtained as the union of minimum cost matchings in $T$ and $T_2, \ldots, T_k$. This partitioning can now be applied to $T$ and repeated until the faulty vertices in each subtree have degree 2.

Lemma 4 tells us that in each subtree constructed above, a perfect matching is induced by the removal of any single spare vertex. Therefore, a minimum cost matching in a subtree is obtained by removing the vertex with maximum cost.

We now illustrate the execution of this algorithm for the faulty tree of Fig. 3(a). First, regions $R_1$ and $R_2$ shown in Fig. 3(c) are found. Region $R_2$ is minimal. Each faulty vertex in $R_2$ has degree 2. Therefore, a spare vertex in $R_2$ with maximum cost is selected from $R_2$ to remain unmatched. Either $y_4$ or $y_5$ may be selected. Assume that $y_4$ is selected.

Then $x_4$ is matched to $y_2$ and $x_5$ is matched to $y_5$. Next, region $R_1$ is considered. Again, each faulty vertex already has degree 2. Therefore, a spare vertex of maximum cost is selected to remain unmatched. Either $y_1$ or $y_7$ may be selected. If $y_1$ is selected, $x_3$ is matched to $y_3$ and $x_7$ is matched to $y_7$. This optimal matching leaves four vertices unrepairable. In contrast, the worst possible matching for this faulty tree leaves seven vertices unrepairable.

Finally, the time complexity of the algorithm is linear in the number of vertices in the tree. To see this, note that the regions can be found in linear time by breadth-first search. Minimal regions can be identified in constant time by maintaining a topological ordering of the regions (which is precomputed in linear time). Finally, a minimum cost matching can be found in each minimal region by partitioning the region into subtrees in linear time and finding minimum cost matchings in each subtree, which again requires only linear time.

### B. Experimental Results

The optimal matching algorithm for acyclic graphs was compared to the standard matching algorithm for acyclic graphs [16] on fault tolerant modular binary trees with 64 primary processors and 32 spare processors.[5] Both algorithms run in linear time. However, the optimal algorithm found assignments with substantially fewer unrepairable processors than those found by the standard matching algorithm. In our experiment, trees were generated with faulty elements introduced at random according to a uniform distribution. The fault frequency ranged from 0% to 30% in increments of 5%. (At fault frequency 35% there were already fewer spare processors than faulty processors, and thus no assignments could be found.) For each fault frequency, 100 faulty trees were generated. When the fault frequency was 15%, the optimal algorithm found assignments in which an average of over 96% of the nonfaulty processors were repairable. In contrast, the standard matching algorithm found assignments in which only 88% of the nonfaulty processors were repairable on average. When the fault frequency increased to 30%, the results were even more striking: The optimal algorithm left nearly 80% of the nonfaulty processors repairable while the standard matching algorithm left fewer than 43% of the nonfaulty processors repairable on average. The results are summarized in Fig. 5.

## V. Degree Bounds on Primary Processors

In this section we characterize the complexity and give algorithms for the safe and optimal matching problems when the primary processors have a bounded number of adjacent spares. Recall that $D(X)$ represent the maximum degree among all vertices in $X$, that is, $D(X)$ is the maximum number of spare processors that are connected to a primary processor. In Section V–A we show that the safe matching problem can be solved in polynomial time when $D(X) \leq 2$. Moreover, the safe matching problem becomes NP-complete when $D(X) = 3$. We also show that the optimal matching problem remains NP-complete even when $D(X) \leq 2$. However, some architectures have additional properties which may be exploited to solve the optimal matching problem efficiently. As an example, in Section V–B we give a linear time algorithm for finding optimal assignments in the augmented mesh architecture proposed in [11]. Experimental results are given in Section V–C.

[5]All simulations described in this paper were implemented in C and C++ on a Sun SparcStation and an Encore Multimax computer.

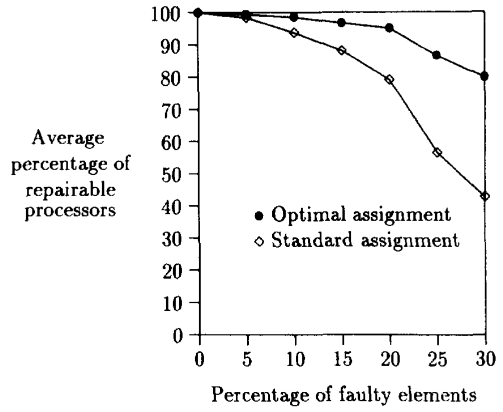


Fig. 5. Optimal versus standard assignments in Singh's tree architecture.

### A. Algorithms and Complexity

We begin by showing that the safe matching problem can be solved by an efficient algorithm when $D(X) \leq 2$. Clearly, for some patterns of faults there exists an optimal matching but no safe matching. However, experimental results given in Section V–C suggest that when the number of faulty processors is not too large, most optimal matchings are in fact safe matchings and thus our algorithm can be used. Moreover, in systems in which high reliability is of the greatest concern, we may insist on matchings to be safe.

We transform the safe matching problem to the 2-satisfiability problem (2SAT) which can be solved in polynomial time by any of several algorithms [4], [5]. 2SAT is defined as follows. Given a set $U$ of variables and a collection $C$ of clauses over $U$ such that each clause $c \in C$ has $|c| = 2$, is there a truth assignment for $U$ that satisfies all of the clauses in $C$?

For every edge $\{x_i, y_j\} \in E$ where $x_i \in F$, we introduce the boolean variable $u_{i,j}$. For every edge $\{x_i, y_j\} \in E$ where $x_i \in N$, we introduce the boolean variable $v_{i,j}$. We let $u_{i,j} =$ **true** represent the situation in which $\{x_i, y_j\}$ is in the matching and $u_{i,j} =$ **false** represent the situation in which $\{x_i, y_j\}$ is not in the matching. Similarly, we let $v_{i,j} =$ **true** represent the situation in which vertex $x_i \in N$ is repairable by vertex $y_j$ and $v_{i,j} =$ **false** represent the situation in which $x_i$ is not repairable by vertex $y_j$. The set $C$ is obtained from the following construction.

*Construction 1:*

$T1$. $(u_{i,j} \vee u_{i,k})$ for each vertex $x_i \in F$ adjacent to vertices $y_j$ and $y_k$.

$T2$. $(\neg u_{i,j} \vee \neg u_{i,k})$ for each vertex $x_i \in F$ adjacent to vertices $y_j$ and $y_k$.

$T3$. $(\neg u_{i,j} \vee \neg u_{h,j})$ for each pair of vertices $x_i, x_h \in F$ adjacent to a common vertex $y_j$.

$T4$. $(v_{i,j} \vee v_{i,k})$ for each vertex $x_i \in N$ adjacent to vertices $y_j$ and $y_k$.

$T5$. $(\neg v_{i,j} \vee \neg u_{h,j})$ for every vertex $x_i \in N$ and vertex $x_h \in F$ adjacent to a common vertex $y_j$.

Since there are only a polynomial number of variables, only a polynomial number of clauses are introduced by this construction.

*Theorem 4:* The instance of 2SAT from Construction 1 is satisfiable iff there exists a matching for $F$ in which all vertices in $N$ are repairable.

*Proof:* Consider a satisfying assignment for the set of clauses. For each vertex $x_i \in F$, we include $\{x_i, y_j\}$ in matching $M$ if $u_{i,j} = \textbf{true}$. Each vertex $x_i \in F$ is matched to at least one vertex in $Y$ since the clauses in $T1$ are satisfied. Moreover, each vertex $x_i \in F$ is not matched to two vertices in $Y$ since the clauses in $T2$ are satisfied. A vertex $y_j \in Y$ is matched to at most one vertex in $F$ since the clauses in $T3$ are satisfied. Therefore, each vertex in $F$ is matched to a unique vertex in $Y$. Each vertex $x_i \in N$ is repairable by a vertex in $Y$ since the clauses in $T4$ are satisfied and a vertex in $N$ is not repairable by a vertex in $Y$ that is matched to a vertex in $F$ since the clauses in $T5$ are satisfied.

Conversely, assume that matching $M$ for $F$ has $|N|$ repairable vertices. For each variable $u_{i,j}$, assign $u_{i,j} = \textbf{true}$ if vertex $x_i \in F$ is matched to $y_j$ and assign $u_{i,j} = \textbf{false}$ otherwise. Similarly, for each variable $v_{i,j}$, assign $v_{i,j} = \textbf{true}$ if vertex $x_i \in N$ is repairable by vertex $y_j$ and assign $v_{i,j} = \textbf{false}$ otherwise. Each clause in $T1$ is satisfied since every vertex in $F$ is matched. Each clause in $T2$ is satisfied since a vertex in $F$ cannot be matched to two vertices in $Y$. Each clause in $T3$ is satisfied since a vertex in $Y$ is matched to a single vertex in $F$. Each clause in $T4$ is satisfied since each vertex in $N$ is repairable. Finally, each clause in $T5$ is satisfied since a vertex in $N$ is only repairable by a vertex $y \in Y$ if $y$ is unmatched. □

The algorithm is summarized in Fig. 6.

This result is tight in the sense that the safe matching problem becomes NP-complete when $D(X) = 3$. This follows from the observation that in the proof of Theorem 1 every vertex in $X$ has degree at most 3. Therefore, we have the following corollary to Theorem 1.

*Corollary 2:* SMDP is NP-complete even if $D(X) = 3$.

Finally, we observe that the result of Theorem 2 cannot be extended to optimal matchings.

*Theorem 5:* OMDP is NP-complete even if $D(X) \leq 2$.

*Proof:* The proof is similar to the proof of Theorem 1 except that the reduction is now from the Maximum 2-Satisfiability Problem (MAX 2SAT). MAX 2SAT is defined as follows [6]. Given a set $U$ of variables, a collection $C$ of clauses over $U$ such that each clause $c \in C$ has $|c| = 2$, and a positive integer $K \leq |C|$, is there a truth assignment for $U$ that simultaneously satisfied at least $K$ of the clauses in $C$?

```
Input: Bipartite graph $G = (X \cup Y, E)$ and set $F$ where $D(X) \leq 2$.
Output: Safe matching for $F$ in $G$.

begin
  Construct an instance of 2SAT as in Construction 1.
  Determine a satisfying assignment for the 2SAT instance, if one exists,
    using a linear-time algorithm [12].
  if 2SAT instance unsatisfiable halt
  for each boolean variable $u_{i,j}$ with value true do
    Match $x_i$ to $y_j$;
  endfor
end
```

Fig. 6. Safe matching algorithm for $D(X) \leq 2$.

The reduction from the MAX 2SAT instance to an instance of OMDP proceeds as in the proof of Theorem 1 except that now each clause $c \in C$ contains only two literals and thus corresponds to a vertex $x_c$ in $N$ adjacent to only two vertices in $Y$.

We claim that there is a truth assignment that satisfies at least $K$ of the clauses in $C$ iff there is a matching for $F$ with at least $K$ repairable vertices. The proof of this claim is analogous to the proof of Theorem 1. □

## *B. Augmented Mesh*

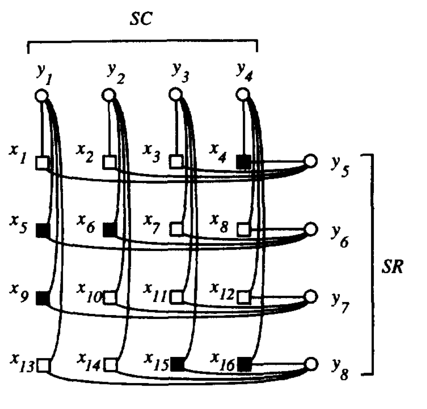
Although we have shown that OMDP is NP-complete for $D(X) \leq 2$, some architectures have topological properties that can be exploited to find efficient algorithms for OMDP. In this section we give an example of an augmented mesh architecture in which $D(X) = 2$ and show how OMDP can be solved in linear time for this architecture.

An *augmented mesh* is a $\sqrt{t} \times \sqrt{t}$ grid of primary vertices such that each row of primary vertices shares a common spare vertex and each column of primary vertices shares a common spare vertex. The $\sqrt{t}$ spare vertices for rows and columns are called *spare row vertices* and *spare column vertices*, respectively. For example, a $4 \times 4$ augmented mesh with 6 faulty primary processors is shown in Fig. 7(a).

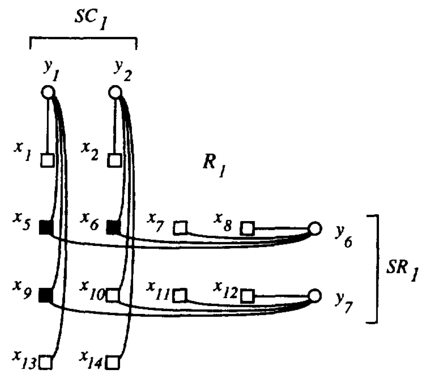
The optimal reconfiguration problem for augmented meshes was proposed in [11]. In [3] an $O(t)$ algorithm for OMDP was given for this architecture. In this section we describe a much simpler $O(\sqrt{t})$ algorithm.

Let $G = (X \cup Y, E)$ be a $\sqrt{t} \times \sqrt{t}$ augmented mesh where $X$ denotes the set of primary vertices, $Y$ denotes the set of spare vertices, and $E$ denotes the set of edges between primary vertices and spare vertices. The set $Y$ of spare processors is partitioned into two sets, SR and SC, denoting the spare row vertices and spare column vertices, respectively.

Let $R_1, \ldots, R_k$ denote the regions in $G$. For example, the graph in Fig. 7(a) has two regions, one of which, $R_1$, is shown in Fig. 7(b). For each region, $R_i$, let $X_i$ denote the set of primary vertices in $R_i$ and let $Y_i$ denote the set of spare vertices in $R_i$. In addition, let $F_i \subseteq X_i$ denote the set of faulty primary vertices in $R_i$ and let SR$_i$, SC$_i \subseteq Y_i$ denote the set of spare row vertices and spare column vertices in $R_i$, respectively. By Lemma 1, $R_i$ remains connected when all nonfaulty vertices are removed from $R_i$, leaving only the vertices in $F_i$ and $Y_i$. Since each vertex in $F_i$ has degree exactly 2 by the irreducibility assumption, from Lemma 2 it follows that $|F_i| \geq |Y_i| - 1$. In other words, each region


SC
$y_1$ $y_2$ $y_3$ $y_4$
$x_1$ $x_2$ $x_3$ $x_4$ $y_5$
$x_5$ $x_6$ $x_7$ $x_8$ $y_6$
$x_9$ $x_{10}$ $x_{11}$ $x_{12}$ $y_7$
$x_{13}$ $x_{14}$ $x_{15}$ $x_{16}$ $y_8$
SR

(a)


$SC_1$
$y_1$ $y_2$
$x_1$ $x_2$
$R_1$
$x_5$ $x_6$ $x_7$ $x_8$ $y_6$
$x_9$ $x_{10}$ $x_{11}$ $x_{12}$ $y_7$
$SR_1$
$x_{13}$ $x_{14}$

(b)

Fig. 7. (a) A 4 × 4 augmented mesh. (b) A region.

contains at most one more spare vertex than faulty vertex. If $|F_i| > |Y_i|$ then there are more faulty vertices than spares in region $R_i$ and thus some faulty vertices cannot be replaced. If $|F_i| = |Y_i|$ then by Lemma 3 there exists a perfect matching of $F_i$ into $Y_i$ that can be found in linear time. Therefore, we must only consider the case that each region has the property $|F_i| = |Y_i| - 1$. In this case, the graph obtained by removing all nonfaulty primary vertices from $R_i$ contains $2|F_i| + 1$ vertices and $2|F_i|$ edges and is therefore a tree. By Lemma 4, a perfect matching can be found from $F_i$ to $Y_i - \{y\}$ for any $y \in Y_i$. It now only remains to be shown how a spare vertex, $y$, from each set $Y_i$ should be selected to be left unmatched such that the total number of unrepairable vertices is minimized.

Assume that $|F_i| = |Y_i| - 1$ for each region $R_i$, $1 \leq i \leq k$. Let $y$ be a spare vertex and let $n(y)$ denote the number of nonfaulty vertices adjacent to $y$. Let $r_i \in \mathrm{SR}_i$ be a spare row vertex with the property $n(r_i) \geq n(y)\ \forall y \in \mathrm{SR}_i$ and let $c_i \in \mathrm{SC}_i$ be a spare column vertex with the property $n(c_i) \geq n(y)\ \forall y \in \mathrm{SC}_i$. That is, $r_i$ is the spare row vertex in region $R_i$ adjacent to the largest number of nonfaulty vertices and $c_i$ is the spare column vertex in region $R_i$ adjacent to the largest number of nonfaulty vertices. For example, in region $R_1$ in Fig. 7(b), $r_1 = y_7$ and $c_1 = y_2$. Intuitively, either of $r_i$ or $c_i$ are good candidates to remain unmatched since many nonfaulty vertices are repairable with respect to these vertices. Indeed, the following lemma suggests that in each region $R_i$ we may restrict our attention to the vertices $r_i$ and $c_i$.

*Lemma 7:* Let $M$ be a matching in $G$ and let $u_i$ denote the unmatched vertex in $R_i$. If $u_i \in SR_i$ ($u_i \in SC_i$) then by replacing $u_i$ by $r_i(c_i)$ as the unmatched vertex, the number of unrepairable vertices does not increase.

*Proof:* Without loss of generality, assume that $u_i \in \mathrm{SR}_i$ and $u_i \neq r_i$. Since $u_i$ is a spare row vertex and it is the only unmatched spare vertex in $R_i$, every spare column vertex in $R_i$ must be matched in $M$. Let $N_{u_i}$ and $N_{r_i}$ denote the set of nonfaulty neighbors or $u_i$ and $r_i$, respectively. Since $n(r_i) \geq n(u_i)$, $|N_{r_i}| \geq |N_{u_i}|$. We partition $N_{u_i}$ into two sets, $A_{u_i}$ and $B_{u_i}$, such that $A_{u_i}$ contains those vertices whose corresponding spare column vertices are not in $R_i$ and those that are in $R_i$, respectively. Similarly, we partition $N_{r_i}$ into two sets $A_{r_i}$ and $B_{r_i}$. Note that $|A_{r_i}| = |A_{u_i}|$ since every spare column vertex not in $R_i$ cannot have a faulty vertex in common with a spare row vertex in $R_i$. Thus, $|B_{r_i}| \geq |B_{u_i}|$. Therefore, the net increase in the number of unrepairable vertices introduced by making $u_i$ matched and $r_i$ unmatched is $|B_{u_i}| - |B_{r_i}| \leq 0$. □

We must now determine which of the spares $r_i$ or $c_i$ should be left unmatched in each region $R_i$. Let $d_i = n(r_i) - n(c_i)$. Let $\pi$ be a permutation of $\{1, \ldots, k\}$ such that $d_{\pi(1)} \geq d_{\pi(2)} \geq \cdots \geq d_{\pi(k)}$. That is, $\pi$ is the permutation that sorts the $d_i$ values in nonincreasing order.

*Lemma 8:* Among all matchings with exactly $\ell$ unmatched spare row vertices, the matching obtained by leaving vertices $r_{\pi(1)}, \ldots, r_{\pi(\ell)},\ c_{\pi(\ell+1)}, \ldots, c_{\pi(k)}$ unmatched has the least number of unrepairable vertices.

*Proof:* Let $M$ be the matching obtained by leaving vertices $r_{\pi(1)}, \ldots, r_{\pi(\ell)}, c_{\pi(\ell+1)}, \ldots, c_{\pi(k)}$ unmatched and let $M'$ be the matching obtained by leaving vertices $r_{\sigma(1)}, \ldots, r_{\sigma(\ell)}$, $c_{\sigma(\ell+1)}, \ldots, c_{\sigma(k)}$ unmatched, where $\sigma$ is any permutation of $\{1, \ldots, k\}$. (By Lemma 7, we may restrict our attention to such matchings without loss of generality.) The number of repairable vertices with respect to $M$ is

$$S_M = \sum_{i=1}^{\ell} n(r_{\pi(i)}) + \sum_{j=\ell+1}^{k} n(c_{\pi(j)}) - \ell \times (k - \ell)$$

since each nonfaulty vertex adjacent to both an unmatched spare row vertex and an unmatched spare column vertex is counted twice and there are $\ell \times (k - \ell)$ such vertices. Similarly, the number of repairable vertices with respect to $M'$ is

$$S_{M'} = \sum_{i=1}^{\ell} n(r_{\sigma(i)}) + \sum_{j=\ell+1}^{k} n(c_{\sigma(j)}) - \ell \times (k - \ell).$$

Let $C = \sum_{i=1}^{k} n(c_i) - \ell \times (k - \ell)$. Then,

$$S_M \geq S_{M'} \iff S_M - C \geq S_{M'} - C \iff \sum_{i=1}^{\ell} d_{\pi(i)} \geq \sum_{i=1}^{\ell} d_{\sigma(i)}$$

By the definition of $\pi$, $d_{\pi(i)} \geq d_{\sigma(i)}$, $1 \leq i \leq k$. Thus, $S_M \geq S_{M'}$ and thus matching $M$ has the least number of unrepairable vertices among all matchings with exactly $\ell$ unmatched spare row vertices. □

```
Input: Augmented mesh $G = (X \cup Y, E)$ and set $F$.
Output: Optimal matching for $F$ in $G$.

begin
  Find regions in $G$. /* Let $k$ denote number of regions */
  For each region, $R_i$, find $r_i$, $c_i$, and $d_i$.
  Sort the $d_i$ values using a linear time sorting algorithm.
  for $\ell = 0$ to $k$ do
    Compute minimum number of unrepairable vertices in a matching
      with exactly $\ell$ unmatched row vertices.
  endfor
  Select the value of $\ell$ minimizing number of unrepairable vertices
    and remove the corresponding row and column spares.
  Use algorithm of Lemma 3 to find matchings in each region.
end
```

Fig. 8. Optimal matching algorithm for the augmented mesh.

An optimal matching can now be obtained by finding the matching with the least number of unrepairable vertices for each $\ell$, $1 \leq \ell \leq k$, and selecting the matching with the minimum number of unrepairable vertices among these matchings.

By employing simple data structures, the time complexity of this algorithm is seen to be $O(\sqrt{t})$. Since there are only $2\sqrt{t}$ spare vertices, if the number of faulty processors exceeds this amount the algorithm reports that the array is not repairable and halts. Otherwise, two arrays are created, *row* and *column*, such that the $i$th element of *row* contains a pointer to a linked list of faulty processors in the $i$th row of the grid and the $j$th element of *column* contains a pointer to a linked list of faulty processors in the $j$th column of the grid. Thus, each fault is inserted once in each array. Since each array has size $O(\sqrt{t})$ and there are $O(\sqrt{t})$ elements to be inserted in the arrays, construction of these arrays takes time $O(\sqrt{t})$. With these data structures, the regions can easily be found in time $O(\sqrt{t})$ [17]. Sorting the $d_i$ values can be performed in time $O(\sqrt{t})$ using a linear time sorting algorithm since $|d_i| \leq \sqrt{t}$. For each value of $\ell$, $0 \leq \ell \leq k \leq \sqrt{t}$, the number of unrepairable vertices in the best matching with $\ell$ unmatched spare row vertices can be computed in constant time. Consequently, the running time of the algorithm is $O(\sqrt{t})$. The algorithm is summarized in Fig. 8.

Chen *et al.* [3] conjectured that similar polynomial time algorithms could be found for OMDP in augmented meshes of higher dimensions. We have "disproved" this conjecture by showing that the existence of such an algorithm, even for three-dimensional meshes, implies that $P = \text{NP}$. The proof of this theorem is rather involved and the interested reader is referred to [9].

### C. Experimental Results

Although the optimal matching problem is NP-complete for $D(X) \leq 2$, experimental results indicate that for low to moderate fault frequencies most optimal assignments are in fact safe assignments. Therefore, in these cases the polynomial time safe matching algorithm for $D(X) \leq 2$ can be employed to find optimal matchings with high probability of success. For example, in a simulation on 100 randomly generated arrays with 32 primary processors, 16 spare processors, and $D(X) = 2$, more than 86% of the optimal assignments found were safe assignments when at most 10% of the processors were faulty. (Optimal assignments were obtained by exhaustive search.)

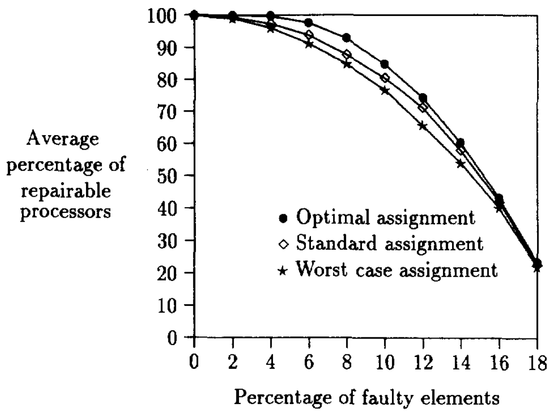


Fig. 9. Optimal, standard, and worst case assignments in the augmented mesh.

Next, we consider the optimal matching algorithm for augmented meshes. The algorithm was implemented and tested on 10 × 10 meshes (20 spares) with faulty processors introduced at random according to a uniform distribution. The fault frequency ranged from 0% to 18% in increments of 2% and for each fault frequency 100 meshes were generated. (At fault frequency of 20%, none of the meshes could be repaired.) The optimal matching algorithm found solutions with as much as 5% more repairable processors than those found by standard matching and as much as 10% more repairable processors than worst case matchings. Since the optimal matching algorithm runs in linear time, no additional cost is incurred in obtaining optimal assignments. The results are summarized in Fig. 9.

## VI. Degree Bounds on Spare Processors

In this section we describe a linear time algorithm for finding optimal matchings when $D(Y) \leq 2$. We then show that this result is tight in the sense that the problem remains NP-complete when $D(Y) = 3$. In Section VI-B, experimental results are given for the optimal matching algorithm for $D(Y) \leq 2$.

### A. Algorithms and Complexity

Let $G = (X \cup Y, E)$ be a bipartite graph with $D(Y) \leq 2$. Without loss of generality, we assume that $G$ is connected, since otherwise each connected component can be considered independently.

*Lemma 9:* Assume that $D(Y) \leq 2$. If $|Y| = |X| + m, m \geq 0$, then there exists $S \subset Y$ such that $|S| = m$ and the removal of $S$ from $G$ does not disconnect $G$. Moreover, the set $S$ can be found in linear time.

*Proof:* Let $\ell = |X|$ and let $T$ be a spanning tree in $G$. We claim that $T$ has at least $m$ leaves in $Y$. Assume otherwise. Then $T$ has $m - j$ leaves in $Y$, for some $j \geq 1$. There are $2\ell + m$ vertices and thus $2\ell + m - 1$ edges in $T$. Also, there are $(\ell + m) - (m - j)$ vertices of degree 2 in $Y$ in tree $T$. Thus, there are

$$2[(\ell + m) - (m - j)] + (m - j) = 2\ell + m + j$$

edges in $T$, a contradiction. Thus, we conclude that there are at least $m$ leaves in $Y$ in tree $T$. Any set $S$ of $m$ leaves can be removed from $T$, resulting in a subtree $T'$. The set $S$ can be found in linear time by constructing a spanning tree and then identifying the first $m$ leaves in the tree. □

By Lemma 9, if $G$ has at least as many spare vertices as primary vertices then it can be reduced to a connected graph with exactly the same number of spare vertices as primary vertices. Then, by Lemma 3, a perfect matching of primary vertices to spare vertices can be found in this reduced graph in linear time. Observe that this matching assigns each faulty vertex in $F$ to a unique spare vertex in $Y$ and assigns each nonfaulty vertex in $N$ to a unique spare vertex in $Y$. Therefore, this matching is clearly an optimal matching for $G$.

We now turn our attention to the remaining case in which $|Y| < |X|$. By Lemma 2, $|Y| = |X| - 1$. Therefore, $G$ must be a tree since there are exactly $2|X| - 1$ vertices and $2|X| - 2$ edges. An optimal matching can be found in $G$ in linear time using the algorithm presented in Section III.

Finally, we show that this result is tight in the sense that the safe matching problem, and consequently the optimal matching problem, remains NP-complete when $D(Y) = 3$.

*Theorem 6:* SMDP is NP-complete for $D(Y) = 3$.

*Proof:* The reduction is again from 3SAT. We transform an instance of 3SAT, given by a set of variables $U$ and a set of clauses $C$, to an instance of OMDP with $D(Y) = 3$ as follows. For each variable $u \in U$ which occurs $k$ times in the clauses in $C$, there are:

1) $k$ vertices $x_{u,1}, \ldots, x_{u,k} \in F$ representing the $k$ occurrences of variable $u$.
2) $2k$ vertices $y_{u,1,T}, y_{u,1,F}, \ldots, y_{u,k,T}, y_{u,k,F} \in Y$ representing the **true** and **false** values for each occurrence of variable $u$.
3) $k$ vertices $z_{u,1}, \ldots, z_{u,k} \in N$ which are "enforcing" vertices that are used to ensure that all $k$ occurrences of variable $u$ are assigned the same boolean value.
4) $2k$ edges $\{x_{u,1}, y_{u,1,T}\}, \{x_{u,1}, y_{u,1,F}\}, \ldots, \{x_{u,k}, y_{u,k,T}\}, \{x_{u,k}, y_{u,k,F}\} \in E$.
5) $2k$ edges $\{z_{u,i}, y_{u,i,T}\}$ for $1 \le i \le k$, $\{z_{u,i}, y_{u,i+1,F}\}$ for $1 \le i \le k-1$, and $\{z_{u,k}, y_{u,1,F}\}$.

In addition, for each clause $c \in C$, where $c$ contains literals, $s, t$, and $r$, there is a vertex $x_c \in N$. Let $v$ denote the variable for literal $s$ and assume $s$ represents the $i$th occurrence of variable $v$. If $s$ is the unnegated variable $v$ then there is an edge $\{x_c, y_{v,i,F}\} \in E$ and if $s$ is $\neg v$ then there is an edge $\{x_c, y_{v,i,T}\} \in E$. Similarly, edges are added for literals $t$ and $r$. Observe that in this construction vertices in $Y$ have degree at most 3.

We claim that there is a truth assignment that satisfies all the clauses of $C$ iff there is a matching for $F$ in which all vertices in $N$ are repairable. We observe that in any matching for $F, M$, with no unrepairable vertices, the following property holds. For each variable $u \in U$, either all edges $\{x_{u,i}, y_{u,i,T}\}$, $1 \le i \le k$, are in $M$ or all edges $\{x_{u,i}, y_{u,i,F}\}, 1 \le i \le k$, are in $M$, where $k$ is the number of occurrences of $u$ in the clauses in $C$. Assume this is not the case. Then there exists a value $p, 1 \le p \le k$, such that $\{x_{u,p}, y_{u,p,T}\}$, $\{x_{u,p}, y_{u,q,F}\} \in M$

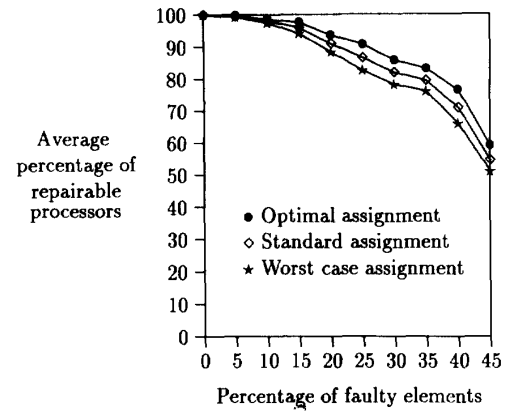


Fig. 10. Optimal, standard, and worst case assignments in an augmented ring.

where $q = p + 1$ if $p < k$ and $q = 1$ if $p = k$. However, in this case vertex $z_{u,p}$ is not repairable, a contradiction. With this fact established, the remainder of the proof is identical to the proof of Theorem 1. □

### B. Experimental Results

The optimal matching algorithm for $D(Y) \le 2$ was compared to standard matchings and worst case matchings on a simple ring of primary processors such that each pair of neighboring primary processors share a unique spare processor. In our experiment, rings with 20 primary processors, and thus 20 spare processors, were generated with faulty elements introduced at random according to a uniform distribution. For each fault frequency, 100 rings were generated with faulty processors introduced at random according to a uniform distribution. The fault frequency ranged from 0% to 45%. (At fault frequency 50% the rings could not be repaired.) The experimental results are summarized in Fig. 10.

## VII. An Approximation Algorithm

Although the optimal matching problem is NP-complete for arbitrary $D(X)$ and $D(Y)$, we now describe an approximation algorithm that can be applied to any architecture and is guaranteed to be at least $\frac{1}{D(X)}$-optimal. In other words, if the number of repairable vertices in an optimal solution is $k$ then the approximation algorithm finds a solution in which the number of repairable vertices is at least $\frac{1}{D(X)} \cdot k$. Experimental results given in Section VII-B suggest that in practice this algorithm performs substantially better than indicated by this theoretical lower bound.

### A. The Algorithm

The approximation algorithm transforms a given instance of the optimal matching problem into a weighted matching problem as follows. From the given bipartite graph $G = (X \cup Y, E)$ and set $F \subseteq X$, we construct a weighted bipartite graph $G' = (F \cup Y, E')$ where $E' \subseteq E$ is the set of

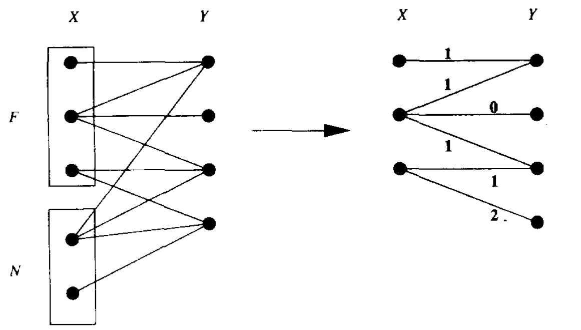


Fig. 11. An example of the construction used in the $\frac{1}{D(x)}$-optimal approximation algorithm.

edges in $E$ that are incident to vertices in $F$. For each edge $e = \{x, y\} \in E$ with $x \in F$, let $w_e$ denote the number of vertices in $N$ that are adjacent to $y$. Edge $e = \{x, y\} \in E'$ is assigned weight $w_e$. An example of this construction is illustrated in Fig. 11.

*Theorem 7:* Let M be a minimum weight complete matching in $G'$ and let $k$ be the number of repairable vertices in an optimal matching for $G$. If matching $M$ is used in $G$, at least $\frac{1}{D(X)} \cdot k$ vertices are repairable.

*Proof:* Let $T$ denote the total number of edges in $G$ incident to vertices in $N$. Let $W$ denote the total weight of matching $M$ in $G'$. Now consider the matching $M$ in $G$. The total number of edges between vertices in $N$ and unmatched vertices with respect to $M$ in $Y$ is $T - W$. Therefore, at least $(T - W)/D(X)$ vertices in $N$ must be adjacent to unmatched vertices with respect to $M$. Let $M_{\text{opt}}$ be an optimal matching in $G$. Since $M_{\text{opt}}$ is a maximum matching in $G'$, the total weight of $M_{\text{opt}}$ is at least $W$. Therefore, in graph $G$, the number of edges between vertices in $N$ and unmatched vertices with respect to $M_{\text{opt}}$ in $Y$ is at most $T - W$. Thus, $k \leq T - W$ and at least $\frac{1}{D(X)} \cdot k$ vertices are repairable with respect to $M$. □

The complexity of the approximation algorithm is that of finding a minimum cost maximum matching. This can be done using one of a variety of polynomial time algorithms [20].

### *B. Experimental Results*

In the first experiment, processor arrays with 32 primary processors and 16 spare processors and $D(X) = 4$ were generated with connections between primary processors and spare processors selected randomly. In each randomly generated array, faulty elements were introduced at random with fault frequency ranging from 0 to 30% in increments of 5% and for each fault frequency value, 100 random arrays were generated. For each array, the approximation algorithm was used to find an assignment. In addition, the optimal assignment was found for each array using an exhaustive search algorithm and an assignment was found using standard matching. The approximation algorithm found solutions that were at least 0.94 times the optimal size, whereas the standard matching algorithm found solutions as low as 0.75 times optimal. Optimal solutions were found using exhaustive search.

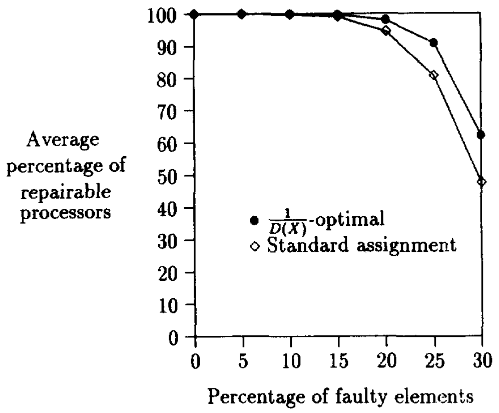


Fig. 12. $\frac{1}{D(x)}$-optimal algorithm versus arbitrary assignment for randomly generated arrays with 100 primary processors, 50 spare processors, and $D(x) = 4$.

Fig. 12 gives results for randomly generated arrays with 100 primary processors and 50 spare processors with $D(X) = 4$. The performance of the approximation algorithm is compared only to that of the standard matching algorithm, since these arrays were too large to be solved optimally by exhaustive search. These results indicate that the approximation algorithm can find matchings that, in many cases, contain 6 to 14% more repairable processors than are obtained by standard matching.

## VIII. Conclusion

In this paper we have presented efficient reconfiguration algorithms for fault tolerant processor arrays operating in real-time environments. Such systems typically alternate between a strict mode and a relaxed mode. Although reconfiguration must be performed in a purely local fashion during the strict mode, global reconfiguration may be performed during the relaxed mode to restore the system to a more reliable state. To this end, the notations of safe and optimal assignments were defined.

We have shown that the problem of finding safe and optimal assignments is, in general, NP-complete. However, we have shown that several broad classes of architectures have properties that allow us to find safe and optimal assignments in polynomial time. First, a linear time algorithm was given for finding optimal assignments in topologies that contain no cycles alternating between primary processors and spare processors. It was observed that several architectures studied in the literature have this property. Next, we considered topologies in which the primary and spare processors have constant degree. A polynomial time safe assignment algorithm was given for the case that each primary processor is adjacent to at most two spare processors. Similarly, a linear time optimal assignment algorithms was given for the case that each spare processor is adjacent to at most two primary processors. Again, several well-known fault tolerant architectures have these properties. Additionally, it was shown that these results are tight in the sense that the problems become NP-complete when they are further generalized. Finally, an approximation algorithm was described that can be applied to any architecture.

## REFERENCES

[1] P. Banerjee, "Strategies for reconfiguring hypercubes under faults," in *Proc. 20th Int. Symp. Fault-Tolerant Comput.*, June 1990, pp. 210–215.

[2] M. Chean and J. A. B. Fortes, "A taxonomy of reconfiguration techniques for fault-tolerant processor arrays," *IEEE Comput.*, vol. 23, pp. 55–69, Jan. 1990.

[3] C. Chen, A. Feng, T. Kikuno, and K. Torii, "Reconfiguration algorithm for fault-tolerant arrays with minimum number of dangerous processors," in *Proc. 21st Int. Symp. Fault-Tolerant Comput.*, 1991, pp. 452–459.

[4] M. Davis and H. Putnam, "A computing procedure for quantification theory," *J. ACM*, vol. 7, pp. 201–215, 1960.

[5] S. Even, A. Itai, and A. Shamir, "On the complexity of timetable and multicommodity flow problems," *SIAM J. Comput.*, vol. 5, pp. 691–703, 1976.

[6] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness.* New York: Freeman, 1979.

[7] A. S. Hassan and V. K. Agarwal, "A fault-tolerant modular approach architecture for binary trees," *IEEE Trans. Comput.*, vol. C-35, pp. 356–361, Apr. 1986.

[8] B. Kim and D. Towsley, "Dynamic flow control protocols for packet-switching multiplexers serving real-time multipacket messages," *IEEE Trans. Commun.*, vol. COM-34, Apr. 1986.

[9] R. Libeskind-Hadas, "Reconfiguration of fault-tolerant VLSI systems," Dep. of Comput. Sci., Univ. of Illinois at Urbana-Champaign, Tech. Rep. UIUCDCS-R-93-1824, July 1993.

[10] F. Lombardi, R. Negrini, M. G. Sami and R. Stefanelli, "Reconfiguration of VLSI arrays: A covering approach," in *Proc. 17th Int. Symp. Fault-Tolerant Comput.*, 1987, pp. 251–256.

[11] R. G. Melhem, "Bi-level reconfigurations of fault tolerant arrays," *IEEE Trans. Comput.*, vol. 41, pp. 231–239, Feb. 1992.

[12] K. Melhorn, *Data Structures and Algorithms 2: Graph Algorithms and NP-Completeness.* New York: Springer-Verlag, 1984.

[13] R. Negrini, M. G. Sami, and R. Stefanelli, *Fault Tolerance Through Reconfiguration in VLSI and WSI Arrays.* Cambridge, MA: The M.I.T. Press, 1989.

[14] C. S. Raghavendra, A. Avizienis, and M. Ercegovac, "Fault tolerance in binary tree architectures," *IEEE Trans. Comput.*, vol. C-33, pp. 568–572, June 1984.

[15] D. A. Rennels, "On implementing fault-tolerance in binary hypercubes," *Proc. 16th Int. Symp. Fault-Tolerant Comput.*, 1986, pp. 344–349.

[16] C. Savage, "Maximum matchings and trees," *Inform. Processing Lett.*, vol. 10, nos. 4, 5, pp. 202–205, July 5, 1980.

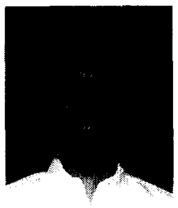
[17] N. Shrivastava and R. G. Melhem, "Efficient and optimal fault-to-spare assignments in doubly fault tolerant arrays," in *Proc. IEEE Int. Workshop on Defect and Fault Tolerance in VLSI Syst.*, Nov. 1991, pp. 247–259.

[18] A. D. Singh, "A reconfigurable modular fault tolerant binary tree architecture," in *Proc. 17th Int. Symp. Fault-Tolerant Comput.*, 1987, pp. 298–304.

[19] ——, "Interstitial redundancy: An area efficient fault tolerance scheme for large area VLSI processor arrays," *IEEE Trans. Comput.*, vol. 37, pp. 1398–1410, Nov. 1988.

[20] R. E. Tarjan, *Data Structures and Network Algorithms, Soc. Industr. and Appl. Math.*, Philadelphia, PA, 1983.

[21] M. W. Yung, M. J. Little, R. D. Etchells, and J. G. Nash, "Redundancy for yield enhancement in the 3-D computer," in *Proc. 1989 IEEE Int. Conf. on Wafer Scale Integr.*, 1989, pp. 73–82.

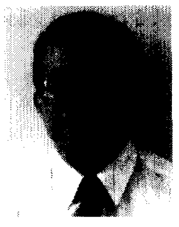


**Ran Libeskind-Hadas** (S'91–M'93) received the A.B. degree in applied mathematics from Harvard University in 1987, and the M.S. and Ph.D. degrees in computer science from the University of Illinois at Urbana-Champaign in 1989 and 1993, respectively.

He is currently an Assistant Professor of Computer Science at Harvey Mudd College. His areas of research interest are fault-tolerant computing, parallel computing, and design and analysis of algorithms.

**Nimish Shrivastava** received the M.S. degree in computer science in 1988 from the University of Pittsburgh, where he is currently working towards the Ph.D. degree.

His research interests are in the area of fault tolerance, in general, and reconfiguration algorithms, in particular.



**Rami G. Melhem** received the B.E. degee in electrical engineering from Cairo University, Egypt, in 1976, the M.A. degree in mathematics, and the M.S. degree in computer science from the University of Pittsburgh in 1981, and the Ph.D. degree in computer science from the University of Pittsburgh in December 1983.

Since 1989, he has been an Associate Professor of Computer Science at the University of Pittsburgh. Previously, he was an Assistant Professor at Purdue University and at the University of Pittsburgh. He has published numerous papers in the areas of systolic architectures, parallel computing, fault-tolerant processor arrays, and optical computing.

Dr. Melhem is a member of the IEEE Computer Society, the Association for Computing Machinery, and the International Society for Optical Engineering. He served on program committees of several conferences and workshops and he is on the editorial board of the IEEE TRANSACTIONS ON COMPUTERS. He was Guest Editor of the Special Issue of the *Journal of Parallel and Distributed Computing on Optical Computing and Interconnection Systems.*

**C. L. Liu** (M'64–SM'82–F'86) received the B.S. degree from Cheng Kung University in Taiwan in 1956. He received the M.S. and the E.E. degrees in 1960, and the Sc.D. degree in 1962, all in electrical engineering from the Massachusetts Institute of Technology.

He is currently a Professor of Computer Science at the University of Illinois at Urbana-Champaign. His research interests are in design and analysis of algorithms, computer-aided design of integrated circuits, and combinatorial mathematics.