On the Embedding of a Class of Regular Graphs in a Faulty Hypercube*

Yu-Chee Tseng and Ten-Hwang Lai

Department of Computer and Information Science
The Ohio State University
Columbus, Ohio 43210
Tel: (614)292-5813, Fax: (614)292-2911
E-mail: {tseng, lai}@cis.ohio-state.edu

Abstract

A wide range of graphs with regular structures are shown to be embeddable in an injured hypercube with faulty links. These include rings, linear paths, binomial trees, binary trees, meshes, tori, and many others. Unlike many existing algorithms which are capable of embedding only one type of graphs, our algorithm embeds the above graphs in a unified way, all centered around a notion called *edge matrix*. In many cases, the degree of fault tolerance offered by the algorithm is optimal or near-optimal.

Key Words: graph embedding, processor allocation, hypercube, binary-reflected trees, fault tolerance.

1 Introduction

Embedding a *guest graph* into a *host graph*, or the *graph embedding problem*, has long been recognized as being suitable for modeling the problem of processor allocation in a parallel or distributed system [1, 2]. Because of the importance and popularity of the hypercube as a network architecture for concurrent computers, the problem of embedding in a hypercube has received much attention from researchers and has been intensively studied for various guest graphs, such as rings [12], trees [20, 21], pyramid [9], and shuffle networks [16]. In general, this problem is computationally difficult. Determining whether an arbitrary graph is embeddable in a hypercube is NP-hard [5], and it remains so even if dilation is allowed [8] or if the guest graph is a tree [19].

In a hypercube of high dimension, the probability of there existing a node/link fault may not be negligible. Should this happen, it is desirable that the faulty components be

*This research was supported by National Science Foundation under Grant CCR-9010589.

isolated from the rest of the hypercube so that guest graphs still can be embedded in the cube. This leads to the problem of embedding a graph is an injured hypercube.

Several results on this topic are available in the literature of parallel computing. Algorithms for constructing a ring as large as possible in an injured n-cube have been proposed in [3], [17], and [14]. These algorithms are able to tolerate up to $n/2$, $2n$, and $\Theta(2^{n/2})$ faulty nodes, respectively. A level-$(n-1)$ complete binary tree can be embedded in an n-cube with $O(n)$ faulty nodes [4]; a better result is available that allows up to $\Theta(n^2)$ faulty nodes [18]. Fault-tolerant embedding of meshes in injured hypercubes was studied in [22]. All these results are concerned with graph embedding in a hypercube with faulty *nodes*, where whenever a node is faulty, it is commonly assumed that not only the node itself but all the links incident upon it are not available.

In this paper, we consider the case where a hypercube has faulty *links* rather than faulty nodes. It has been suggested (e.g., in [4, 17, 18]) that faulty links be treated as faulty nodes: if link $\langle u, v\rangle$ is faulty then regard either node u or node v as faulty. With this treatment, faulty links are converted to faulty nodes and the above mentioned results are immediately applicable. While convenient, this approach may be inefficient in some cases. For instance, if in some application a user wishes to use as many processors as possible, it is certainly undesirable to give up a processor (node) just because a link connected to it is faulty.

Assuming faulty links do not jeopardize their end nodes, it is shown in [10, 13] that a ring of length 2^n can be embedded in an n-cube with up to $n-2$ faulty links. (Such an embedding would be impossible if faulty links are treated as faulty nodes.) In this paper, we considerably extend this result and show that a wide range of graphs can be embedded in an injured n-cube with faulty links: rings, linear paths, binary trees, binomial trees, meshes, tori, products of these graphs, and many others. In particular, we have the following results (among others):

1. Any level-n *binary-reflected tree* can be embedded in an n-cube with up to $n-1$ faulty links. (Binary-reflected trees will be defined in Section 2. Linear paths and binomial trees are two of the most important binary-reflected trees.)

2. A level-n complete binary tree (with 2^n-1 nodes) can be embedded, with congestion 2 and dilation 2, in an n-cube with up to $n-1$ faulty links.

3. Any k-dimensional, $2^d \times \cdots \times 2^d$, mesh or torus can be embedded in an n-cube with up to r faulty links, where $kd = n$ and $r = \min\{\Theta(2^d), n - \Theta(k \log d)\}$. The same is true even if the guest graph is not a mesh or torus but the product of any k of the

above mentioned graphs.

The purpose of this paper is twofold: 1) to report the above new results, and 2) to demonstrate the advantage of representing a subgraph of a hypercube as an edge matrix. In a previous work [15], we introduced the notion of *edge matrix* as a possible data structure for representing a subgraph of a hypercube (or the embedding of a guest graph in a hypercube). Any of the above mentioned graphs can be conveniently represented as an edge matrix, and so can the set of faulty links. As such, our embedding problem becomes one of transforming an edge matrix from one form to another such that the reformed matrix is *disjoint* (defined in Section 2) from the one representing the faulty links and represents the same guest graph. All our results are obtained in this way. Thus, unlike many existing embedding algorithms that were designed for only one type of graphs, our algorithms are applicable to a broad range of graphs.

We will define edge matrices and binary-reflected trees in the next section; show how to embed rings, binary-reflected trees, and complete binary trees in Section 3; and extend these results to the embedding of a product graph in Section 4.

2 Preliminaries

2.1 Basic Definitions

An n-cube is an undirected graph consisting of 2^n nodes each labeled by a distinct binary number $b_1 \dots b_n$. Two nodes $b_1 \dots b_{i-1} b_i b_{i+1} \dots b_n$ and $b_1 \dots b_{i-1} \overline{b}_i b_{i+1} \dots b_n$ are joined by an edge *along* dimension i, where $\overline{b}_i$ is the 1's complement of b_i. A d-dimensional subcube, or d-subcube, is denoted by a ternary string $x_1 \dots x_n$, where $x_i \in \{0, 1, *\}$, with exactly d bits of $*$'s (called "don't care" bits) in the string.

For any two d-subcubes, $Q = x_1 \dots x_{i-1} x_i x_{i+1} \dots x_n$ and $Q' = x_1 \dots x_{i-1} \overline{x}_i x_{i+1} \dots x_n$, whose bits in the ith position complement each other, there are exactly 2^d edges between Q and Q'. We call these edges, as a whole, the *edge set* between Q and Q' and denote it by the string $x_1 \dots x_{i-1} \xi x_{i+1} \dots x_n$. The position of ξ in the string indicates the dimension in the hypercube along which these edges are. For instance, the edge set between subcubes $1*0*$ and $1*1*$, denoted by $1*\xi*$, contains four (2^2) edges all along dimension 3.

We do not distinguish between a string $x_1 x_2 \dots x_n$ and a vector $(x_1, x_2, \dots, x_n)$, and will use the two terms alternatively.

(b)

Figure 1: A 4×4 edge matrix defining a spanning tree in the 4-cube.

2.2 Edge Matrices

If each edge set is represented by a length-n vector, then a collection of m edge sets can be represented by an $m \times n$ matrix, each row representing an edge set. Such a matrix is called an *edge matrix*. For instance, the matrix in Fig. 1(a) is an edge matrix. For any edge matrix X, let $E(X)$ denote the set of edges that equals the union of the edge sets represented by the rows of X; and define $G(X) = (\mathcal{V}, \mathcal{E})$ to be the graph with vertex set $\mathcal{V}$ equaling the set of all nodes of the n-cube and with edge set $\mathcal{E} = E(X)$. $G(X)$ is a subgraph of the n-cube or can be regarded as a guest graph after embedded into the cube. The subgraph defined by the example edge matrix of Fig. 1(a) is depicted in Fig. 1(b).

If X is a matrix, we denote by $X[i..j][k..l]$ the *submatrix* of X that locates at the intersection of rows $i..j$ and columns $k..l$. The notation $[-]$ is the abbreviation of $[1..n]$, and $[i]$ is that of $[i..i]$.

Definition 1 An edge matrix X is said to be *well-formed* if all ξ's in the same column are in adjacent rows. Such an X can be written as

$$X = \begin{pmatrix} X_1 \\ X_2 \\ \vdots \\ X_r \end{pmatrix}$$

or, to save space, as $(X_1, X_2, \ldots, X_r)^T$, where each X_i is a submatrix of X containing all

rows in X whose ξ's are in a same column. If, for all $i = 1..r$, the edges in $E(X_i)$ are all along dimension $n - r + i$ (i.e., the ξ's in X_i are all in column $n - r + 1$), then X is called *dimension-sorted.*

Definition 2 If $a \in \{*, \xi, 0, 1\}$ and $b \in \{0, 1\}$ are two bits, define $a \oplus b$ to be the "exclusive-or" of a and b with the extension that $* \oplus b = *$ and $\xi \oplus b = \xi$. Given an edge set $x = (x_1, \ldots, x_n)$ and a binary vector $C = (c_1, \ldots, c_n)$, define $x \oplus C = (x_1 \oplus c_1, \ldots, x_n \oplus c_n)$. If X is an $m \times n$ edge matrix, define $X \oplus C$ to be the $m \times n$ edge matrix whose ith row is $X[i][-] \oplus C$.

Definition 3 Two edge matrices X and Y are said to be *disjoint*, denoted as $X \leftrightarrow Y$, if $E(X) \cap E(Y) = \emptyset$. Two edge matrices X and Y are said to be *isomorphic*, denoted as $X \approx Y$, if $X = \pi(Y) \oplus C$ for some π and C, where π is any permutation on the columns of Y and C is any length-n binary vector.

The notion of isomorphism between edge matrices corresponds to the regular notion of isomorphism in graph theory. Indeed, we have the following lemma that directly follows from definitions.

Lemma 1 *If two edge matrices X and Y are isomorphic, then the graphs they define, $G(X)$ and $G(Y)$, are isomorphic.*

The following lemma is also straightforward and can be easily verified.

Lemma 2 *For two edge matrices X and Y, if $X \leftrightarrow Y$, then $\pi(X) \oplus C \leftrightarrow \pi(Y) \oplus C$, for any column-permutation π and any binary vector C.*

2.3 Binary-Reflected Trees

The next lemma, proved in [15], describes the edge matrices of a wide range of trees in the n-cube.

Lemma 3 [15] *Let*

$$X_{br} = \begin{pmatrix} \xi & * & * & \ldots & * & * \\ b_{2,1} & \xi & * & \ldots & * & * \\ b_{3,1} & b_{3,2} & \xi & \ldots & * & * \\ & & & \ldots & & \\ b_{n-1,1} & b_{n-1,2} & b_{n-1,3} & \ldots & \xi & * \\ b_{n,1} & b_{n,2} & b_{n,3} & \ldots & b_{n,n-1} & \xi \end{pmatrix} \tag{1}$$

where each $b_{i,j}$ is 0 or 1. Then

(a) $G(X_{br})$ *defines a spanning tree in the n-cube;*

(b) if for each $i = 1..n-2$, $b_{i+1,i} = b_{i+2,i} = b_{i+3,i} = \cdots = b_{n,i}$, *then* $G(X_{br})$ *is a* binomial tree*; and*

(c) if for each $i = 1..n-2$, $b_{i+1,i} = \overline{b}_{i+2,i} = \overline{b}_{i+3,i} = \cdots = \overline{b}_{n,i}$, *then* $G(X_{br})$ *is a* Hamiltonian path.

Definition 4 A graph is said to be a *level-n binary-reflected tree* or *BR-tree* if it is isomorphic to $G(X)$ for some X conforming to Eq. (1).

The term "binary-reflected" somehow reflects the structure of a BR-tree: a level-0 BR-tree consists of a single node, while a level-n BR-tree can be constructed recursively by taking two identical (isomorphic) level-$(n-1)$ BR-trees and joining them with a single edge at any two *corresponding* nodes (so that the endpoints of the new edge correspond to each other under the isomorphism between the two trees). (One may see such a recursive structure in the tree of Figure 1(b).) We saw some similarity between the above construction and the binary-reflected Gray code and therefore borrowed the term binary-reflected.

There are a large number of BR-trees of different shapes, as joining the two level-$(n-1)$ BR-trees at different places may yield different level-n BR-trees. Two of the most important level-n BR-trees are the length-2^n linear path and the level-n binomial trees[†]. Binomial trees are useful for implementing broadcasting [7] and parallel divide-and-conquer [11] in a hypercube.

Note that each level-n BR-tree is an abstract tree isomorphic to a spanning tree in the n-cube. Thus, a ring is not a BR-tree, and a Hamiltonian cycle of an n-cube cannot be described as an edge matrix conforming to Eq. (1). Nevertheless, there is a way to depict a Hamiltonian cycle with an edge matrix.

Lemma 4 *Let*

$$X_{hc} = \begin{pmatrix} \xi & * & * & \dots & * & * & * \\ b_1 & \xi & * & \dots & * & * & * \\ \overline{b}_1 & b_2 & \xi & \dots & * & * & * \\ \overline{b}_1 & \overline{b}_2 & b_3 & \dots & * & * & * \\ & & & \dots & & & \\ \overline{b}_1 & \overline{b}_2 & \overline{b}_3 & \dots & \xi & * & * \\ \overline{b}_1 & \overline{b}_2 & \overline{b}_3 & \dots & b_{n-2} & \xi & * \\ \overline{b}_1 & \overline{b}_2 & \overline{b}_3 & \dots & \overline{b}_{n-2} & * & \xi \end{pmatrix} \tag{2}$$

[†]Denote by B_i the *level-i binomial tree*. B_0 consists of a single node, which is also regarded as its root. $B_i, i \geq 1$, is constructed out of two B_{i-1}'s with their roots joined by an edge and one of the two roots of B_{i-1}'s becoming the root of B_i. Such a construction conforms to the one for BR-trees.

where each b_i is 0 or 1, $i = 1..n-2$. Then $G(X_{hc})$ defines a Hamiltonian cycle in the n-cube.

Proof. Let X be the matrix obtained from X_{hc} by substituting any binary bit b for the $*$ bit in the bottom row. $E(X_{hc}) = E(X) \cup \{e\}$, where e is the edge $\overline{b}_1\overline{b}_2\overline{b}_3 \ldots \overline{b}_{n-2}\overline{b}\xi$. By Lemma 3, $G(X)$ is a Hamiltonian path in the n-cube. As path $G(X)$'s endpoints are at the nodes incident by e (i.e., node $\overline{b}_1\overline{b}_2\overline{b}_3 \ldots \overline{b}_{n-2}\overline{b}0$ and node $\overline{b}_1\overline{b}_2\overline{b}_3 \ldots \overline{b}_{n-2}\overline{b}1$), adding e to $G(X)$ results in a Hamiltonian cycle. □

We conclude this section with a lemma that will be used frequently in the rest of the paper. Note that throughout this paper, the logarithmic base is 2.

Lemma 5 *Let $s_1 + s_2 + \cdots + s_k = s$, where $s_i, i = 1..k$, are positive integers. Then $\sum_{i=1}^{k} \lceil \log(1 + s_i) \rceil \leq s$.*

3 Embedding BR-trees, Complete Binary Trees, and Rings

In the previous section, we have shown how to use an edge matrix to represent a set of edges and, in particular, the embedding of a wide range of guest graphs. In this section, we show how such representations may facilitate fault-tolerant embedding of these graphs in an n-cube with faulty links.

Consider the problem of embedding a level-n BR-tree in an n-cube in which there are exactly f faulty links and no faulty nodes. Let the faulty links be represented as an $f \times n$ edge matrix F, each fault corresponding to a row. Let X be an edge matrix (conforming to Eq. (1)) that represents the BR-tree in question. If we can find an edge matrix X' such that $X' \approx X$ and $X' \leftrightarrow F$, then because $G(X')$ is isomorphic to $G(X)$ (by Lemma 1) and $E(X')$ is disjoint from $E(F)$ (by Definition 3), $G(X')$ is a fault-free embedding of the BR-tree. Our embedding problem thereby reduces to one of finding such an X', given X and F.

We first establish a sufficient condition for such an X' to exist.

Theorem 1 *Let X be an edge matrix conforming to Eq. (1), and let $F = (F_1, \ldots, F_r)^T$ be a well-formed edge matrix. If $\sum_{i=1}^{r} \lceil \log(1 + |E(F_i)|) \rceil \leq n - 1$, then there exists an edge matrix X' such that $X' \approx X$ and $X' \leftrightarrow F$.*

Proof. We construct a column permutation function π and a binary vector C such that $X' = \pi^{-1}(X \oplus C)$ has the desired properties. The overall construction and the relationship among F, X and the yet-to-be-constructed π, C, and X' are depicted in Fig. 2.

Figure 2: Constructing $X' = \pi^{-1}(X \oplus C)$ such that $X' \approx X$ and $X' \leftrightarrow F$.

Step 1: Let π be a column permutation of F (i.e., the columns of F are permuted according to the permutation function π) such that $\pi(F)$ is dimension-sorted (it is not hard to find such a π):

$$\pi(F) = \begin{pmatrix} \pi(F_1) \\ \pi(F_2) \\ \vdots \\ \pi(F_r) \end{pmatrix} = \begin{pmatrix} F'_1 \\ F'_2 \\ \vdots \\ F'_r \end{pmatrix}.$$

Step 2: In this step, we construct a length-n binary vector C such that $\pi(F) \leftrightarrow X \oplus C$. Specifically, we compute a binary vector C_i for each $F'_i, i = 1..r$, and then concatenate $C_1, C_2, \ldots, C_r$ (with minor modification) to yield C. The purpose of C_i is to guarantee the relationship $F'_i \leftrightarrow X[n-r+i][-] \oplus C$. Since F'_i is the only submatrix of $\pi(F)$ that contains edges along dimension $n-r+i$, we will have $\pi(F) \leftrightarrow X \oplus C$.

The following procedure constructs C:

Procedure *Binary_Vector*()

a) **for** $i := 1$ **to** r **do**

   i) Let $u_i := 1 + \sum_{j=1}^{i-1} \lceil \log(1 + |E(F'_j)|) \rceil$ and $v_i := \sum_{j=1}^{i} \lceil \log(1 + |E(F'_j)|) \rceil$;

   ii) Let $Z_i = (z_{u_i}, \ldots, z_{v_i})$ be a length-$(v_i - u_i + 1)$ binary vector that is distinct from each row of $F'_i[-][u_i..v_i]$;

   iii) $C_i := X[n-r+i][u_i..v_i] \oplus Z_i$;

b) Let $C := (C_1, C_2, \ldots, C_r, C^*)$, where C^* is any sequence of 0/1's that makes C to be of length n.

In the ith iteration of step a), Z_i and C_i are computed. Since Z_i is a binary vector of

length $v_i - u_i + 1$, there are

$$2^{v_i - u_i + 1} = 2^{\lceil \log(1 + |E(F_i')|) \rceil} \geq 2^{\log(1 + |E(F_i')|)} \geq 1 + |E(F_i')|$$

possible choices for Z_i. On the other hand, F_i' contains $|E(F_i')|$ rows, each having a ξ in position $n - r + i$. If we assume that $v_i < n - r + i$ (proved later), then matrix $F_i'[-][u_i..v_i]$ contains only binary bits. As there are fewer rows in $F_i'[-][u_i..v_i]$ than the possible settings of Z_i, a Z_i as specified in the above procedure must exist.

With the same assumption $v_i < n - r + i$, one readily sees from Eq. (1) that vector $X[n - r + i][u_i..v_i]$ contains only binary bits. Hence, step a.iii) implies

$$X[n - r + i][u_i..v_i] \oplus C_i = Z_i,$$

which is distinct from each row of $F_i'[-][u_i..v_i]$. This in turn implies that

$$X[n - r + i][-] \oplus (C_1, \ldots, C_i, \ldots, C_r, C^*) \leftrightarrow F_i'[-][-] = F_i',$$

independent of how the other $C_j, j \neq i$, and C^* are set. Thus, $\pi(F) \leftrightarrow X \oplus C$.

Step 3: Let $X' = \pi^{-1}(X \oplus C)$, where π^{-1} is the reverse permutation of π. By Lemma 1, $X' \approx X$. Because of Lemma 2, it follows from $\pi(F) \leftrightarrow X \oplus C$ that $F \leftrightarrow \pi^{-1}(X \oplus C)$. Thus, X' has the desired properties.

It remains to prove the above assumed inequality: $v_i < n - r + i$. A simple induction on i, from r down to 1, will do the job. When $i = r$, by definition, $v_r = \sum_{i=1}^{r} \lceil \log(1 + |E(F_i)|) \rceil \leq n - 1 < n$. To prove the induction step, simply observe that as i reduces by 1, the right-hand side of the inequality also decreases by 1, while the left-hand side decreases by $\lceil \log(1 + |E(F_i)|) \rceil \geq 1$. □

Example 1: Figure 3 demonstrates the procedure described in the above proof. Given are an X and an F with 6 faults in a 6-cube. Within F, submatrices F_1, F_2, and F_3 contain rows 1∼2, row 3, and rows 4∼6, respectively. Step 1 sorts F into $\pi(F)$, where the notation "$i \rightarrow j$" means "permute column i to column j". From $\pi(F)$, step 2 computes the values of u_i, v_i, and $Z_i, i = 1..3$. Then, using Z_i's and the elements highlighted in X, C_i's are computed. Finally, X' is constructed. □

The following corollary shows the degree of fault tolerance guaranteed by the above theorem in embedding a level-n BR-tree.

Corollary 1 *Any level-n BR-tree can be embedded in an injured n-cube that contains no more than* $n - 1$ *faulty links.*

$$\text{step 1: } F = \begin{pmatrix} 1 & \xi & 0 & 0 & 1 & 1 \\ 1 & \xi & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & \xi \\ 1 & 0 & \xi & 0 & 1 & 1 \\ 0 & 0 & \xi & 1 & 0 & 0 \\ 0 & 1 & \xi & 0 & 0 & 1 \end{pmatrix} \Longrightarrow \pi(F) = \begin{pmatrix} \boxed{1} & \boxed{0} & 1 & \xi & 1 & 0 \\ \boxed{1} & \boxed{0} & 0 & \xi & 1 & 0 \\ 0 & 0 & \boxed{0} & 1 & \xi & 1 \\ 1 & 0 & 1 & \boxed{0} & \boxed{1} & \xi \\ 0 & 1 & 0 & \boxed{0} & \boxed{0} & \xi \\ 0 & 0 & 0 & \boxed{1} & \boxed{1} & \xi \end{pmatrix}, \text{ where } \pi = \begin{cases} 1 \to 1 \\ 2 \to 4 \\ 3 \to 6 \\ 4 \to 2 \\ 5 \to 3 \\ 6 \to 5 \end{cases}$$

$$\text{step 2: } \pi(F) \Longrightarrow \begin{array}{l} u_1 = 1, v_1 = 2, Z_1 = (0,1) \\ u_2 = 3, v_2 = 3, Z_2 = (1) \\ u_3 = 4, v_3 = 5, Z_3 = (1,0) \end{array}$$

$$X = \begin{pmatrix} \xi & * & * & * & * & * \\ 1 & \xi & * & * & * & * \\ 0 & 1 & \xi & * & * & * \\ \boxed{1} & \boxed{0} & 0 & \xi & * & * \\ 0 & 0 & \boxed{1} & 1 & \xi & * \\ 0 & 1 & 1 & \boxed{0} & \boxed{0} & \xi \end{pmatrix} \Longrightarrow \begin{array}{l} C_1 = (1,0) \oplus Z_1 = (1,1) \\ C_2 = (1) \oplus Z_2 = (0) \\ C_3 = (0,0) \oplus Z_3 = (1,0) \\ C^* = (0) \end{array} \Longrightarrow C = (1,1,0,1,0,0)$$

$$\text{step 3: } X' = \pi^{-1}(X \oplus C) = \pi^{-1}\left(\begin{pmatrix} \xi & * & * & * & * & * \\ 0 & \xi & * & * & * & * \\ 1 & 0 & \xi & * & * & * \\ 0 & 1 & 0 & \xi & * & * \\ 1 & 1 & 1 & 0 & \xi & * \\ 1 & 0 & 1 & 1 & 0 & \xi \end{pmatrix} \right) = \begin{pmatrix} \xi & * & * & * & * & * \\ 0 & * & * & \xi & * & * \\ 1 & * & * & 0 & \xi & * \\ 0 & \xi & * & 1 & 0 & * \\ 1 & 0 & * & 1 & 1 & \xi \\ 1 & 1 & \xi & 0 & 1 & 0 \end{pmatrix}$$

Figure 3: An example for the proof of Theorem 1.

Proof. Without loss of generality, assume F to be well-formed, $F = (F_1, \ldots, F_r)^T$. By Lemma 5, $\sum_{i=1}^{r} \lceil \log(1 + |E(F_i)|) \rceil \leq |E(F)|$. Hence, Theorem 1 is applicable if $|E(F)| \leq n - 1$. □

In the above proof, observe that $\sum_{i=1}^{r} \lceil \log(1 + |E(F_i)|) \rceil = |E(F)|$ only if $|E(F_i)| = 1$ or 2 for each i. Thus, if faulty links occur in only a very small number of dimensions, the embedding algorithm has potential to tolerate more than $n - 1$ faults.

As was commented earlier, complete binary trees and rings are not BR-trees. Nevertheless, Theorem 1 can be applied to the embedding of such graphs, as shown in the following two corollaries.

Corollary 2 *If there are no more than $n - 1$ faulty links in the n-cube, a level-n complete binary tree (with $2^n - 1$ nodes) can be embedded in the cube with dilation 2 and congestion 2.*

Proof. A level-n complete binary tree can be embedded into a level-n binomial tree with congestion 2 and dilation 2 (see [15]). Since binomial trees are BR-trees (by Lemma 3) and there exists an $(n-1)$-resilient embedding of a level-n binomial tree in an n-cube (by Corollary 1), this corollary is proved. □

The following result has been established in [10] in a relatively complicated way. Here we offer a much simpler proof.

Corollary 3 *If there are no more than $n - 2$ faulty links in the n-cube, then there exists a Hamiltonian cycle in the cube.*

Proof. By Lemma 4, $G(X_{hc})$ is a Hamiltonian cycle. Rewrite X_{hc} as

$$X_{hc} = \begin{pmatrix} X_{U1} & X_{U2} \\ X_{D1} & X_{D2} \end{pmatrix}$$

such that $X_{U1} = X_{hc}[1..n-1][1..n-1]$ and $X_{D2} = X_{hc}[n][n] = (\xi)$.

As there are at most $n - 2$ faults in F, one can easily find a column-permutation function π_1 on F such that in $\pi_1(F) = F'$ *no* edge is along dimension n. Consider both $F'[-][1..n-1]$ and X_{U1} as edge matrices in an $(n-1)$-cube. Applying Theorem 1 to these two edge matrices, we can find a permutation π_2 and a length-$(n-1)$ binary vector C such

that $F'[-][1..n-1] \leftrightarrow \pi_2^{-1}(X_{U1} \oplus C)$. From this we obtain the following:

$$
\begin{array}{ll}
 & F'[-][1..n-1] \leftrightarrow \pi_2^{-1}(X_{U1} \oplus C) \\
\Longrightarrow & /* \text{ add } F'[-][n] \text{ and } X_{U2} \text{ into the above equation } */ \\
 & F' = (F'[-][1..n-1], F'[-][n]) \leftrightarrow (\pi_2^{-1}(X_{U1} \oplus C), X_{U2}) \\
\Longrightarrow & /* \text{ no edge in } F' \text{ is along dimension } n \text{ and } (X_{D1}, X_{D2}) \text{ is along dimension } n \, */ \\
 & F' \leftrightarrow \begin{pmatrix} \pi_2^{-1}(X_{U1} \oplus C) & X_{U2} \\ \pi_2^{-1}(X_{D1} \oplus C) & X_{D2} \end{pmatrix} \equiv Z \\
\Longrightarrow & /* \text{ from Lemma 2 } */ \\
 & F \leftrightarrow \pi_1^{-1}(Z) \\
\Longrightarrow & /* \text{ from } X_{hc} \approx Z \approx \pi_1^{-1}(Z) \text{ and Lemma 1 } */ \\
 & G(\pi_1^{-1}(Z)) \text{ is a fault-free Hamiltonian cycle.}
\end{array}
$$

□

Comments. We have shown that a level-n BR-tree and a level-n complete binary tree (resp., a Hamiltonian cycle) can always be embedded in an n-cube with no more than $n-1$ (resp., $n-2$) faulty links. It is not hard to see that when there are n (resp., $n-1$) faulty links all incident to a same node, it is impossible to find a level-n BR-tree (resp., a Hamiltonian cycle) in the n-cube. Thus, the result has reached the worst-case bounds for these graphs. However, as commented earlier, if faulty links occur in only a small number of dimensions, the embedding algorithm has potential to tolerate more faults.

The time needed to compute the X' in Theorem 1, given any f faulty links, can be analyzed as follows. To sort f links (step 1) takes $O(f \log f)$ time. To compute the vectors $B_i, i = 1..r$, (step 2) needs f comparisons. To compute the final edge matrix (step 3) needs n^2 steps. So the overall time complexity is $O(f \log f + n^2)$.

4 Embedding Product Graphs

The *product* of two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, denoted as $G_1 \times G_2$, is a graph $G = (V, E)$ such that $V = \{(v_1, v_2) : v_1 \in V_1 \wedge v_2 \in V_2\}$ and $E = \{\langle (v_1, v_2), (v_1', v_2') \rangle : (v_1 = v_1' \wedge \langle v_2, v_2' \rangle \in E_2) \vee (v_2 = v_2' \wedge \langle v_1, v_1' \rangle \in E_1)\}$. The product of more than two graphs is similarly defined. For instance, an n-cube is the product of n graphs each of which consists of two nodes connected by an edge, and a torus is the product of a number of rings.

Let $H = H_1 \times H_2 \times \cdots \times H_k$ be the product of k graphs, where each H_i is a level-d_i BR-tree and $d_1 + d_2 + \cdots + d_k = n$. We study the problem of embedding H in an injured

$$X = \begin{pmatrix} \xi & * & * & * & * & * & * \\ 0 & \xi & * & * & * & * & * \\ 1 & 0 & \xi & * & * & * & * \\ * & * & * & \xi & * & * & * \\ * & * & * & 1 & \xi & * & * \\ * & * & * & 1 & 1 & \xi & * \\ * & * & * & 1 & 1 & 1 & \xi \end{pmatrix}$$

Figure 4: An edge matrix representing the product of a length-8 linear path and a level-4 binomial tree.

n-cube with faulty links. Without loss of generality, assume $1 \leq d_1 \leq d_2 \leq \cdots \leq d_k$. Also, let $d'_0 = 0$ and $d'_i = \sum_{j=1}^{i} d_j$ for $i = 1..k$.

Since H_i is a level-d_i BR-tree, by definition it is isomorphic to a subgraph of the d_i-cube which can be represented as a $d_i \times d_i$ edge matrix, say, X_i conforming to Eq. (1). Using these $X_i, i = 1..k$, the graph $H = H_1 \times H_2 \times \cdots \times H_k$ can be embedded in an n-cube as $G(X)$, where X is an $n \times n$ edge matrix such that

$$X = \begin{pmatrix} X_1 & *_{1,2} & \cdots & *_{1,k} \\ *_{2,1} & X_2 & \cdots & *_{2,k} \\ \vdots & \vdots & \ddots & \vdots \\ *_{k,1} & *_{k,2} & \cdots & X_k \end{pmatrix}, \tag{3}$$

where $*_{i,j}$ is a $d_i \times d_j$ matrix of $*$'s. Figure 4 illustrates a 7×7 edge matrix X (with $d_1 = 3$ and $d_2 = 4$), in which $X[1..3][1..3]$ represents a length-8 linear path and $X[4..7][4..7]$ does a level-4 binomial tree (see also Lemma 3).

Let F be the edge matrix representing the set of faulty links. In the following, we describe a heuristic procedure that tries to find a column-permutation function π and a binary vector C such that $X \oplus C \leftrightarrow \pi(F)$. If the procedure succeeds, then $G(\pi^{-1}(X \oplus C))$ is a fault-free embedding of graph H. Without loss of generality, let $F = (F_1, F_2, \ldots, F_r)^T$ be well-formed.

Procedure *Product_Graph*()

Step 1: Partition the $F_1, F_2, \ldots, F_r$ into k groups, say $\hat{F}_1, \hat{F}_2, \ldots, \hat{F}_k$, such that for each $i = 1..k$ (i) $\hat{F}_i$ contains a collection of F_j's in F and is represented as an edge matrix and (ii) if $\hat{F}_i = (F_{i_1}, \ldots, F_{i_s})^T$, where each F_{i_t} is an F_j in the original F, then the inequality

$\sum_{t=1}^{s} \lceil \log(1 + |E(F_{i_t})|) \rceil \leq d_i - 1$ holds. We write such partition as a new matrix

$$\hat{F} = \begin{pmatrix} \hat{F}_1 \\ \hat{F}_2 \\ \vdots \\ \hat{F}_k \end{pmatrix}.$$

See Fig. 5 for an example of such partition. (Note that adjacent F_j's in F are not necessarily assigned to a same group. Also note that in some cases, for example, when $r < k$, some $\hat{F}_i$ might be null.) The above condition (ii) will enable us to use the proof technique in Theorem 1.

Step 2: Then, find a column-permutation π on $\hat{F}$, i.e.,

$$\pi(\hat{F}) = \begin{pmatrix} \pi(\hat{F}_1) \\ \pi(\hat{F}_2) \\ \vdots \\ \pi(\hat{F}_k) \end{pmatrix},$$

such that (i) each $\pi(\hat{F}_i)$ contains only edges along dimensions from $d'_{i-1} + 1$ to d'_i and (ii) $\pi(\hat{F}_i)[-][d'_{i-1} + 1..d'_i]$ is dimension-sorted.

Step 3: For each $i = 1..k$, applying procedure *Binary_Vector*() on X_i and $\pi(\hat{F}_i)[-][d'_{i-1} + 1..d_i]$, we can obtain a length-d_i binary vector C_i such that

$$X_i \oplus C_i \leftrightarrow \pi(\hat{F}_i)[-][d'_{i-1} + 1..d_i].$$

The above equation implies a disjoint relation in an n-cube:

$$(*_{i,1}, \ldots, *_{i,i-1}, X_i, *_{i,i+1}, \ldots, *_{i,k}) \oplus (C', C_i, C'') \leftrightarrow \pi(\hat{F}_i)[-][-] = \pi(\hat{F}_i) \qquad (4)$$

where C' is *any* length-d'_{i-1} binary vector, and C'' *any* length-$(n - d'_i)$ binary vector. Note that the left-most matrix of the above equation is a submatrix of X, from row $d'_{i-1} + 1$ to row d'_i. As the binary vector $(C_1, C_2, \ldots, C_k)$ is a candidate for (C', C_i, C'') in Eq. (4), by substituting $(C_1, C_2, \ldots, C_k)$ for (C', C_i, C'') and putting together Eq. (4) for $i = 1..k$, we obtain:

$$\begin{pmatrix} (X_1, *_{1,2}, \ldots, *_{1,k}) \oplus (C_1, C_2, \ldots, C_k) \\ (*_{2,1}, X_2, \ldots, *_{2,k}) \oplus (C_1, C_2, \ldots, C_k) \\ \vdots \\ (*_{k,1}, *_{k,2}, \ldots, X_k) \oplus (C_1, C_2, \ldots, C_k) \end{pmatrix} = X \oplus (C_1, C_2, \ldots, C_k) \leftrightarrow \begin{pmatrix} \pi(\hat{F}_1) \\ \pi(\hat{F}_2) \\ \vdots \\ \pi(\hat{F}_k) \end{pmatrix} = \pi(\hat{F}).$$

Since $\hat{F}$ is only a row-permutation of F, the aforementioned goal is achieved and we are done.

Example 2: Figure 5 shows how *Product_Graph*() works for the edge matrix X of Fig. 4 and an F with 6 faulty links. In step 1, F is row-permuted into an $\hat{F} = (\hat{F}_1, \hat{F}_2)^T$ that satisfies conditions (i) and (ii) of step 1. In step 2, π permutes $\hat{F}$ into $\pi(\hat{F}_1)$ that satisfies conditions (i) and (ii) of step 2. In step 3, C_1 and C_2 are computed using procedure *Binary_Vector*() with the elements highlighted in X and $\pi(\hat{F})$. Note that now the relation $X \oplus (C_1, C_2) \leftrightarrow \pi(\hat{F})$ holds. □

In the above approach, steps 2 and 3 are fairly easy to achieve (and always achievable). Unfortunately, step 1 is not always doable, and to determine whether the F matrix can be so partitioned is an NP-hard problem (when $k = 2$ and $d_1 = d_2$, the problem becomes the well-known, NP-complete *PARTITION Problem* [6]).

One may readily observe that $n-k$ is a worst-case upper bound on the number of faulty links that any algorithm can tolerate, since $G(X)$ is a graph of degree greater than or equal to k. One may also observe that when there are no more than $d_k - 1$ faulty links in the n-cube, step 1 can be easily done by letting $\hat{F}_k = F$ (so all other $\hat{F}_i, i = 1..k-1$, are null). In the following, we present a heuristic algorithm for step 1 that provides a degree of fault tolerance much higher than $d_k - 1$, in some cases near $n - k$.

Procedure *k-Partition*() /* A heuristic for step 1 of *Product_Graph*() */

a) Sort $F = (F_1, \ldots, F_r)^T$ into $F' = (F'_1, \ldots, F'_r)^T$, where each F'_i is some F_j, such that $|E(F'_i)| \leq |E(F'_j)|$ iff $i \leq j$.

b) $r' := r$;

c) **for** $i := k$ **downto** 1 **do**

   i) Let $\hat{F}_i := (F'_{r''}, F'_{r''+1}, \ldots, F'_{r'})^T$, where r'' is the minimum possible value such that $\sum_{j=r''}^{r'} \lceil \log(1 + |E(F'_j)|) \rceil \leq d_i - 1$.

   ii) $r' := r'' - 1$;

d) **if** $r' \leq 0$ **then** "Continue Steps 2 and 3"; /* The algorithm works. */
   else "Abort the algorithm"; /* The algorithm fails. */

The above heuristic sorts the F_i's of F in ascending order according to the number of faulty links in F_i (step a) and computes $\hat{F}_i$ from $i = k$ down to 1 in a greedy manner (step c). It is not hard to see that if all $F'_1, \ldots, F'_r$ are successfully assigned to $\hat{F}_1, \ldots, \hat{F}_k$, then *k-Partition*() (and thus *Product_Graph*()) succeeds. The following theorem derives the degree of fault tolerance guaranteed by the above algorithm.

$$\text{step 1: } F = \begin{pmatrix} 1 & \xi & 0 & 0 & 0 & 1 & 1 \\ 1 & \xi & 0 & 1 & 0 & 0 & 1 \\ 0 & \xi & 1 & 1 & 0 & 0 & 1 \\ 1 & \xi & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & \xi & 0 \\ 0 & 1 & 1 & 0 & \xi & 0 & 1 \end{pmatrix} \Longrightarrow \hat{F} = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & \xi & 0 \\ 0 & 1 & 1 & 0 & \xi & 0 & 1 \end{pmatrix} \equiv \hat{F}_1 \\ \begin{pmatrix} 1 & \xi & 0 & 0 & 0 & 1 & 1 \\ 1 & \xi & 0 & 1 & 0 & 0 & 1 \\ 0 & \xi & 1 & 1 & 0 & 0 & 1 \\ 1 & \xi & 0 & 0 & 1 & 1 & 1 \end{pmatrix} \equiv \hat{F}_2 \Bigg)$$

$$\text{step 2: } \hat{F} \Longrightarrow \pi(\hat{F}) = \begin{pmatrix} \boxed{0} & \xi & 0 & 0 & 1 & 0 & 0 \\ 0 & \boxed{0} & \xi & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & \boxed{0} & \boxed{0} & \boxed{1} & \xi \\ 1 & 0 & 0 & \boxed{0} & \boxed{1} & \boxed{1} & \xi \\ 0 & 0 & 0 & \boxed{1} & \boxed{1} & \boxed{1} & \xi \\ 1 & 1 & 1 & \boxed{0} & \boxed{0} & \boxed{1} & \xi \end{pmatrix}, \text{ where } \pi = \begin{cases} 1 \rightarrow 1 \\ 2 \rightarrow 7 \\ 3 \rightarrow 4 \\ 4 \rightarrow 5 \\ 5 \rightarrow 3 \\ 6 \rightarrow 2 \\ 7 \rightarrow 6 \end{cases}$$

$$\text{step 3: } X = \begin{pmatrix} \xi & * & * & * & * & * & * \\ \boxed{0} & \xi & * & * & * & * & * \\ 1 & \boxed{0} & \xi & * & * & * & * \\ * & * & * & \xi & * & * & * \\ * & * & * & 1 & \xi & * & * \\ * & * & * & 1 & 1 & \xi & * \\ * & * & * & \boxed{1} & \boxed{1} & \boxed{1} & \xi \end{pmatrix} \Longrightarrow \begin{array}{l} C_1 = (1, 1, \underline{0}) \\ C_2 = (0, 1, 1, \underline{0}) \\ \text{Note: the underlined bits are} \\ \qquad \text{randomly chosen.} \end{array}$$

Figure 5: The application of procedure *Product_Graph*() to the X in Fig. 4, assuming an F with 6 faulty links.

Theorem 2 *Let $H = H_1 \times \cdots \times H_k$ be a product graph, with each H_i being a BR-tree of level d_i, where $d_1 + \cdots + d_k = n$ and $d_1 \leq \cdots \leq d_k$. If an n-cube has no more than f_k faulty links, then Product_Graph() can embed H into the cube without using any faulty link, where f_k is recursively defined as*

$$f_i = \begin{cases} d_1 - 1 & \text{if } i = 1 \\ \max\{d_i - 1, \min\{2^{d_i - 1} - 1, f_{i-1} + d_i - \lceil \log d_i \rceil\}\} & \text{if } i = 2..k. \end{cases} \tag{5}$$

Proof. It suffices to show that procedure *k-Partition*() will successfully assign $F'_1, \ldots, F'_r$ to $\hat{F}_1, \ldots, \hat{F}_k$, provided that $|E(F)| \leq f_k$. The proof is by induction on k.

INDUCTION BASE: When $k = 1$, $f_1 = n - 1$ and $d_1 = n$. By Lemma 5, procedure *k-Partition*() will assign all $F'_1, \ldots, F'_r$ to $\hat{F}_1$. So the induction base is established.

INDUCTION STEP: When $k > 1$, assuming that the above claim is true for $k - 1$, we prove it for k. This is done by considering $E(\hat{F}_k)$ and $E(F) - E(\hat{F}_k)$ in two different cases:

Case 1: If $d_k - 1 \geq \min\{2^{d_k - 1} - 1, f_{k-1} + d_k - \lceil \log d_k \rceil\}$, then by definition $f_k = d_k - 1$. As $|E(F)| \leq f_k = d_k - 1$, by Lemma 5 again, procedure *k-Partition*() will assign all $F'_1, \ldots, F'_r$ to $\hat{F}_k$. So we are done.

Case 2: Otherwise, $f_k = \min\{2^{d_k - 1} - 1, f_{k-1} + d_k - \lceil \log d_k \rceil\}$. Consider three possibilities regarding F'_r (which is the largest matrix among $F'_1, \ldots, F'_r$):

a) If $|E(F'_r)| \geq 2^{d_k - 1}$, then $|E(F)| \geq 2^{d_k - 1}$, which contradicts with $|E(F)| \leq f_k \leq 2^{d_k - 1} - 1$ (the last inequality follows from the definition of f_k). So this case is impossible.

b) If $2^{d_k - 1} - 1 \geq |E(F'_r)| \geq d_k - 1$, then F'_r will be assigned to $\hat{F}_k$. Since $d_k \neq 1$ (otherwise $|E(F'_r)| = 0$),

$$|E(F) - E(\hat{F}_k)| \leq f_k - (d_k - 1) \leq f_k - (d_k - \lceil \log d_k \rceil) \leq f_{k-1}, \tag{6}$$

where the last inequality is obtained from the definition of f_k. By the induction hypothesis, procedure *k-Partition*() will successfully assign $E(F) - E(\hat{F}_k)$ to $\hat{F}_1, \ldots, \hat{F}_{k-1}$ because $|E(F) - E(\hat{F}_k)| \leq f_{k-1}$.

c) If $d_k - 1 > |E(F'_r)|$, then let $\hat{F}_k = (F'_{r''}, F'_{r''+1}, \ldots, F'_r)^T$ with r'' being as small as possible. There are two cases for the value of r'':

   i) If $r'' = 1$, then all $F'_1, \ldots, F'_r$ are consumed and we are done.

   ii) If $r'' > 1$, we will show that $E(\hat{F}_k)$ contains at least $d_k - \lceil \log d_k \rceil$ elements. If so, a reasoning similar to Eq. (6) will show $|E(F) - E(\hat{F}_k)| \leq f_{k-1}$, and the proof

will follow directly from the induction hypothesis. That $|E(\hat{F}_k)| \geq d_k - \lceil \log d_k \rceil$ can be established as follows:

$$
\begin{aligned}
& /* \text{ as } r'' \text{ is set to the minimum possible } */ \\
& \sum_{i=r''-1}^{r} \lceil \log(1 + |E(F'_i)|) \rceil \geq d_k \\
\Longrightarrow \quad & \sum_{i=r''}^{r} \lceil \log(1 + |E(F'_i)|) \rceil \geq d_k - \lceil \log(1 + |E(F'_{r''-1})|) \rceil \\
\Longrightarrow \quad & /* \text{ by assumption, } d_k - 1 > |E(F'_r)| \geq |E(F'_{r''-1})|, \text{ which implies} \\
& \qquad \lceil \log d_k \rceil \geq \lceil \log(1 + |E(F'_{r''-1})|) \rceil \; */ \\
& \sum_{i=r''}^{r} \lceil \log(1 + |E(F'_i)|) \rceil \geq d_k - \lceil \log d_k \rceil \\
\Longrightarrow \quad & /* \text{ as } |E(F'_i)| \geq \lceil \log(1 + |E(F'_i)|) \rceil \text{ for any } i \; */ \\
& |E(\hat{F}_k)| = \sum_{i=r''}^{r} |E(F'_i)| \geq d_k - \lceil \log d_k \rceil
\end{aligned}
$$

□

Embedding equilateral product graphs such as square meshes is of particular importance in applications. When all H_i's in $H = H_1 \times H_2 \times \cdots \times H_k$ are BR-trees of the same level, the f_k in Theorem 2 can be expressed in a simpler form, and its value gets closer to the worst-case bound $n - k$, as shown in the following corollary.

Corollary 4 *Let $H = H_1 \times \cdots \times H_k$ be a product graph, with each H_i being a BR-tree of level d and $kd = n$. If there are no more than f_k faulty links in the n-cube, then H can be embedded in the cube without using any faulty link, where*

$$
\begin{aligned}
f_k &= \min\{2^{d-1} - 1, n - (k-1)\lceil \log d \rceil - 1\} \\
&= \min\{\Theta(2^d), n - \Theta(k \log d)\}.
\end{aligned}
$$

Proof. This can be proved by first substituting each d_i in Eq. (5) with d and then showing inductively that $f_i = \min\{2^{d-1} - 1, (i-1)(d - \lceil \log d \rceil) + d - 1\}$ for $i = 1..k$. □

A torus is a product of a number of rings. If $H = H_1 \times \cdots \times H_k$, where each H_i is a ring of length 2^{d_i}, then H can be embedded in a fault-free n-cube using the edge matrix X in Eq. (3) by substituting each X_i in Eq. (3) with one conforming to Eq. (2) (i.e., $G(X_i)$ is a Hamiltonian cycle in a d_i-cube). Due to the resemblance between the matrix representing a BR-tree and that representing a ring, the above results concerning the numbers of tolerable faults for embedding a product of BR-trees can be easily modified

to ones for tori. In particular, we have the following theorem, which is rephrased from Theorem 2 by substituting each d_i with $d_i - 1$.

Theorem 3 *Consider a* $2^{d_1} \times 2^{d_2} \times \cdots \times 2^{d_k}$ *torus such that* $\sum_{i=1}^{k} d_i = n$ *and* $d_1 \leq d_2 \leq \cdots \leq d_k$. *There exists an fault-free embedding of such a torus in an n-cube as long as there are no more than* f'_k *faulty links in the cube, where* f'_k *is recursively defined as*

$$f'_i = \begin{cases} d_1 - 2 & \text{if } i = 1. \\ \max\{d_i - 2, \min\{2^{d_i - 2} - 1, f'_{i-1} + d_i - 1 - \lceil \log(d_i - 1) \rceil\}\} & \text{if } i = 2..k. \end{cases} \quad (7)$$

Proof. (sketched) This theorem can be obtained by re-deriving Theorem 2 with the following minor modifications: (1) substitute each occurrence of d_i with $d_i - 1$ and (2) whenever the derivation uses Lemma 5, prove it using the technique applied in Corollary 3. □

5 Conclusion

We have reported new results on the embedding a wide range of graphs — rings, BR-trees, complete binary trees, and products of these graphs — in injured hypercubes that contain faulty links. Through these results, we also have demonstrated how the new data structure, edge matrix, can facilitate the otherwise difficult embedding job, especially when the hypercube contains only faulty links. We are currently investigating the possibility of applying our matrix technique to the case where nodes may also fail.

References

[1] F. Berman and L. Snyder. On mapping parallel algorithms into parallel architectures. *J. of Parallel and Distrib. Comput.*, 4:439–458, 1987.

[2] S. Bokhari. On the mapping problem. *IEEE Trans. on Comput.*, C-30:207–214, 1981.

[3] M. Y. Chan and S.-J. Lee. Distributed fault-tolerant embeddings of rings in hypercubes. *J. of Parallel and Distrib. Comput.*, 11:63–71, 1991.

[4] M. Y. Chan and S.-J. Lee. Fault-tolerant embeddings of complete binary trees in hypercubes. *IEEE Trans. on Comput.*, 4(3):277–288, March 1993.

[5] G. Cybenko, D. W. Krumme, and K. N. Venkataraman. Fixed hypercube embedding. *Info. Process. Letts.*, 25:35–39, April 1987.

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

[7] S. L. Johnsson and C. T. Ho. Optimal broadcasting and personalized communication in hypercubes. *IEEE Trans. on Comput.*, 38(9):1249–68, Sep. 1989.

[8] Y. M. Kim and T.-H. Lai. The complexity of congestion-1 embedding in a hypercube. *J. of Algorithm* **12**, pages 246–280, 1991.

[9] T.-H. Lai and W. White. Mapping pyramid algorithms into hypercubes. *J. of Parallel and Distrib. Comput.*, 9(1):42–54, 1990.

[10] S. Latifi, S. Zheng, and N. Bagherzadeh. Optimal ring embedding in hypercubes with faulty links. In *Fault-Tolerant Computing Symp.*, pages 178–184, 1992.

[11] V. M. Lo *et al.* Mapping divide-and-conquer algorithm to parallel architectures. In *Int'l Conf. on Parallel Processing*, pages III.128–135, 1990.

[12] Y. Saad and M. H. Schultz. Topological properties of hypercubes. *IEEE Trans. on Comput.*, 37(7):867–872, July 1988.

[13] A. Sen, A. Sengupta, and S. Bandyopadhyay. On some topological properties of hypercube, incomplete hypercube and supercube. In *Int'l Parallel Processing Symp.*, pages 636–642, 1993.

[14] Y.-C. Tseng and T.-H. Lai. Ring embedding in an injured hypercube. In *Int'l Conf. on Parallel Processing*, pages III–149–152, 1993.

[15] Y.-C. Tseng, T.-H. Lai, and L.-F. Wu. Matrix representation of graph embedding in a hypercube. *J. of Parallel and Distrib. Comput.*, To appear.

[16] R. Varadarajan. Embedding shuffle networks in hypercubes. *J. of Parallel and Distrib. Comput.*, 11:252–256, 1991.

[17] A. Wang and R. Cypher. Fault-tolerant embeddings of rings, meshes, and tori in hypercubes. In *IEEE Symp. on Parallel and Distributed Processing*, pages 20–29, 1992.

[18] A. Wang, R. Cypher, and E. Mayr. Embedding complete binary trees in faulty hypercubes. In *IEEE Symp. on Parallel and Distributed Processing*, pages 112–119, 1991.

[19] A. Wanger and D. Corneil. Embedding trees in the hypercube is NP-complete. *SIAM J. of Comput.*, 19(4):570–590, 1990.

[20] A. S. Wanger. Embedding arbitrary binary trees in a hypercube. *J. of Parallel and Distrib. Comput.*, 7:503–520, 1989.

[21] A. Y. Wu. Embedding of tree networks in hypercubes. *J. of Parallel and Distrib. Comput.*, 2:238–249, 1985.

[22] P.-J. Yang, S.-B. Tien, and C. S. Raghavendra. Embedding of multidimensional meshes on to faulty hypercubes. In *Int'l Conf. on Parallel Processing*, pages I–571–574, 1991.