Distributed algorithms for building Hamiltonian cycles in k-ary n-cubes and hypercubes with faulty links

Iain A. Stewart
Department of Computer Science, Durham University,
Science Labs, South Road, Durham DH1 3LE, U.K.

Abstract

We derive a sequential algorithm `Find-Ham-Cycle` with the following property. On input: k and n (specifying the k-ary n-cube Q_n^k); F, a set of at most $2n-2$ faulty links; and $\mathbf{v}$, a node of Q_n^k, the algorithm outputs nodes $\mathbf{v}^+$ and $\mathbf{v}^-$ such that if `Find-Ham-Cycle` is executed once for every node $\mathbf{v}$ of Q_n^k then the node $\mathbf{v}^+$ (resp. $\mathbf{v}^-$) denotes the successor (resp. predecessor) node of $\mathbf{v}$ on a fixed Hamiltonian cycle in Q_n^k in which no link is in F. Moreover, the algorithm `Find-Ham-Cycle` runs in time polynomial in n and $\log k$. We also obtain a similar algorithm for an n-dimensional hypercube with at most $n-2$ faulty links. We use our algorithms to obtain distributed algorithms to embed Hamiltonian cycles k-ary n-cubes and hypercubes with faulty links; our hypercube algorithm improves on a recently-derived algorithm due to Leu and Kuo, and our k-ary n-cube algorithm is the first distributed algorithm for embedding a Hamiltonian cycle in a k-ary n-cube with faulty links.

Keywords: interconnection networks; k-ary n-cubes; hypercubes; fault-tolerance; Hamiltonian cycles; distributed algorithms; embeddings.

1 Introduction

Numerous interconnection networks have been proposed as underlying topologies for parallel computers. As to which network is chosen depends upon a number of factors relating to the topological, algorithmic and communication properties of the network and the types of problems to which the resulting computer is to be applied. One of the most popular interconnection networks is undoubtedly the n-dimensional hypercube Q_n. Some of its pleasing properties, with regard to parallel computation, include: it is node- and link-symmetric; it is Hamiltonian; it has diameter n; it has a recursive decomposition; and it contains, or "nearly" contains (as subgraphs), almost all interconnection networks currently vogue in parallel computing (see [29] for these results and more on the hypercube). Some of the commercial machines whose underlying topology is based on the hypercube are the Cosmic Cube [37], the Ametek S/14 [8], the iPSC [21, 22], the Ncube [14, 22] and the CM-200 [15].

However, every node of Q_n has degree n, and consequently, as n increases so does the degree of every node. High degree nodes in interconnection networks can lead to technological problems in parallel computers whose underlying topology is that of the said interconnection network. One method of circumventing this problem, so as to still retain a "hypercube-like" interconnection network, is to build parallel computers so that the underlying topology is the k-ary n-cube Q_n^k. The k-ary n-cube Q_n^k is similar in essence to the hypercube, but by a judicious choice of k and n we can include a large number of nodes yet keep the degree of each node fixed. For example, the hypercube Q_{12} has 4096 nodes and every node has degree 12. However, Q_3^{16} has 4096 nodes and every node has degree 6. Of course, one usually loses out in some other respect (for example, in terms of diameter) but often this loss is not too catastrophic. The k-ary n-cube Q_n^k has not been investigated to the same extent as the hypercube, but it is known to have the following properties (amongst many others): it is node- and link-symmetric [7]; it is Hamiltonian [9, 12]; it has diameter $n\lfloor k/2 \rfloor$ [9, 12]; it has a recursive decomposition; and it contains many important interconnection networks such as cycles (of certain lengths) [7], meshes (of certain dimensions) [9] and even hypercubes (of certain dimensions) [12]. Machines whose underlying topology is based on a k-ary n-cube include the Mosaic [38], the iWARP [11], the J-machine [32], the Cray T3D [26] and the Cray T3E [5].

In parallel computers which have a large number of processors, it is not uncommon for nodes or links between nodes to fail. This experience has motivated investigations into how able different interconnection networks are to cope with a limited number of node and/or link faults. Most investigations have focussed on the resulting communication capabilities or embeddability of faulty interconnection networks, e.g., whether a certain network is still connected or whether certain guest networks can still be embedded in certain host networks, given a limited number of faults. Also, most research has centered around hypercubes; see, for example, [10, 13, 14, 16–19, 24, 27, 28, 30, 31, 34–36, 39–46] for recent work. The k-ary n-cube has not been considered to such a great extent although there is some existing research; see, for example, [1–4, 6, 20].

In this paper, we are primarily interested in the distributed embedding of a Hamiltonian cycle in a k-ary n-cube when some of the links are faulty, and secondarily in a hypercube with faulty links. The existence of a Hamiltonian cycle in an interconnection network is extremely useful as, for one thing, it facilitates all-to-all broadcasts, with messages being "daisy-chained" around the cycle. Also, the existence of a Hamiltonian path between two nodes enables algorithms designed for linear arrays to be simulated in the (faulty) k-ary n-cube. It is well known that Q_n and Q_n^k possess fault-free Hamiltonian cycles in the presence of at most $n-2$ and $2n-2$ faulty links, respectively; indeed, under the additional modest assumption that a node in Q_n or Q_n^k is incident with at least 2 healthy links, there are still Hamiltonian cycles in Q_n and Q_n^k when there are $2n-5$ [16] and $4n-5$ [6] faulty links, respectively, and these results are optimal. However, the existence of a Hamiltonian cycle is not necessarily sufficient for algorithmic viability as the cycle needs to be constructed, and not by a centralized algorithm which necessarily takes time exponential in n, but by a distributed algorithm which hopefully will run in polynomial-time and have minimal message-passing overhead.

Our main result is the derivation of a sequential algorithm `Find-Ham-Cycle` with the following property. On input: k and n (specifying the k-ary n-cube Q_n^k); F, a set of at most $2n-2$ faulty links; and $\mathbf{v}$, a node of Q_n^k, the algorithm outputs nodes

$\mathbf{v}^+$ and $\mathbf{v}^-$ such that if `Find-Ham-Cycle` is executed once for every node $\mathbf{v}$ of Q_n^k then the node $\mathbf{v}^+$ (resp. $\mathbf{v}^-$) denotes the successor (resp. predecessor) node of $\mathbf{v}$ on a fixed Hamiltonian cycle in Q_n^k in which no link is in F. Moreover, the algorithm `Find-Ham-Cycle` runs in time polynomial in n and $\log k$. We also obtain a similar algorithm for an n-dimensional hypercube with at most $n-2$ faulty links. Note that executing our algorithms at each node of a distributed-memory multiprocessor whose underlying topology is a k-ary n-cube or a hypercube results in an embedding of a fault-free Hamiltonian cycle, assuming that each node has complete knowledge of all faulty links but where no message passing is required. Consequently, in order to use our algorithms in a distributed fashion, all we need to do is to disseminate knowledge of which links are faulty, which is a problem that has been well studied.

In Section 2, we give the basic definitions relating to the content of this paper, before outlining and commenting on related work in Section 3. In Section 4, we develop our main algorithm for k-ary n-cubes, and in Section 5 we show how this algorithm can be adapted for hypercubes. Finally, we show how our algorithms can be applied in Section 6.

2 Basic definitions

The *k-ary n-cube* Q_n^k, for $k \geq 2$ and $n \geq 2$, has k^n nodes indexed by $\{0, 1, \ldots, k-1\}^n$, and there is a link $((x_1, x_2, \ldots, x_n), (y_1, y_2, \ldots, y_n))$ if, and only if, there exists $j \in \{1, 2, \ldots, n\}$ such that $min\{|x_j - y_j|, k - |x_j - y_j|\} = 1$, and $x_i = y_i$, for every $i \in \{1, 2, \ldots, n\} \setminus \{j\}$ (for example, Q_2^k is a $k \times k$ mesh with wrap-around). When $k = 2$, Q_n^2 is the *hypercube of dimension n* and we denote it simply Q_n. When $k \geq 3$, Q_n^k can be thought of as the direct (graph) product of n cycles of length k, or the Cayley graph of the finitely generated group $\langle g_1, g_2, \ldots, g_k : g_i^k = 1, g_i^{-1} g_j^{-1} g_i g_j = 1, \text{ for all } i, j = 1, 2, \ldots, k\rangle$ [9].

Each component of an n-tuple describing the nodes of Q_n^k is called a *dimension*, and Q_n^k can be *partitioned over a dimension i* by regarding it as consisting of k disjoint copies of Q_{n-1}^k, where the jth copy is induced by the nodes of Q_n^k in which the value of the ith component is fixed at $j \in \{0, 1, \ldots, k-1\}$ (corresponding nodes of these k copies of Q_{n-1}^k are joined by a link, if $k = 2$, and joined in a cycle of length k, if $k \geq 3$). We denote the nodes of Q_n^k in bold type, e.g., $\mathbf{x}$, so as to reflect their representation as n-tuples. The *Lee distance* between two nodes $\mathbf{x}$ and $\mathbf{y}$ of Q_n^k is defined as

$$\sum_{i=1}^{n} min\{|x_i - y_i|, k - |x_i - y_i|\}$$

(so, two nodes are joined in Q_n^k if, and only if, the Lee distance between them is 1).

A *faulty link*, or simply *fault*, in Q_n^k is a link which can be considered as missing. We assume that the nodes incident with a faulty link are fully cognisant that this link is faulty, and that the fault does not affect the nodes' performance. Any link that is not faulty is *healthy*. An *oriented* graph in an undirected graph is a subgraph where each link has had an orientation imposed so as to transform it into a directed subgraph.

3 Related work

The problem of developing a feasible distributed algorithm for the algorithmic construction of a Hamiltonian cycle in a faulty hypercube was first tackled in [17]. In this paper, Chan and Lee obtain a distributed algorithm for a distributed-memory multiprocessor whose underlying interconnection network is an n-dimensional hypercube, which builds a cycle of size at least $2^n - 2f$ in the hypercube when $f \leq \lfloor (n+1)/2 \rfloor$ nodes are faulty. They assume that initially each node has only local knowledge of which nodes are faulty; that is, each node is aware only of which of its neighbours are faulty (all links are assumed to be healthy). After construction of the cycle, each node on the cycle knows its successor and predecessor on the cycle. Chan and Lee do not explicitly define their model of computation but only say how their own particular algorithm is executed on this model. Their model is such that initially a given source node is active. This source node computes and then activates, at some appropriate point, some of its neighbours and these neighbours begin to compute. These neighbours then activate some of their neighbours, and so forth. Nodes can be activated at some future point even if they have already been activated. Chan and Lee's algorithm essentially simultaneously executes $2f - 1$ cycle-building algorithms each of which is based on a distinct (binary-reflected) Gray code. Chan and Lee do not concern themselves with a complexity analysis but one can easily see that their algorithm involves $\Omega((2^n - 2f)(2f - 1))$ node activations and also has the (severe) drawback that $\Omega(2^n)$ space is required (and consequently $\Omega(2^n)$ time) at each node (in order to compute and store a Gray code).

Chan and Lee mention in their conclusion that there is a possibility of deriving a similar distributed algorithm for the situation where the hypercube has at most $\lfloor (n+1)/2 \rfloor$ faulty links and where the resulting cycle will be Hamiltonian. They hint that this might be accomplished by using link-disjoint Hamiltonian cycles in a hypercube although they leave as open the question of whether this can actually be done. Following their suggestions would almost surely result in a distributed algorithm which, like Chan and Lee's faulty-node algorithm, requires $\Omega(2^n)$ space at each node.

Latifi, Zheng and Bagherzadeh [27, 28] tackle the question posed by Chan and Lee and develop a centralized sequential algorithm which builds a Hamiltonian cycle in an n-dimensional hypercube when at most $n - 2$ links are faulty. We use the term centralized in order to convey the fact that all faults are known *a priori* to the algorithm. What they do is to develop an $O(n^2)$ time algorithm which on being given a set of at most $n - 2$ faulty links produces a characterization of a fault-free Hamiltonian cycle. This characterization is a permutation of $\{1, 2, \ldots, n\}$ which encodes the construction of a Gray code, i.e., a Hamiltonian cycle in the hypercube. Of course, in order to obtain the actual Hamiltonian cycle, the permutation needs to be expanded, which takes $\Omega(2^n)$ time. So, although Latifi, Zheng and Bagherzadeh's algorithm can be used to construct a Hamiltonian cycle in a hypercube with faulty links, in a distributed fashion, full knowledge of all faulty links must be acquired by each node and the subsequent expansion of the characterization of the cycle by each node takes $\Omega(2^n)$ time.

Significant progress was made in [30] where Leu and Kuo develop a distributed algorithm for a distributed-memory multiprocessor whose underlying interconnection network is an n-dimensional hypercube, which builds a Hamiltonian cycle in the hypercube when at most $n-2$ links are faulty. They assume that initially each node has global knowledge of which links are faulty but remark that even if each node only has knowledge of the links incident with it which are are faulty, a broadcast algorithm due to Park and Bose [33] can be used to distribute this local knowledge so that every node acquires global knowledge of all the faulty links in the network. Leu and Kuo's algorithm works by incrementally joining appropriate cycles so as to avoid faulty links and ultimately obtain a fault-free Hamiltonian cycle. Their algorithm runs in polynomial-time. However, it requires message-passing between nodes in the construction of the cycle (once global fault knowledge has been acquired by each node) which should, in theory, be avoidable. Wang, Leu and Kuo extend the research in [30] when in [43] they embed not only Hamiltonian cycles in hypercubes but also other topologies.

In this paper, we develop distributed algorithms to construct Hamiltonian cycles in a distributed-memory multiprocessor whose underlying interconnection network is a k-ary n-cube or an n-dimensional hypercube when there is a limited number of faulty links. Our approach extends and improves that adopted by Leu and Kuo, above, with the result that we obtain an improved hypercube algorithm and a new k-ary n-cube algorithm. As far as we are aware, ours is the first feasible distributed algorithm for constructing Hamiltonian cycles in k-ary n-cubes with faulty links.

4 Finding Hamiltonian cycles

Throughout this section, $k \geq 3$ and $n \geq 2$ unless otherwise stated. We detail a sequential algorithm `Find-Ham-Cycle` with the following property. The algorithm has input variables:

- k and n, which specify Q_n^k;
- F, a set of at most $2n-2$ faulty links in Q_n^k; and
- $\mathbf{v}$, a node of Q_n^k,

and output variables:

- $\mathbf{v}^+$ and $\mathbf{v}^-$, nodes of Q_n^k,

and is such that if `Find-Ham-Cycle(`k`,`n`,`F`,`$\mathbf{v}$`;`$\mathbf{v}^+$`,`$\mathbf{v}^-$`)` is executed once for every node $\mathbf{v}$ of Q_n^k (with the same set of faults F each time) then the node $\mathbf{v}^+$ (resp. $\mathbf{v}^-$) denotes the successor (resp. predecessor) node of $\mathbf{v}$ on a fixed, oriented Hamiltonian cycle in Q_n^k in which no link is in F (we already know that such a Hamiltonian cycle exists by [6], although our analysis and constructions yield an alternative proof of this fact). Moreover, the algorithm `Find-Ham-Cycle` runs in time polynomial in n and $\log k$. Consequently, if we were to execute the algorithm `Find-Ham-Cycle(`k,n,F,$\mathbf{v}$;$\mathbf{v}^+$,$\mathbf{v}^-$`)` at every node $\mathbf{v}$ of a machine whose underlying topology is that of the k-ary n-cube Q_n^k and in which there are at most $2n-2$ faulty links (thus we assume that every node has global knowledge relating to the set of faults) then after termination, every node would know its successor and predecessor nodes on a (fault-free) oriented Hamiltonian cycle, and this information would have been obtained in polynomial time (in n and $\log k$) and without sending any messages amongst nodes.

4.1 Basic construction

Let Q_n^k have at most $2n-2$ faults, for some $n \geq 3$, which we denote by the set F. The basic structure of our algorithm is as follows, although, as we shall see, there are some subtleties.

- Partition Q_n^k over some dimension d so as to obtain k copies, Q_0, Q_1, …, Q_{k-1}, of Q_{n-1}^k in which the total number of faults, summing over all copies of Q_{n-1}^k, is at most $2(n-1)-2$.
- Superimpose all the faults in Q_0, Q_1, …, Q_{k-1} onto one copy of Q_{n-1}^k; call it P (thus there are at most $2(n-1)-2$ faults in P).
- Recursively find a Hamiltonian cycle C in P.
- The Hamiltonian cycle C in P is also a Hamiltonian cycle C_i in Q_i, for $i = 0, 1, \ldots, k-1$. Join these cycles together using links in dimension d so as to obtain a Hamiltonian cycle in Q_n^k (which avoids all the faults).

By "join", above, we mean the following. Choose two distinct links $(\mathbf{x}, \mathbf{y})$ and $(\mathbf{x}, \mathbf{y}')$ of the Hamiltonian cycle C. These links correspond to the links $(\mathbf{x}_i, \mathbf{y}_i)$ and $(\mathbf{x}_i, \mathbf{y}'_i)$ of C_i, for $i = 0, 1, \ldots, k-1$, and there are links $(\mathbf{x}_i, \mathbf{x}_{i+1})$, $(\mathbf{y}_i, \mathbf{y}_{i+1})$ and $(\mathbf{y}'_i, \mathbf{y}'_{i+1})$ in a healthy Q_n^k, for $i = 0, 1, \ldots, k-1$ with addition modulo k (of course, in our faulty Q_n^k some of these links may be faulty, but more of this later). We add and remove links as follows.

- For each $i \in \{1, 2, \ldots, k-2\}$, remove the links $(\mathbf{x}_i, \mathbf{y}_i)$ and $(\mathbf{x}_i, \mathbf{y}'_i)$ from C_i, and remove the link $(\mathbf{x}_0, \mathbf{y}'_0)$ from C_0. If $k-1$ is even (resp. odd) then remove the link $(\mathbf{x}_{k-1}, \mathbf{y}_{k-1})$ (resp. $(\mathbf{x}_{k-1}, \mathbf{y}'_{k-1})$) from C_{k-1}.
- For even $i \in \{0, 1, \ldots, k-2\}$, include the links $(\mathbf{x}_i, \mathbf{x}_{i+1})$ and $(\mathbf{y}'_{i+1}, \mathbf{y}'_i)$. For odd $i \in \{0, 1, \ldots, k-2\}$, include the links $(\mathbf{x}_i, \mathbf{x}_{i+1})$ and $(\mathbf{y}_{i+1}, \mathbf{y}_i)$.

The construction can be visualized as in Fig. 1, where k is odd and where we have provided an orientation on the links so that a directed cycle results.

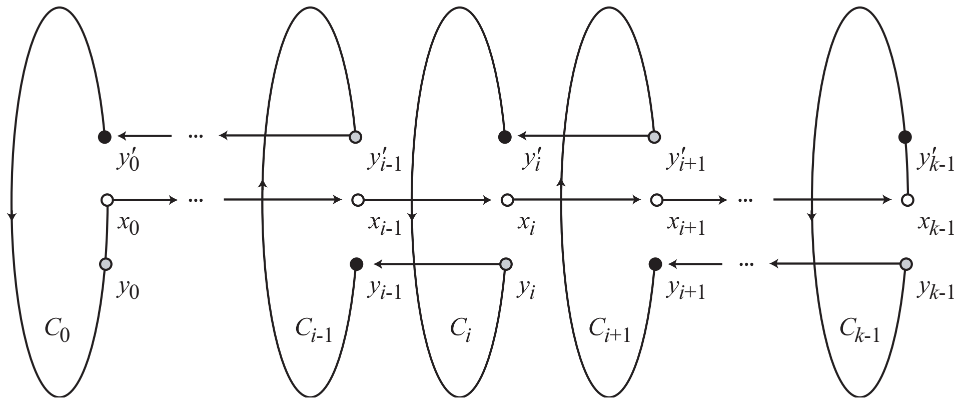

Figure 1. Joining the cycles C_0, C_1, …, C_{k-1} (where $k-1$ is even).

Thus, we obtain a Hamiltonian cycle in Q_n^k. However, as we have just mentioned, in a faulty k-ary n-cube Q_n^k we must ensure that we can find two links $(\mathbf{x},\mathbf{y})$ and $(\mathbf{x},\mathbf{y}')$ in C so that none of the links in dimension d used to join the Hamiltonian cycles C_0, C_1, …, C_{k-1} are faulty. Also, we need to deal separately with the base case of the recursion, when $n = 2$.

Apart from the above consideration, deriving an algorithm would be easy if we were not concerned that the algorithm should run in polynomial time (in n and $\log k$). As any (sequential) algorithm which finds a Hamiltonian cycle in a k-ary n-cube Q_n^k necessarily runs in time $\Omega(k^n)$, we seek a (parallel) algorithm which we can run (in polynomial time) at every node of a machine whose underlying topology is that of our faulty k-ary n-cube Q_n^k so that a Hamiltonian cycle is found but where every node only has local knowledge of this Hamiltonian cycle. Of course, this local knowledge is enough for us to utilize the Hamiltonian cycle for message routing and so on. Furthermore, in such a distributed algorithm we must also ensure that orientations of recursively-constructed cycles are monitored when these oriented cycles are joined together to former a larger oriented cycle.

At first glance, it might appear relatively straightforward to obtain such an algorithm. However, consider the final act of joining together the cycles C_0, C_1, …, C_{k-1} in Q_n^k. Suppose that a decision as to which links $(\mathbf{x},\mathbf{y})$ and $(\mathbf{x},\mathbf{y}')$ of C are chosen (over which to join the cycles C_0, C_1, …, C_{k-1}) is deferred until after the recursive calls which build C. Each node $\mathbf{v}$ only has local knowledge of C and consequently only knows for sure that the links $(\mathbf{v}^-,\mathbf{v})$ and $(\mathbf{v},\mathbf{v}^+)$ are links of C. If node $\mathbf{v}$ decides to join the cycles over these two links (or, more precisely, if in the execution of the algorithm at node $\mathbf{v}$ we decide to join the cycles over these two links), how does some other node $\mathbf{u}$ of Q_n^k know to do likewise, given that $\mathbf{u}$ only has local knowledge of the Hamiltonian cycle C? Also, it is not obvious as to how the links $(\mathbf{x},\mathbf{y})$ and $(\mathbf{x},\mathbf{y}')$ of C can be decided upon before preforming the recursive calls which build C; as how can we be sure that any chosen links will end up being in the resulting Hamiltonian cycle C?

What we do is to find a node $\mathbf{z}$ of P with the property that no matter which two of its neighbours are the successor and predecessor on an oriented Hamiltonian cycle within C, it will be possible to join the resulting cycles C_0, C_1, …, C_{k-1} using the two links corresponding to those involving $\mathbf{z}$ and its predecessor and successor.

4.2 Joining cycles

Consider our k-ary n-cube Q_n^k in which there is a set F of at most $2n-2$ faults. Let d be the dimension in which most faults of F occur. Partition Q_n^k over dimension d to obtain k copies, Q_0, Q_1, …, Q_{k-1}, of Q_{n-1}^k. Note that the total number of faults, summing over Q_0, Q_1, …, Q_{k-1}, is at most $2(n-1)-2$, if $n \geq 3$, or at most 1, if $n = 2$. Let P be a copy of Q_{n-1}^k in which all the faults in Q_0, Q_1, …, Q_{k-1} have been superimposed. For each fault $(\mathbf{a},\mathbf{b})$ of F (in Q_n^k), if $(\mathbf{a},\mathbf{b})$ is in dimension d then denote by $\mathit{proj}((\mathbf{a},\mathbf{b}),d)$ the node (of Q_{n-1}^k)

$$(a_1,\ldots,a_{d-1},a_{d+1},\ldots,a_n)\ (= (b_1,\ldots,b_{d-1},b_{d+1},\ldots,b_n)),$$

and if $(\mathbf{a},\mathbf{b})$ is not in dimension d then denote by $\mathit{proj}((\mathbf{a},\mathbf{b}),d)$ the link (of Q_{n-1}^k)

$$((a_1,\ldots,a_{d-1},a_{d+1},\ldots,a_n),(b_1,\ldots,b_{d-1},b_{d+1},\ldots,b_n)).$$

Let D be the set (of nodes of P) of such projections of all faults in dimension d. Hence, $|D| \leq 2n-2$, if $n \geq 3$. We shall use the related operation *expand* later: if $\mathbf{u} = (u_1, u_2, \ldots, u_{n-1}) \in \{0, 1, \ldots, k-1\}^{n-1}$, $w \in \{0, 1, \ldots, k-1\}$ and $d \in \{1, 2, \ldots, n\}$ then *expand*$(\mathbf{u}, w, d)$ is the node $(u_1, \ldots, u_{d-1}, w, u_d, \ldots, u_{n-1})$ of Q_n^k.

Fix $n \geq 3$ until further notice. We wish to choose canonical links $(\mathbf{x}, \mathbf{y})$ and $(\mathbf{x}, \mathbf{y}')$ of the recursively built Hamiltonian cycle C of P so that $\{\mathbf{x}, \mathbf{y}, \mathbf{y}'\} \cap D = \emptyset$. In particular, we look for a node $\mathbf{x} \in P \setminus D$ none of whose neighbours in P is in D. We now verify that such a node $\mathbf{x}$ exists.

A simple counting argument proves the existence of $\mathbf{x}$ in most cases. Suppose that such a node $\mathbf{x}$ did not exist. Thus, every node of P must be adjacent to a node of D. The number of nodes adjacent to a node of D is at most $(2n-2)2(n-1)$. Hence, $(2n-2) + (2n-2)2(n-1) = (2n-2)(2n-1) \geq k^{n-1}$. This yields a contradiction whenever ($k \geq 5$ and $n \geq 3$) or ($k = 4$ and $n \geq 4$) or ($k = 3$ and $n \geq 5$), i.e., whenever $(k, n) \notin \{(3,3), (3,4), (4,3)\}$. So, in these cases $\mathbf{x}$ exists, and an algorithm to find $\mathbf{x}$ can be constructed which examines $O(n^2)$ nodes of P.

However, we can do better (and this constructive affirmation will result in an improved algorithm to find such a node $\mathbf{x}$ which need only examine $O(n)$ nodes of P). Suppose that $k \geq 5$ and $n \geq 5$. Define the set of nodes S of P as follows.

$$\begin{aligned} S \;=\;& \{(a_1, a_2, \ldots, a_{n-2}, 0) : (a_i, a_{i+1}) = (1, 2),\ \text{for some } i \in \{1, 2, \ldots, n-3\}, \\ & \text{and } a_j = 0,\ \text{for } j \in \{1, 2, \ldots, n-2\} \setminus \{i, i+1\}\} \cup \{(2, 0, \ldots, 0, 1, 0)\} \\ \cup\;& \{(a_1, a_2, \ldots, a_{n-2}, 1) : (a_i, a_{i+1}) = (2, 1),\ \text{for some } i \in \{1, 2, \ldots, n-3\}, \\ & \text{and } a_j = 0,\ \text{for } j \in \{1, 2, \ldots, n-2\} \setminus \{i, i+1\}\} \cup \{(1, 0, \ldots, 0, 2, 1)\} \\ \cup\;& \{(3, 3, \ldots, 3, 3), (4, 4, \ldots, 4, 4), (0, 0, \ldots, 0, 2)\}. \end{aligned}$$

Suppose that $k \geq 5$ and $n = 4$. Define the set of nodes S of P as follows.

$$S = \{(0,0,0), (1,2,0), (2,4,0), (4,3,0), (3,1,0), (2,2,2), (3,3,3)\}.$$

Suppose that $k \geq 5$ and $n = 3$. Define the set of nodes S of P as follows.

$$S = \{(0,0), (1,2), (2,4), (4,3), (3,1)\}.$$

Suppose that $k = 4$ and $n \geq 4$. Define the set of nodes S of P as follows.

$$\begin{aligned} S \;=\;& \{(a_1, a_2, \ldots, a_{n-1}) : (a_i, a_{i+1}) = (1, 2),\ \text{for some } i \in \{1, 2, \ldots, n-2\}, \\ & \text{and } a_j = 0,\ \text{for } j \in \{1, 2, \ldots, n-1\} \setminus \{i, i+1\}\} \cup \{(2, 0, \ldots, 0, 1)\} \\ \cup\;& \{(a_1, a_2, \ldots, a_{n-1}) : (a_i, a_{i+1}) = (2, 1),\ \text{for some } i \in \{1, 2, \ldots, n-2\}, \\ & \text{and } a_j = 3,\ \text{for } j \in \{1, 2, \ldots, n-1\} \setminus \{i, i+1\}\} \cup \{(1, 3, \ldots, 3, 2)\} \\ \cup\;& \{((0, 0, \ldots, 0)\}. \end{aligned}$$

Suppose that $k = 3$ and $n \geq 5$. Define the set of nodes S of P as follows.

$$\begin{aligned} S \;=\;& \{(a_1, a_2, \ldots, a_{n-2}, 1) : (a_i, a_{i+1}) = (1, 2),\ \text{for some } i \in \{1, 2, \ldots, n-3\}, \\ & \text{and } a_j = 0,\ \text{for } j \in \{1, 2, \ldots, n-2\} \setminus \{i, i+1\}\} \cup \{(2, 0, \ldots, 0, 1, 1)\} \\ \cup\;& \{(a_1, a_2, \ldots, a_{n-2}, 2) : (a_i, a_{i+1}) = (2, 1),\ \text{for some } i \in \{1, 2, \ldots, n-3\}, \\ & \text{and } a_j = 0,\ \text{for } j \in \{1, 2, \ldots, n-2\} \setminus \{i, i+1\}\} \cup \{(1, 0, \ldots, 0, 2, 2)\} \\ \cup\;& \{(0, 0, \ldots, 0, 0), (1, 1, \ldots, 1, 0), (2, 2, \ldots, 2, 0)\}. \end{aligned}$$

In particular, in all cases $|S| = 2n-1$ and the Lee distance between any two nodes of S is at least 3. Thus, as $|D| \leq 2n-2$, there must exist at least one node, $\mathbf{x}$, of S that is adjacent to no node of D. Furthermore, we can easily devise an algorithm, call it `Find-x`, which, on being given k and n, where n and k fall into one of the above cases, and also a set of nodes D, finds such a node $\mathbf{x}$ by searching the nodes of S in a canonical, pre-determined order. It is straightforward to see that this algorithm can be implemented to run in $O(n^3 \log k)$ time.

Now consider the case when $n = 2$ and there are 2 faulty links; we need to deal with the 'base case' of our forthcoming recursive algorithm. Suppose further that $k \geq 4$ (it is trivial to derive a Hamiltonian cycle for the case when $n = 2$ and $k = 3$ and when there are 2 faulty links). Without loss of generality, either both faults lie in dimension 1, or one lies in dimension 1 and one in dimension 2; moreover, we may assume (by link-symmetry) that one dimension 1 fault, f_1, is the link $((0,0),(k-1,0))$. Consequently, the other fault, f_2, is of the form $((i',j'),(i'+1,j'))$ (a dimension 1 fault) or of the form $((i',j'),(i',j'+1))$ (a dimension 2 fault).

If f_2 is a dimension 1 fault then we can join the dimension 2 cycles just as we did earlier, as depicted in Fig. 1, ensuring that we avoid the fault f_2, which is possible as $k \geq 4$ (note that the fault f_1 is not used to join cycles regardless). If f_2 is a dimension 2 fault then we need to join $k-1$ cycles and one path (each of which lies within dimension 2). We do this exactly as we did earlier except that we ensure that f_2 is one of the links omitted in our construction, as depicted in Fig. 1.

4.3 Implementation details

We can now present our algorithm. Recall that this algorithm is only to describe the resulting oriented Hamiltonian cycle in a localized fashion. As expected, all addition and subtraction of values from $\{0,1,\ldots,k-1\}$ is modulo k.

```
Find-Ham-Cycle(input variables
   $k,n$   :  parameters of $Q_n^k$
   $F$     :  set of at most $2n-2$ link faults
   $\mathbf{v}$     :  node at which this algorithm is executed
output variables
   $\mathbf{v}^+$    :  successor of $\mathbf{v}$ on the Hamiltonian cycle
   $\mathbf{v}^-$    :  predecessor of $\mathbf{v}$ on the Hamiltonian cycle)
begin

/* Deal with the base cases of the recursion.  */

if ($n=3$ and ($k=3$ or $k=4$)) or ($n=4$ or $k=3$) then
   return the successor and predecessor nodes on a
   canonical Hamiltonian cycle
else
   if $n=2$ then
      return $\mathbf{v}^+$ and $\mathbf{v}^-$ from Find-Ham-Cycle-n=2($k$,$F$,$\mathbf{v}$;$\mathbf{v}^+$,$\mathbf{v}^-$)

/* Find the dimension $d$ in which most faults lie.  */
```

```
if there are no faults in $F$ then
   $d$ := $1$
else
   let $d$ be the dimension containing most faults

/* Partition over dimension $d$ to obtain $k$-ary $(n-1)$-   */
/* cubes $Q_0$, $Q_1$, ..., $Q_{k-1}$, and superimpose the faults */
/* in $Q_0$, $Q_1$, ..., $Q_{k-1}$ on a $k$-ary $(n-1)$-cube $P$ to    */
/* obtain the fault set $G$.  Build the set of nodes $D$. */
/* Recall, if $(\mathbf{y},\mathbf{z})$ is in dimension $d$ then $proj((\mathbf{y},\mathbf{z}),d)$ */
/* is the node $(y_1,\ldots,y_{d-1},y_{d+1},\ldots,y_n)$; otherwise         */
/* $proj((\mathbf{y},\mathbf{z}),d)$ is the link $((y_1,\ldots,y_{d-1},y_{d+1},\ldots,y_n),(z_1,$ */
/* $\ldots,z_{d-1},z_{d+1},\ldots,z_n))$.                           */

$G$ := $\emptyset$ /* will consist of at most $2n-4$ faults of $Q^k_{n-1}$ */
$D$ := $\emptyset$ /* will consist of at most $2n-2$ nodes of $Q^k_{n-1}$  */
for every $(\mathbf{y},\mathbf{z}) \in F$ do
   if $(\mathbf{y},\mathbf{z})$ is in dimension $d$ then
      put the node $proj((\mathbf{y},\mathbf{z}),d)$ in $D$
   else
      put the link $proj((\mathbf{y},\mathbf{z}),d)$ in $G$

/* Obtain a node $\mathbf{x}$ of $P$ so that the resulting cycles $C_0$, $C_1$, */
/* ..., $C_{k-1}$ are to be joined over two links incident with $\mathbf{x}$. */

Find-x($k$,$n-1$,$D$;$\mathbf{x}$)

/* Recursively find a Hamiltonian cycle $C$ in $P$.  */

$\mathbf{u}$ := $proj(\mathbf{v},d)$ and $w$ := $d$th component of $\mathbf{v}$
Find-Ham-Cycle($n-1$,$k$,$G$,$\mathbf{u}$;$\mathbf{u}^+$,$\mathbf{u}^-$)

/* Obtain $\mathbf{v}^+$ and $\mathbf{v}^-$.  Recall, $expand(\mathbf{u},w,d)$ is $(u_1,\ldots,u_{d-1},w,$ */
/* $u_d,\ldots,u_{n-1})$.                                         */

$\mathbf{v}^+$ := $expand(\mathbf{u}^+,w,d)$ and $\mathbf{v}^-$ := $expand(\mathbf{u}^-,w,d)$

/* Reverse the direction of the cycles $C_0$, $C_2$, $C_4$, ... */

if $w$ is even then swap $\mathbf{v}^+$ and $\mathbf{v}^-$

/* Join the cycles $C_0$, $C_1$, ..., $C_{k-1}$.  */

if $\mathbf{u}=\mathbf{x}$ then
   if $w \neq 0$ and $w \neq k-1$ then
      $\mathbf{v}^+$ := $expand(\mathbf{u},w+1,d)$ and $\mathbf{v}^-$ := $expand(\mathbf{u},w-1,d)$
   if $w=0$ then $\mathbf{v}^+$ := $expand(\mathbf{u},1,d)$
```

```
      if $w = k-1$ then $\mathbf{v}^-$ := $expand(\mathbf{u}, k-2, d)$
   else
      if $\mathbf{u}^+$ := x then
         if $w \neq 0$ then $\mathbf{v}^+$ := $expand(\mathbf{u}, w-1, d)$
      else
         if $\mathbf{u}^-$ := x then
            if $w \neq k-1$ then $\mathbf{v}^-$ := $expand(\mathbf{u}, w+1, d)$
   end
```

Assuming that `Find-Ham-cycle-n=2(`k`,`F`,`$\mathbf{v}$`;`$\mathbf{v}^+$`,`$\mathbf{v}^-$`)` finds an oriented Hamiltonian cycle in a k-ary 2-cube, the algorithm `Find-Ham-Cycle` essentially follows the construction as described in Sections 4.1 and 4.2 and as depicted in Fig. 1. With reference to Fig. 1 and the final part of the algorithm, nodes for which $\mathbf{u} = \mathbf{x}$ are coloured white, nodes for which $\mathbf{u}^+ = \mathbf{x}$ are coloured grey, and nodes for which $\mathbf{u}^- = \mathbf{x}$ are coloured black. We remark that the "canonical Hamiltonian cycle" for the cases when $(n, k) \in \{(3,3), (3,4), (4,3)\}$ exists (by [6], for example) and can be derived by hand or using a simple computer program.

```
   Find-Ham-Cycle-n=2(input variables
      $k$   :  parameter of $Q_2^k$
      $F$   :  set of at most 2 faults
      $\mathbf{v}$   :  node at which this algorithm is executed
   output variables
      $\mathbf{v}^+$ :  successor of v on the Hamiltonian cycle
      $\mathbf{v}^-$ :  predecessor of v on the Hamiltonian cycle)
   begin

   if $k = 3$ then
      return the successor and predecessor nodes on a
      canonical Hamiltonian cycle

   /* Reorient the $k$-ary 2-cube if necessary.  */

   if dimension 2 has more faults than dimension 1 then
      swap dimensions via a transformation $\sigma$ (so that now
      dimension 1 has most faults)
   if dimension 1 has at least 1 fault then
      rename nodes via a transformation $\tau$ so that one of the
      dimension 1 faults is $((0,0),(k-1,0))$

   /* So apart from possibly $((0,0),(k-1,0))$, there is at most  */
   /* one other fault $f$, either of the form $((i',j'),(i',j'+1))$     */
   /* or $((i',j'),(i'+1,j'))$.                                    */

   suppose that $\mathbf{v}$ = $(i,j)$

   /* Initialize successor and predecessor nodes.  */
```

```
if $i$ is even then
   $\mathbf{v}^+$ := $(i, j-1)$ and $\mathbf{v}^-$ := $(i, j+1)$
else
   $\mathbf{v}^+$ := $(i, j+1)$ and $\mathbf{v}^-$ := $(i, j-1)$

/* Find links over which to join the dimension 2 cycles/paths */
/* so as to avoid faults.                                      */

if $f = ((i', j'), (i', j'+1))$ then
   if $w = k-1$ and $w$ is odd then
      $x$ := $j'+1$
   else
      $x$ := $j'$
else
   if $f = ((i', j'), (i'+1, j'))$ then $x$ := $j'+2$

/* Join the cycles/paths.  */

if $j = x$ then
   if $i \neq 0$ and $i \neq k-1$ then $\mathbf{v}^+$ := $(i+1, j)$ and $\mathbf{v}^-$ := $(i-1, j)$
   if $i = 0$ then $\mathbf{v}^+$ := $(i+1, j)$
   if $i = k-1$ then $\mathbf{v}^-$ := $(i-1, j)$
else
   if $j = x-1$ then
      if $i \neq k-1$ is even then
         $\mathbf{v}^-$ := $(i+1, x-1)$
      else
         if $i$ is odd then $\mathbf{v}^+$ := $(i-1, x-1)$
   else
      if $j = x+1$ then
         if $i \neq 0$ is even then
            $\mathbf{v}^+$ := $(i-1, x+1)$
         else
            if $i \neq k-1$ is odd then $\mathbf{v}^-$ := $(i+1, x+1)$

/* Un-reorient the $k$-ary 2-cube if necessary.  */

if $\tau$ was performed earlier then perform $\tau^{-1}$
if $\sigma$ was performed earlier then perform $\sigma^{-1}$
end
```

In the algorithm `Find-Ham-Cycle-n=2`, the reorientation τ of Q_2^k is easily performed by adding appropriate offsets to the components of nodes (and subtracting the same offsets in order to perform τ^{-1}); the same offsets apply for every node. Otherwise, the algorithm essentially implements the construction as described in Sections 4.1 and 4.2 and as depicted in Fig. 1, except where cycles/paths to be joined lie in dimension 2 in Q_2^k. Note that the choice of x always ensures that if there is a fault in dimension 2 then the construction is such that this fault is not used in the resulting Hamiltonian cycle.

The algorithm `Find-Ham-Cycle-n=2` trivially has time complexity $O(\log k)$ and apart from the recursive call, the time complexity of `Find-Ham-Cycle` is $O(n^3 \log k)$ (which is the time complexity of `Find-x`). Hence, the time complexity $f(n,k)$ of `Find-Ham-Cycle` satisfies the recurrence

$$f(n,k) \leq f(n-1,k) + \alpha n^3 \log k,$$

for some constant α, and so $f(n,k) = O(n^4 \log k)$. Hence, we have the following result.

Theorem 1 *The algorithm* `Find-Ham-Cycle` *takes as input the parameters n and k of Q_n^k, a set F of at most $2n-2$ faulty links in Q_n^k and a node $\mathbf{v}$ of Q_n^k and outputs two nodes $\mathbf{v}^+$ and $\mathbf{v}^-$ of Q_n^k. If we fix the parameters n, k and F and we run* `Find-Ham-Cycle` *once for every node $\mathbf{v}$ of Q_n^k as input then*:

- *the oriented graph induced by the directed links $\{(\mathbf{v},\mathbf{v}^+) : \mathbf{v} \in Q_n^k\}$ is an oriented Hamiltonian cycle in Q_n^k in which no faulty link appears*; *and*
- *the oriented graph induced by the directed links $\{(\mathbf{v},\mathbf{v}^-) : \mathbf{v} \in Q_n^k\}$ is the same Hamiltonian cycle but oriented in the opposite direction.*

Furthermore, the algorithm `Find-Ham-Cycle` *has time complexity $O(n^4 \log k)$.* □

5 The hypercube

The algorithm `Find-Ham-Cycle` can be adapted so that it similarly gives a Hamiltonian cycle in an n-dimensional hypercube Q_n with at most $n-2$ faulty links, where $n \geq 3$. This adaptation, which we outline here, is conceptually simpler than `Find-Ham-Cycle`. (Recall that Chan and Lee have already shown that a Hamiltonian cycle exists in Q_n in the presence of $n-2$ link faults [16].)

The basic structure of our algorithm `Find-Ham-Cycle-Hyper` for our faulty Q_n is identical to that of `Find-Ham-Cycle`. We begin by partitioning Q_n over some dimension d in which there is at least 1 fault (or over any dimension if there are no faults), and then superimpose the faults not in dimension d on an $(n-1)$-dimensional hypercube P so that P has at most $n-3$ faulty links. We also project all faulty links in dimension d so as to obtain a set D of nodes of P of size at most $n-2$, as before.

We then determine a node $\mathbf{x}$ of P such that $\mathbf{x} \notin D$ and nor is any of its neighbours (our counting argument, as used earlier, verifies that such an $\mathbf{x}$ always exists). After recursively finding a Hamiltonian cycle C in P, we obtain two isomorphic Hamiltonian cycles, C_0 and C_1, in the two "halves" of our original faulty Q_n, joined by links in dimension d. We then join C_0 and C_1 using the links in dimension d whose projections are $\mathbf{x}$ and one of its neighbours in C, just as we did before.

The following corollary follows from the preceding analysis.

Corollary 2 *There is an algorithm* `Find-Ham-Cycle-Hyper` *which takes as input the parameter $n \geq 3$ of Q_n , a set F of at most $n-2$ faulty links in Q_n and a node $\mathbf{v} \in Q_n$ and outputs two nodes $\mathbf{v}^+$ and $\mathbf{v}^-$ of Q_n. If we fix the parameters n and F and we run* `Find-Ham-Cycle-Hyper` *once for every node $\mathbf{v}$ of Q_n as input then*:

- *the oriented graph induced by the directed links* $\{(\mathbf{v}, \mathbf{v}^+) : \mathbf{v} \in Q_n\}$ *is an oriented Hamiltonian cycle in* Q_n *in which no faulty link appears*; *and*
- *the oriented graph induced by the directed links* $\{(\mathbf{v}, \mathbf{v}^-) : \mathbf{v} \in Q_n\}$ *is the same Hamiltonian cycle but oriented in the opposite direction.*

Furthermore, the algorithm `Find-Ham-Cycle-Hyper` *has time complexity* $O(n^4)$. $\square$

6 Applying our algorithms

As stated earlier, our intention is to develop an algorithm for a distributed-memory multiprocessor whose underlying interconnection network is, primarily, the k-ary n-cube or, secondarily, the n-dimensional hypercube, so that after execution of the algorithm, every node knows its successor and a predecessor on a Hamiltonian cycle within the underlying network, even when at most $2n-2$ links are faulty in the case of the k-ary n-cube or at most $n-2$ links are faulty in the case of the n-dimensional hypercube. Moreover, we should assume that each node only has local knowledge of which of its incident links are faulty. Clearly, we are almost done as all we need to do is to arrange for all fault information to be disseminated throughout the network prior to execution of our algorithm. Luckily, considerable work as been done on this problem.

One simple way of disseminating the fault information is to obtain a spanning tree (which contains only healthy links) of depth d. In a multi-port model, where nodes can both transmit and receive messages to and from any number of neighbours in one step, the time taken to disseminate all fault information is $2d$. We are assuming, quite reasonably, that information relating to more than one fault can be bundled together into one message, given that there are at most $2n-2$ faults in the k-ary n-cube and the description of each fault can be squeezed into only $n\lceil\log(k)\rceil + \lceil\log(n)\rceil + 1$ bits (the situation is even more favourable for the hypercube). In a one-port model, where in any one step any node can transmit at most one message to a neighbour and receive at most one message from a neighbour, the time taken is easily seen to be at most $2d\Delta$, where Δ is the maximal degree of any node of the tree (note that a tree can be edge-coloured using at most Δ colours and this edge-colouring allows us to send messages along every bi-directional link in Δ steps). The above is only a very rudimentary method for disseminating the fault information but is sufficient for our needs (this paper's main concern is with building Hamiltonian cycles rather than performing all-to-all broadcasts).

In [23], it is shown that a k-ary n-cube has at least $2n$ link-disjoint spanning trees of depth at most $n\lfloor\frac{k}{2}\rfloor + k - 1$, and in [25] it is shown that an n-dimensional hypercube has at least n link-disjoint spanning trees of depth at most $n+1$. Hence, we have our distributed algorithms as described above. Note that our hypercube algorithm is vastly superior to that of Chan and Lee's (when incorporated within their model of computation) and also improves upon that of Leu and Kuo (as once we have disseminated the fault information, we do not need to send any further messages). Of course, our distributed algorithm for the k-ary n-cube is the first such feasible algorithm (irrespective of the underlying model of computation).

References

[1] B. Almohammad and B. Bose, Fault-tolerant communication in toroidal networks, *IEEE Trans. Parallel Distrib. Syst.* **10** (1999) 976–983.

[2] J. Al-Sadi, K. Day and M. Ould-Khaoua, Unsafety vectors: a fault-tolerant routing for k-ary n-cubes, *Microprocess. Microsys.* **25** (2001) 239–246.

[3] J. Al-Sadi, K. Day and M. Ould-Khaoua, Probability vectors: a new fault-tolerant routing algorithm for k-ary n-cubes, *Proc. of. ACM Symp. on Applied Computing*, ACM Press (2002) 830–834.

[4] J. Al-Sadi, K. Day and M. Ould-Khaoua, A new probabilistic approach for fault-tolerant routing in k-ary n-cubes, *Proc. of 9th Int. Conf. on Parallel and Distributed Systems*, IEEE Press (2002) 509–516.

[5] E. Anderson, J. Brooks, C. Grassl and S. Scott, Performance of the Cray T3E multiprocessor, *Proc. of ACM/IEEE Confrence on Supercomputing*, ACM Press (1997) 1–17.

[6] Y.A. Ashir and I.A. Stewart, Fault-tolerant embeddings of Hamiltonian circuits in k-ary n-cubes, *SIAM J. Disc. Math.* **15** (2002) 317–328.

[7] Y.A. Ashir and I.A. Stewart, On embedding cycles in k-ary n-cubes, *Parallel Process. Lett.* **7** (1997) 49–55.

[8] W.C. Athas and C.L. Seitz, Multicomputers: message-passing concurrent computers, *Computer* **21** (1988) 9–24.

[9] S. Bettayeb, On the k-ary hypercube, *Theoret. Comput. Sci.* **140** (1995) 333–339.

[10] F. Bao, Y. Igarashi and R. Öhring, Reliability of hypercubes for broadcasting with random faults, *IEICE Trans. Inf. Syst.* **E79-D** (1996) 22–28.

[11] S. Borkar, R. Cohen, G. Cox, S. Gleason, T. Gross, H.T. Kung, M. Lam, B. Moore, C. Peterson, J. Pieper, L. Rankin, P.S. Tseng, J. Sutton, J. Urbanski and J. Webb, iWarp: An integrated solution to high-speed parallel computing, *Proc. of Supercomputing* '88, IEEE Press (1988) 330–339.

[12] B. Bose, B. Broeg, Y. Kwon and Y. Ashir, Lee distance and topological properties of k-ary n-cubes, *IEEE Trans. Computers* **44** (1995) 1021–1030.

[13] J. Bruck, On optimal broadcasting in faulty hypercubes, *Disc. Applied Maths.* **53** (1994) 3–14.

[14] Y. Bruck, R. Cypher and C.-T. Ho, Efficient fault-tolerant mesh and hypercube architectures, *Proc. of 22nd Int. Symp. on Fault-Tolerant Computing*, IEEE Press (1992) 162–169.

[15] J.-P. Brunet and S.L. Johnsson, All-to-all broadcast and applications on the connection machine, *Int. J. Supercomput. Applications* **6** (1992) 241–256.

[16] M.Y. Chan and S.-J. Lee, On the existence of Hamiltonian circuits in faulty hypercubes, *SIAM J. Disc. Maths.* **4** (1991) 511–527.

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

[18] S.-C. Chau and A. W.-C. Fu, An optimal $(d-1)$-fault-tolerant all-to-all broadcasting scheme for d-dimensional hypercubes, *Proc. of 3rd Int. Conf. on Parallel and Distributed Computing, Applications and Technologies*, IEICE (2002) 351–356.

[19] G. Chiu, A fault-tolerant broadcasting algorithm for hypercubes, *Inform. Process Lett.* **66** (1998) 93–99.

[20] K. Day, The conditional node connectivity of the k-ary n-cube, *J. Interconnect. Networks* **5** (2004) 13–26.

[21] R. Duncan, A survey of parallel computer architectures, *Computer* **23** (1990) 5–16.

[22] T.H. Duncan, Performance of the Intel iPSC/860 and Ncube 6400 hypercubes, *Parallel Comput.* **17** (1991) 1285–1302.

[23] P. Fraigniaud and C. Laforest, Disjoint spanning trees of small depth, *Proc. of Parallel Computing*: *Trends and Application* (G. Joubert, D. Trystram, F. Peters, D.Evans, eds.), Elsevier (1994) 105-112.

[24] J. Fu, Fault-tolerant cycle embedding in the hypercube, *Parallel Comput.* **29** (2003) 821–832.

[25] S.L. Johnsson and C.T. Ho, Optimum broadcasting and personalized communication in hypercubes, *IEEE Trans. Comput.* **38** (1989) 1249–1268.

[26] R.E. Kessler and J.L. Schwarzmeier, CRAY T3D: a new dimension for Cray research, *Proc. of 38th IEEE Computer Society Int. Conf.*, IEEE Press (1993) 176–182.

[27] S. Latifi, S.Q. Zheng and N. Bagherzadeh, Optimal ring embedding in hypercubes with faulty links, *Proc. of 22nd Int. Symp. on Fault-Tolerant Computing*, IEEE Press (1992) 178–184.

[28] S. Latifi, S.Q. Zheng and N. Bagherzadeh, Hamiltonian path and cycle in hypercubes with faulty links, *Proc. of* 5th IEEE Int. Conf. on Algorithms and Architectures for Parallel Processing, IEEE Press (2002) 178–184.

[29] F.T. Leighton, *Introduction to Parallel Algorithms and Architectures*: *Arrays. Trees. Hypercubes*, Morgan Kaufmann (1992).

[30] Y. Leu and S. Kuo, Distributed fault-tolerant ring embedding and reconfiguration in hypercubes, *IEEE Trans. Comput.* **48** (1999) 81–88.

[31] T. Li, C. Tsai, J. Tan and L. Hsu, Bipanconnectivity and edge-fault-tolerant bipanciclicity of hypercubes, *Inform. Process Lett.* **87** (2003) 107-110.

[32] M.D. Noakes, D.A. Wallach and W.J. Dally, The J-machine multicomputer: an architectural evaluation, *Proc. of 20th Ann. Int. Symp. on Computer Architecture*, IEEE Press (1993) 224–235.

[33] S. Park and B. Bose, Broadcasting in hypercubes with link/node failures, *Proc. of 4th Symp. on Frontiers of Massively Parallel Computation*, IEEE Press (1992) 286–290.

[34] S. Park and B. Bose, All-to-all broadcasting in faulty hypercubes, *IEEE Trans. Comput.* **46** (1997) 749–755.

[35] D. Peleg, A note on optimal time broadcast in faulty hypercubes, *J. Parallel Dist. Comput.* **26** (1995) 132–135.

[36] C.S. Raghavendra and M.A. Sridhar, Dimension ordering and broadcast algorithms in faulty SIMD hypercubes, *J. Parallel Dist. Comput.* **35** (1996) 57–66.

[37] C.L. Seitz, The Cosmic Cube, *Comm. Assoc. Comput. Mach.* **28** (1985) 22–33.

[38] C.L. Seitz, W.C. Athas, C.M. Flaig, A.J. Martin, J. Scizovic, C.S. Steele and W.-K. Su, Submicron systems architecture project semiannual technical report, California Inst. of Technology Tech. Rep. Caltec-CS-TR-88-18 (1988).

[39] A. Sengupta, On ring embedding in hypercubes with faulty nodes and links, *Inform. Process Lett.* **68** (1998) 207–214.

[40] C. Tsai, J. Tan, T. Liang and L. Hsu, Fault-tolerant hamiltonian laceability of hypercubes, *Inform. Process Lett.* **83** (2002) 301–306.

[41] C. Tsai, Linear array and ring embeddings in conditional faulty hypercubes, *Theoret. Comput. Sci.* **314** (2004) 431–443.

[42] Y. Tseng, Embedding a ring in a hypercube with both faulty links and faulty nodes, *Inform. Process Lett.* **59** (1996) 217–222.

[43] S.-C. Wang, Y.-R. Leu and S.-Y. Kuo, Distributed fault-tolerant embedding of several topologies in hypercubes, *J. Inf. Sci. Eng.* **20** (2004) 707–732.

[44] J. Wu, An optimal fault-tolerant nonredundant broadcasting scheme in injured hypercubes, *J. Parallel Distrib. Comput.* **22** (1994) 295–313.

[45] J. Wu, Optimal broadcasting in hypercubes with link faults using limited global information, *J. Syst. Architect.* **42** (1996) 367–380.

[46] J.-M. Xu, Z.-Z. Du and M. Xu, Edge-fault-tolerant bipancyclicity of hypercubes, *Inform. Process Lett.* **96** (2005) 146-150.