Resource Allocation in Cube Network Systems Based on the Covering Radius

Nian-Feng Tzeng, *Senior Member, IEEE*, and Gui-Liang Feng, *Senior Member, IEEE*

Abstract—When multiple copies of a certain resource exist in a cube network system, it is desirable that every nonresource node can reach the resource in a given number of hops. In this paper, we introduce systematic approaches to resource allocation in a cube system so that each nonresource node is connected with a specified number of resource copies and that the allocation performance measure of interest is optimized. The methodology used is based on the covering radius results of known codes. These codes aid in constructing desired linear codes whose codewords address nodes where resource copies are placed. The resource allocation problem is translated to an integer nonlinear program whose best possible solution can be identified quickly by taking advantage of basic properties derived from the known codes, yielding an optimal or near-optimal allocation result. Those basic properties lead to drastic time complexity reduction (up to several orders of magnitude smaller), in particular for large system sizes. Our approaches are applicable to any cube size, often arriving at more efficient allocation outcomes than what are attainable using prior schemes.

Index Terms—Binary n-cubes, integer nonlinear programs, linear block codes, multiple connection, resource allocation, single connection.

1 Introduction

The cube network has drawn considerable attention recently due to its powerful topological properties [1] and efficient support of various algorithms [2], making it attractive for interconnecting large-scale multiprocessor systems. Successful research and commercial systems based on the cube network have been built, including notably Cosmic Cube [3], Mark III [4], the Intel iPSC/2 [5], the Ncube [6], and the Connection Machine [7], among others. Such a system is also known by the name of hypercube. A three-dimensional hypercube is depicted in Fig. 1.

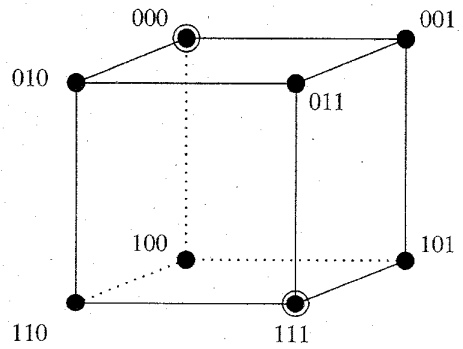


Fig. 1. A three-dimensional cube network, Q_3. (Nodes with circles have one resource copy each.)

• *The authors are with the Center for Advanced Computer Studies, University of Southwestern Louisiana, Lafayette, LA 70504. E-mail: tzeng@cacs.usl.edu.*

Manuscript received Nov. 16, 1993; revised Apr. 13, 1995. A preliminary version of this work was presented at the 22nd International Conference on Parallel Processing in August 1993.
For information on obtaining reprints of this article, please send e-mail to: transactions@computer.org, and reference IEEECS Log Number D95080.

Resources in a cube network system might be shared to reduce the overall system cost, in particular when a resource is expensive or is used by cube nodes infrequently. By resources, we mean hardware devices (like I/O processors, disks, vector units, etc.) or software modules (like library routines, compilers, data files, etc.), which can be attached to or placed at cube nodes. The access of a shared resource tends to involve delay attributed by possible contention with other access requests and by communication latency from a node without such a resource to reach a node with the resource. Multiple copies of each type of resource often exist in a large-scale system to keep the delay of shared resource access reasonably low, and also to ensure high availability of a resource (because the loss of one copy then would not render a type of resource totally unavailable).

The problem of allocating multiple copies of a resource in a cube network system is challenging, and a systematic allocation approach is desirable to achieve best results. There are two possible situations to be considered in the context of resource allocation, depending upon how many resource copies each node is connected with. If a cube node without the resource is in connection with a single copy, called single-connection (as shown in Fig. 1), fewer resource copies are required than multiple-connection, in which every node without the resource is in connection with more than one copy, as exemplified by Fig. 2. On the other hand, multiple-connection gives rise to less contention, possibly yielding better performance, and reduces potential performance degradation after a copy is lost, because every node still can get access to one resource copy in the same number of hops. Most prior resource allocation articles deal with single-connection.

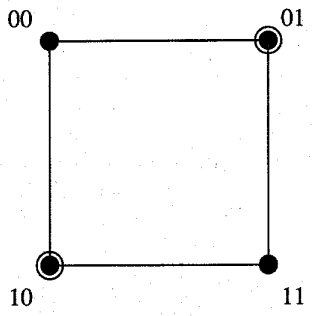


Fig. 2. Each nonresource node in connection with two copies. (Nodes with circles have one resource copy each.)

Distributing resource copies in a hypercube with an attempt to optimize system performance measures of interest has been investigated [8], [9], [10]. Livingston and Stout have studied [8] the minimum number of resource copies needed to meet certain specified requirements when distributing the resource in a hypercube computer. Reddy [9] considered allocating I/O processors (the resource) to cube nodes in a way that every node is adjacent to at least one I/O processor. A perfect allocation results if and only if every cube node is adjacent to exactly one I/O processor. A perfect allocation is shown to exist only for certain cube sizes and is obtained by placing the I/O processors at cube nodes with addresses being the codewords of the Hamming codes, which are known to be perfect codes [11], [12]. An extension to I/O processor allocation in other cube sizes was also presented in [9]. Recently, efficient algorithms have been developed by Chiu and Raghavendra [10] for allocating a given number of resource copies to a hypercube system in an effort to optimize a defined performance measure, called the *resource diameter*. Resource diameter is the maximum resource distance among all the cube nodes, where the resource distance of a node is the minimum number of hops from the node to a node equipped with a copy of the resource. Their algorithms partition hierarchically a system under consideration into levels, in each of which allocation is accomplished following a perfect code (like the Hamming code, the Golay code [11], [12]) or a basic strategy.

More recently, multiple-connection allocation in hypercubes has been proposed on the basis of multiple-adjacency linear block codes [14], [18], which specify the addresses of the nodes involving resource copies. The allocation results are provably optimal if the resource diameter is 1. For the resource diameter greater than 1, the allocation procedure presented is somewhat involved, and the results obtained are not necessarily best possible. Multiple-connection resource allocation in k-ary n-cubes is treated in [17].

In this paper, we deal with allocating copies of a certain resource to cube nodes efficiently in a large system for both single- and multiple-connection situations. Our approaches are based on the covering radius results of known codes to aid in constructing desired linear codes whose codewords address nodes at which resource copies are placed. Our focus is on the cases of the resource diameter greater than 1, since an optimal solution for the case of unit resource diameter has been provided in [14]. The resource allocation problem is translated to an integer nonlinear programming problem whose best solution can be obtained efficiently, giving rise to optimal or near-optimal allocation. It is found from measured data that execution times are shortened drastically by reducing the search space of the integer nonlinear programming problem. Our introduced approaches are applicable to any cube size systematically and often achieve allocation results better than those derived using any previous strategy.

The rest of this paper is organized as follows. In Section 2, necessary notations and useful background are given. Section 3 presents our basic methodology for the resource allocation problem. Section 4 considers allocation under the single-connection situation, i.e., every cube node without a resource is ensured to reach one resource copy in a specified number of hops (resource diameter). Multiple-connection resource allocation is treated in Section 5. Section 6 concludes this paper.

2 Preliminaries

2.1 Notations

A cube network, denoted by Q_n, consists of 2^n nodes, each addressed by a unique n-bit number. A link exists between two nodes of Q_n if and only if their node addresses differ in exactly one bit position, and the two nodes are adjacent. A link is said to be along dimension i if it connects two nodes whose addresses differ in the ith bit (where the least significant bit is referred to as the 0th bit). Q_3 is illustrated in Fig. 1. The *distance* between any two cube nodes is the number of bits differing in their addresses. For example, the distance between nodes (011) and (101) is 2. The number of *hops* (i.e., traversals) needed to reach a node from another node equals the distance between the two nodes.

An allocation of multiple identical copies of a certain resource is the set of cube nodes to which these resource copies are assigned. The resource distance of node X under allocation A, represented by $\zeta_A(X)$, is the minimum number of hops it has to travel to reach a copy of the resource. A small $\zeta_A(X)$ is preferred for node X, as it contributes low traffic to the system, thereby leading to a fast response. An allocation should take the whole network into account, minimizing $\zeta_A(X)$ for all cube nodes Xs. The resource diameter (defined in Section 1) under an allocation A, denoted by d_A, is defined as the maximum $\zeta_A(X)$ among all cube nodes Xs.

As the size of systems in question is a power of 2, in this paper, we limit our consideration to the cases where the number of copies of a resource is an exact power of 2, because full load balance in such a case can be achieved by making each copy support an equal number of cube nodes. Let 2^k resource copies be allocated to a cube network in accordance with allocation A, resulting in the resource diameter denoted by $d_A(n, k)$. When there is no confusion, $d_A(n, k)$ can be expressed by $d(n, k)$ for simplicity. An allocation in Q_n is optimized if either $d(n, k)$ is minimum for a given k, or k is minimum for a given resource diameter.

2.2 Background

Our resource allocation approach is based on the *binary linear block code* (see, for example, [11], [12]), which is briefly reviewed here. A binary linear block code (n, k), denoted by $\Psi(n, k)$, comprises a set of 2^k binary sequences of length n called *codewords*. Code $\Psi(n, k)$ can be described concisely using a k by n matrix $\mathbf{G}$ known as the *generator matrix*. Any codeword of $\Psi(n, k)$ is a linear combination of the rows of $\mathbf{G}$, an associated generator matrix. The linear combination is performed by modulo-2 additions over corresponding bits.

Whether or not an n-tuple $\mathbf{c}$ is a codeword of code $\Psi(n, k)$ can be checked by using an $(n - k)$ by n matrix $\mathbf{H}$, called a *parity-check matrix* of the code. $\mathbf{c}$ is a codeword if and only if it is orthogonal to every row vector of $\mathbf{H}$, namely, $\mathbf{c} \cdot \mathbf{H}^T = \mathbf{0}$, where $\mathbf{H}^T$ is the transpose of matrix $\mathbf{H}$. Notice that the choice of $\mathbf{H}$ is not unique, and any chosen $\mathbf{H}$ satisfies $\mathbf{G} \cdot \mathbf{H}^T = \mathbf{0}$. This equality makes it possible to calculate the corresponding $\mathbf{G}$ (that specifies all the codewords) for a given $\mathbf{H}$, and thus is essential. A linear block code is completely defined as long as its parity check matrix $\mathbf{H}$ is known.

If a parity-check matrix $\mathbf{H}$ for a binary code has α rows, then each column is an α-bit binary number and there are at most $2^\alpha - 1$ nonzero columns. The "largest" parity-check matrix $\mathbf{H}$ obtainable is α by $2^\alpha - 1$. In other words, the columns of this "largest" parity-check matrix $\mathbf{H}$ consist of all the possible nonzero α-tuples (totally, $2^\alpha - 1$ of them). This "largest" parity-check matrix $\mathbf{H}$ is special and, in fact, is the *parity-check matrix of a Hamming code* [11]. The Hamming code is a special linear code $\Psi(n, k)$ such that n equals $2^\alpha - 1$ and k is equal to $2^\alpha - 1 - \alpha$. Linear codes $\Psi(3, 1)$, $\Psi(7, 4)$, $\Psi(15, 11)$, and $\Psi(31, 26)$ are examples of the Hamming codes. A possible parity-check matrix $\mathbf{H}$ of the Hamming code $\Psi(7, 4)$ is

$$\mathbf{H} = \begin{bmatrix} 1001101 \\ 0101011 \\ 0010111 \end{bmatrix}. \tag{1}$$

A basic parameter of linear codes is the covering radius [13]. Consider linear code $\Psi(n, k)$ and any n-tuple $\mathbf{y}$. Let $\Lambda(\mathbf{y})$ be the minimum distance between $\mathbf{y}$ and a codeword in $\Psi(n, k)$, then the *covering radius* of the code is defined as the maximum $\Lambda(\mathbf{y})$ among all $\mathbf{y}$s [15]. As a result, the covering radius of a linear code is exactly the same as the resource diameter of an allocation obtained by following the code. Since our allocation is equivalent to finding out a desired linear code (whose codewords specify the locations of nodes at which resource copies are placed), hereinafter, the terms resource diameter and covering radius will be used interchangeably.

The quality of linear codes can be compared according to the following:

1) Let two codes have the same size of parity-check matrices, i.e., they have the same n and k, then the code with a smaller covering radius is better and clearly gives rise to a more efficient allocation.
2) Let two codes have an identical covering radius but different sizes of parity-check matrices, $(n_1 - k_1)$ by n_1 and $(n_2 - k_2)$ by n_2, respectively, with $n_1 - k_1$ equal to $n_2 - k_2$. If $n_1 < n_2$, then k_1 is less than k_2 and the former code is better, because the ratio of codewords in the former code to all n_1 tuples is less than the ratio of codewords in the latter code to all n_2 tuples due to $k_1/n_1 < k_2/n_2$ (which results directly from $n_1 - k_1 = n_2 - k_2$ and $n_1 < n_2$). In other words, the allocation result derived from the former code is more efficient, as every resource copy then "covers" more nodes. The former code is said to be *shorter* than the latter code for the given covering radius, and a shorter code is preferable.

The allocation problem can be solved elegantly by making use of covering radius results provided in [16], which are given below to facilitate the presentation of our approach. Let codes $\Psi(n_i, k_i)$, $i = 1, 2$, have covering radius r_i and parity check matrix $\mathbf{H}_i$. Then, the direct sum of the two codes is a linear code $\Psi(n_1 + n_2, k_1 + k_2)$ with parity check matrix

$$\begin{bmatrix} \mathbf{H}_1 & \mathbf{O} \\ \mathbf{O} & \mathbf{H}_2 \end{bmatrix}$$

and covering radius $r_1 + r_2$. Multiple linear codes may form another linear code through the direct sum.

Only a few codes have optimal (i.e., smallest possible) covering radii, and are referred to as perfect codes. In particular, Hamming codes $\Psi(2^\alpha - 1, 2^\alpha - 1 - \alpha)$ have the optimal covering radius of 1, and the Golay code (23, 12) has the optimal covering radius of 3 [11]. Hamming codes and the Golay code are the shortest codes for covering radius equal to 1 and 3, respectively. They are the only two types of codes known so far to have optimal covering radii. For any other covering radius, there is no method yet, by which a shortest code can be found. However, near-shortest linear codes with the covering radius of 2 have been introduced [16], where the parameters n and k of the near-shortest linear codes are given by

$$n = 5, \text{ if } n - k = 4; \qquad n = 9, \text{ if } n - k = 5;$$
$$n = 13, \text{ if } n - k = 6; \qquad n = 19, \text{ if } n - k = 7;$$

and for integer $p \geq 2$,

$$\begin{aligned} n &= (2^p - 1)(2^{p+1} + 1), && \text{if } n - k = 4p; \\ n &= (2^p - 1)(2^{p+1} + 1) + (2^{2p} - 1), && \text{if } n - k = 4p + 1; \\ n &= (2^p - 1)(2^{p+1} + 1) + (2^{2p+1} - 1), && \text{if } n - k = 4p + 2; \\ n &= (2^p - 1)(2^{p+1} + 1) + (2^{2p+2} - 1), && \text{if } n - k = 4p + 3. \end{aligned} \tag{2}$$

As an example, the near-shortest linear code with covering radius $r = 2$ for $n - k = 6$ is $\Psi(13, 7)$, as k then equals 7. Similarly, for $p = 2$ and $n - k = 8$, we have the near-shortest linear code $\Psi(27, 19)$, as n is 27 in this situation. The parity check matrix of every near-shortest linear codes with covering radius 2 is specified in [16]. The following gives the parity check matrix of near-shortest code $\Psi(13, 7)$:

$$\mathbf{H} = \begin{bmatrix} 1111111100000 \\ 1111000000000 \\ 0101011010001 \\ 0110001101001 \\ 0110010100101 \\ 0011011000011 \end{bmatrix}$$

3 Proposed Methodology

Our basic methodology for resource allocation is to determine the parity check matrix **H** of an appropriate code whose codewords specify the node addresses at which resource copies are located, in an attempt to optimize the allocation performance measure of interest. For the single-connection situation, two performance measures are of our particular interest: the resource diameter and the number of resource copies.

3.1 Minimizing Resource Diameter

In order to minimize resource diameter $d(n, k)$ for given n and k, we search for an h by l parity check matrix, $\mathbf{H}_o$, which is composed of parity check matrices of Hamming codes, near-shortest codes with covering radius = 2 (called NS codes for simplicity), and the Golay code, such that the resultant covering radius is minimized. The constituent parity check matrices lie along the diagonal of $\mathbf{H}_o$, as depicted in Fig. 3, where the resulting linear code is constituted by the direct sum of codes with parity check matrices $H_1, H_2, ..., H_a$, as explained in Section 2.2. From $\mathbf{H}_o$, we want to trivially get an $(n - k)$ by n matrix, **H**, which fulfills the requirement of $\mathbf{G} \cdot \mathbf{H}^T = \mathbf{0}$, so that the resource addresses can be derived (from **G**). Now, if the size of $\mathbf{H}_o$ obtained satisfies $h \geq n - k$ and $l \leq n$, we may trivially get **H** from $\mathbf{H}_o$ by trimming off its lowest $h - (n - k)$ rows and adding $n - l$ columns of zero vectors to its right end, as shown in Fig. 4. This is obviously true, since $\mathbf{H}_o$ is a parity check matrix, there must be a k by l matrix $\mathbf{G}'$ satisfying $\mathbf{G}' \cdot \mathbf{H}^T = \mathbf{0}$, and **G** can be $\mathbf{G}'$ augmented with $n - l$ columns of arbitrary vectors to its right end.

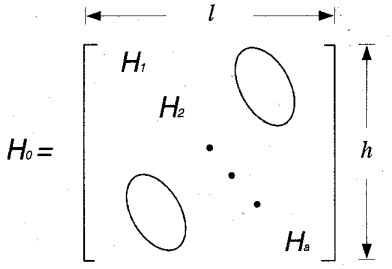


Fig. 3. H_0 formed by $H_1, H_2, ..., H_a$.

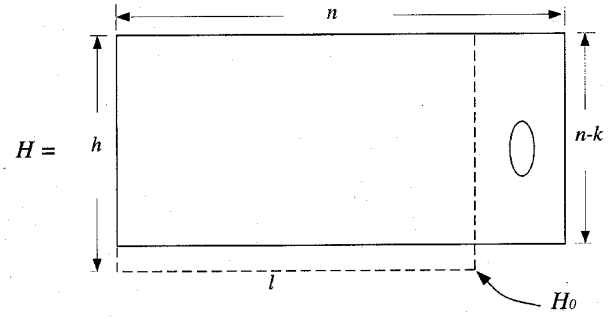


Fig. 4. H constructed from H_0 by trimming its lowest $h - (n - k)$ rows and adding $(n - l)$ columns of zero vectors to its right end.

The constraint on h and l values is essential, making it possible to translate the search of an appropriate code (for allocation) to an integer programming problem. Consider the case of $k = 13$ and $n = 20$ as an example $(n - k = 7)$. There are many different ways to construct the parity check matrix $\mathbf{H}_o$, including the three illustrated in Fig. 5. In Fig. 5a, the parity check matrices of four Hamming codes constitute the resulting matrix, yielding $h = 7$, $l = 12$, and covering radius = 4. The matrix given in Fig. 5b is composed of the parity check matrices of three Hamming codes, exhibiting $h = 8$, $l = 17$, and covering radius = 3. On the other hand, Fig. 5c consists of only the parity check matrix of an NS code. It is apparent that the matrix depicted in Fig. 5c is the best choice, since it leads to a minimum covering radius.

For any set of n and k, the selection of a parity check matrix with the smallest covering radius is achieved by solving the following integer nonlinear program. Let the constituent parity check matrices involve x_α Hamming codes $\Psi(2^\alpha - 1, 2^\alpha - 1 - \alpha)$, y_β NS codes $\Psi(w(\beta), w(\beta) - \beta)$, and z Golay codes, where $w(\beta)$, β, and $\zeta(\beta)$ are, respectively, n, $n - k$, and p given in (2), with $\zeta(\beta) \geq 2$, and x_α, y_β, and z are nonnegative integers. We have

$$\sum_{\forall\alpha} \alpha\, x_\alpha + \sum_{\forall\beta} \beta\, y_\beta + 11z \geq n - k \tag{4}$$

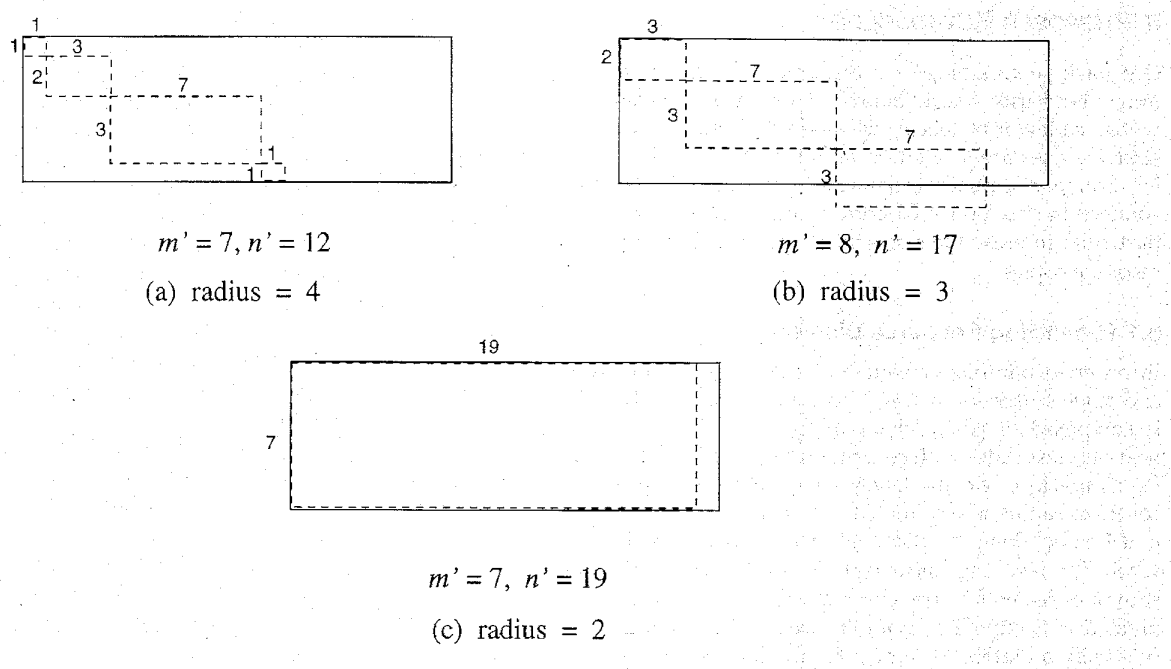
$$\sum_{\forall\alpha} x_\alpha (2^\alpha - 1) + \sum_{\forall\beta} y_\beta w(\beta) + 23z \leq n \tag{5}$$

$$\sum_{\forall\alpha} x_\alpha + \sum_{\forall\beta} 2y_\beta + 3z \ \textit{minimized} \tag{6}$$

Solving the set of preceding equations directly is very time-consuming, as those equations involve integer variables x_α and y_β, for all α, $\beta \geq 0$, in addition to z. Fortunately, we may reduce the complexity of the solution significantly by taking advantage of the basic properties associated with Hamming codes, NS codes, and the Golay code: For any given n, an NS code gives rise to no larger covering radius than the code resulting from a direct sum of multiple Hamming codes, and the Golay code gives rise to no larger covering radius than the code from a direct sum of multiple Hamming codes and NS codes. For $n = 13$, as an example, NS code $\Psi(13, 7)$ has covering radius 2, whereas the code from a direct sum of Hamming codes $\Psi(7, 4)$, $\Psi(3, 1)$, and $\Psi(3, 1)$ has covering radius 3. A solution can be obtained without searching all possibilities of x_α and y_β exhaustively, as will be seen in the next section.

3.2 Minimizing the Number of Resource Copies

For a given resource diameter d, it is interesting to find the minimum number of resource copies required in Q_n. This is equivalent to deriving a linear code $\Psi(n, k)$ for a given covering radius $d(n, k)$ such that it contains as few codewords as possible (i.e., minimum k, as each codeword corresponds to a resource copy). In other words, a suitable parity check matrix of the code with covering radius r is to be constructed in the same way as described above, i.e., making use of parity check matrices of Hamming codes, NS codes, and the Golay code to constitute the resulting parity check



$m' = 7, n' = 12$

(a) radius = 4

$m' = 8, n' = 17$

(b) radius = 3

$m' = 7, n' = 19$

(c) radius = 2

Fig. 5. Three possible ways to form a parity check matrix satisfying $n = 20$ and $k = 13$.

matrix. However, the set of inequalities (or the nonlinear program) to be satisfied becomes

$$\sum_{\forall\alpha} x_\alpha + \sum_{\forall\beta} 2y_\beta + 3z \le r \tag{7}$$

$$\sum_{\forall\alpha} x_\alpha(2^\alpha - 1) + \sum_{\forall\beta} y_\beta w(\beta) + 23z \le n \tag{8}$$

$$f \triangleq \sum_{\forall\alpha} \alpha x_\alpha + \sum_{\forall\beta} \beta y_\beta + 11z \text{ maximized} \tag{9}$$

where nonnegative integers x_α, y_β, and z are as defined before, with $\zeta(\beta) \ge 2$. Note that for a given n, minimizing k means maximizing $n - k$, which is expressed by (9). Again, this minimization problem is translated to an integer nonlinear programming problem. Like the prior one, this integer program has a very high time complexity, and significant reduction in time complexity can result from the basic properties of Hamming codes, NS codes, and the Golay code. The treatment of this minimization problem will be detailed in the next section.

4 Single-Connection Resource Allocation

An allocation with a minimum resource diameter is highly desirable, because it tends to yield lightest traffic over every link and thereby a fastest response time. The first problem under consideration is to find a minimum resource diameter for given n and k, by solving the integer nonlinear program defined by (4)-(6).

4.1 Achieving Minimum Resource Diameter

This integer program can be simplified substantially, according to the next theorem, whose proof is provided in Appendix A.

THEOREM 1. *A solution for* (4)-(6) *involves at most one nonzero* x_α, *say* x_{α^*}, *and four nonzero* y_βs, $\zeta(\beta) \ge 2$. *In addition,* x_{α^*} *is no larger than* 1, *and the nonzero* y_βs *are consecutive, denoted by say* $y_{\beta^*}, y_{\beta^*+1}, y_{\beta^*+2}$, *and* y_{β^*+3}.

Theorem 1 indicates that it is impossible for a solution to involve multiple Hamming code parity check matrices; a solution contains at most one Hamming code parity check matrix. The search process for the solution is thus simplified to

$$\alpha^* x_{\alpha^*} + \sum_{i=0}^{3} (\beta^* + i) y_{\beta^*+i} + 11z \ge n - k \tag{10}$$

$$x_{\alpha^*}(2^{\alpha^*} - 1) + \sum_{i=0}^{3} y_{\beta^*+i} w(\beta^* + i) + 23z \le n \tag{11}$$

such that the covering radius,

$$x_{\alpha^*} + 2\sum_{i=0}^{3} y_{\beta^*+i} + 3z$$

is minimized, where α^* is less than $\log_2(n + 2)$ (from (11) by letting $x_{\alpha^*} = 1$, and the remaining five variables equal to 0). For a set of known x_{α^*} and z, (10)-(11) become, respectively

$$\sum_{i=0}^{3} (\beta^* + i) y_{\beta^*+i} \ge m' \tag{12}$$

$$\sum_{i=0}^{3} y_{\beta^*+i} w(\beta^* + i) \le n' \tag{13}$$

where $m' = n - k - \alpha^* x_{\alpha^*} - 11z$ and $n' = n - x_{\alpha^*}(2^{\alpha^*} - 1) - 23z$, with $n' \ge m' > 0$. The expression to be minimized becomes

$$\sum_{i=0}^{3} y_{\beta^*+i}.$$

A solution for (12)-(13) is reached efficiently by making use of the next lemma and Theorem 2.

LEMMA 1. *Any solution for* (12)-(13): $\beta^*, y_{\beta^*+i}, 0 \le i \le 3$, *satisfies*

$$\frac{w(\beta^*)}{\beta^* + 3} < \frac{n'}{m'}. \tag{14}$$

PROOF. Equation (12) leads to

$$(\beta^* + 3)\sum_{i=0}^{3} y_{\beta^*+i} \le m'.$$

Since $w(\beta^*)$ is a monotonically increasing function, i.e., $w(\beta + 1) > w(\beta)$, it is easy from (13) to get

$$w(\beta^*)\sum_{i=0}^{3} y_{\beta^*+i} \le n'.$$

Equation (14) results directly from the above two newly derived inequalities (note that

$$\frac{w(\beta^*)}{\beta^* + 3}$$

cannot equal

$$\frac{n'}{m'}$$

because they are equal only if $y_{\beta^*+i} = 0$, for all $0 \le i \le 3$, which is not a set of solution for $m' > 0$). □

It should be noted that any solution for (12)-(13) satisfies (14), as proved above; but a β satisfying (14) does not necessarily fulfill (12)-(13). In the next theorem, we refer β only to an integer which fulfills (12)-(13).

THEOREM 2. *Let* β_u *be the largest integer satisfying* (14), *then there is a solution set for the nonlinear program given in* (12)-(13): $\beta^*, y_{\beta^*}, y_{\beta^*+1}, y_{\beta^*+2}, y_{\beta^*+3}$ *such that* β^* *equals* β_u, $\beta_u - 1$, *or* $\beta_u - 2$.

A proof of Theorem 2 can be found in Appendix A. This theorem suggests that only three possible β^* values have to be examined in the process of search for a solution. For each β^* value, a set of $y_{\beta^*}, y_{\beta^*+1}, y_{\beta^*+2}, y_{\beta^*+3}$ is identified and the covering radius calculated. The solution is thus arrived at directly without checking all possible β^* values. An algorithm for obtaining the solution in this efficient way, called Algorithm 1, is provided in Appendix B.

EXAMPLE 1. Consider allocating four copies of a certain resource in Q_8 such that the resource diameter is minimized. Since $n = 8$ and $k = 2$ in this case, we have from Algorithm 1, $f = 3$, $\alpha^* = 2$, $\beta^* = 4$, $x_\alpha^* = 1$, $y_{\beta^*} = 1$, and the remaining output variables = 0, giving rise to parity check matrix

$$\mathbf{H} = \begin{bmatrix} 0\,1\,1 & 1\,0\,0\,0\,0 \\ 1\,0\,1 & 1\,0\,0\,0\,0 \\ - - - & - - - - - \\ 0\,0\,0 & 1\,0\,0\,0\,1 \\ 0\,0\,0 & 1\,0\,1\,0\,0\,1 \\ 0\,0\,0 & 1\,0\,0\,1\,0\,1 \\ 0\,0\,0 & 1\,0\,0\,0\,1\,1 \end{bmatrix}$$

in accordance with (2), where the upper left and the lower right rectangles are, respectively, the parity check matrices of Hamming code $\Psi(3, 1)$ and NS code $\Psi(5, 1)$ [16]. With $\mathbf{H}$, the generator matrix $\mathbf{G}$ is found according to $\mathbf{G} \cdot \mathbf{H}^T = 0$, as given by

$$\mathbf{G} = \begin{bmatrix} 1\,1\,1\,0\,0\,0\,0\,0 \\ 0\,0\,0\,1\,1\,1\,1\,1 \end{bmatrix}.$$

The four resource copies are allocated to the nodes addressed by the linear combinations of the rows of $\mathbf{G}$, i.e., 00000000, 11100000, 00011111, 11111111, achieving the resource diameter = 3, which is minimum.

It is naturally interesting to find out the computation time savings due to the results obtained above for reducing the search space in arriving at a solution set. Algorithm 1 is implemented using C++ and run on a Sun SPARCstation 2 to get solution sets under various n and k values, with measured execution times (in *ms*) listed in Table 1. For comparison, solving (4)-(6) directly without search space reduction is also conducted for the same n and k values, with execution times (in *ms*) recorded and given in brackets. These two ways give exactly identical solution sets for all cases, as expected. Algorithm 1 involves dramatically shorter computation times because its search space is reduced enormously, in particular, for large n values. As an example, for $n = 25$ and $k = 22$, Algorithm 1 has an execution time of 3*ms*, about four orders of magnitude smaller in comparison with that required by solving (4)-(6) directly. More importantly, the execution time of Algorithm 1 remains very small throughout the range of n values under study. This is in sharp contrast to its direct solving counterpart.

TABLE 1
EXECUTION TIME (IN *ms*) UNDER VARIOUS n AND k VALUES

k	$n=10$	$n=15$	$n=20$	$n=25$	$n=30$
$n-7$	7.9 [16.1]	12.5 [437]	5.1 [10,100]	5.9 [34,289]	8.6 [717,448]
$n-3$	3.4 [16.1]	3.5 [437]	2.9 [10,281]	3.0 [35,010]	6.2 [734,121]

(Times for solving (4)-(6) directly without search space reduction are given in brackets.)

4.2 Achieving Fewest Resource Copies

Finding the minimum number of copies needed to achieve a given resource diameter in Q_n is natural, since this number tells the lowest cost possible. It is translated to an integer nonlinear program specified by (7)-(9), whose search space is reduced drastically according to the subsequent theorem. This theorem can be proved in exactly the same way as that for Theorem 1, as explained in Appendix A.

THEOREM 3. *A solution for (7)-(9) involves at most one nonzero x_α and four nonzero y_βs, $\zeta(\beta) \geq 2$. Nonzero x_α is no larger than 1, and the nonzero y_βs are consecutive, denoted by say $y_{\beta^*}, y_{\beta^*+1}, y_{\beta^*+2}$, and y_{β^*+3}.*

Let the nonzero x_α, if any, be denoted by x_{α^*}. As a result of Theorem 3, the nonlinear program under consideration is simplified to

$$x_{\alpha^*} + 2\sum_{i=0}^{3} y_{\beta^*+i} + 3z \leq r \tag{15}$$

$$x_{\alpha^*}(2^{\alpha^*} - 1) + \sum_{i=0}^{3} y_{\beta^*+i}\ w(\beta^* + i) + 23z \leq n \tag{16}$$

such that

$$m \triangleq \alpha^* x_{\alpha^*} + \sum_{i=0}^{3} (\beta^* + i) y_{\beta^*+i} + 11z$$

is maximized, with $\alpha^* \leq \log_2 (n + 2)$. For given x_{α^*} and z, (15)-(16) become, respectively,

$$\sum_{i=0}^{3} y_{\beta^*+i} \leq r' \tag{17}$$

$$\sum_{i=0}^{3} y_{\beta^*+i}\ w(\beta^* + i) \leq n' \tag{18}$$

where $r' = (r - x_{\alpha^*} - 3z)/2$, n' is as defined in (13), $n' \geq r' > 0$. The expression to be maximized is

$$m' = \sum_{i=0}^{3} (\beta^* + i) y_{\beta^*+i}.$$

We may identify the search space of the integer nonlinear program specified by (17)-(18) in two different cases, depending on the value of

$$\sum_{i=0}^{3} y_{\beta^*+i}.$$

Case 1. $\sum_{i=0}^{3} y_{\beta^*+i} = r'$.

In this case, we have

$$w(\beta^*) r' = w(\beta^*) \sum_{i=0}^{3} y_{\beta^*+i} < \sum_{i=0}^{3} w(\beta^* + i) y_{\beta^*+i} \leq n',$$

leading to

$$w(\beta^*) < \frac{n'}{r'}$$

(recall that $w(\beta)$ is a monotonically increasing function). Let β_u be the largest integer value satisfying

$$w(\beta^*) < \frac{n'}{r'},$$

then the β^* value to be examined ranges from 2 to β_u, inclusively. For each β^* (an integer) examined, the above nonlinear program becomes a linear program and a solution set can be found, with m' obtained. The solution under this case is the one which gives rise to the maximum m'.

Case 2. $\sum_{i=0}^{3} y_{\beta^*+i} < r'$.

In this case, let

$$\sum_{i=0}^{3} y_{\beta^*+i} = r' - \delta,$$

with $1 \leq \delta \leq r' - 1$. If $\beta^*, y_{\beta^*}, y_{\beta^*+1}, y_{\beta^*+2}, y_{\beta^*+3}$, are a set of solution for the above nonlinear program, we have the following inequality (see Appendix A for a proof).

$$\sum_{i=0}^{3} y_{\beta^*+i}\ w(\beta^* + i) > n' - \delta\ w(\beta^*) \tag{19}$$

Rearranging (19) arrives at

$$n' < (y_{\beta^*} + \delta) w(\beta^*) + \sum_{i=1}^{3} y_{\beta^*+i}\ w(\beta^* + i)$$

which is less than

$$(\delta + \sum_{i=0}^{3} y_{\beta^*+i}) w(\beta^* + 3) = r'\ w(\beta^* + 3).$$

This results in

$$w(\beta^* + 3) > \frac{n'}{r'}. \tag{20}$$

On the other hand, from

$$\sum_{i=0}^{3} y_{\beta^*+i} > 1$$

(i.e., at least one of the three terms is nonzero), we have

$$w(\beta^*) \leq \sum_{i=0}^{3} y_{\beta^*+i}\ w(\beta^* + i),$$

which is $\leq n'$, according to (18). Let β_s and β_u be, respectively, the minimum and the maximum values which satisfy both (20) and $w(\beta^*) \leq n'$. The β^* value to be searched ranges from $\max(\beta_s, 2)$ to β_u, where max is a maximum function. For each value searched, (17)-(18) become a linear program and a solution is then obtained. The solution under Case 2 is the one which yields maximum m'. We, thus, solve this problem based on the Case 1 solution and the Case 2 solution. An algorithm for arriving at the solution using these results, called Algorithm 2, is listed in Appendix B.

EXAMPLE 2. Suppose that an allocation with resource diameter $r = 3$ is to be determined in Q_{20} using fewest resource copies. From Algorithm 2, we have output variables $m = 9$, $\alpha^* = 3$, $\beta^* = 6$, $x_{\alpha^*} = 1$, $y_{\beta^*} = 1$, and the rest being 0, resulting in a parity check matrix

TABLE 2
EXECUTION TIME (IN *ms*) UNDER VARIOUS n AND r VALUES

r	$n=10$	$n=15$	$n=20$	$n=25$	$n=30$
2	3.3 [495]	6.1 [15,358]	15.5 [187,840]	23.2 [1,290,819]	45.2 [?]
3	5.2 [1,168]	13.9 [15,941]	759 [190,752]	792 [1,296,747]	857 [?]
5	5.5 [1,172]	14.7 [15,968]	767 [194,365]	800 [1,316,872]	866 [?]

(Times for solving (7)-(9) directly without search space reduction are given in brackets, and ? indicates a very large execution time.)

$$\begin{bmatrix} \mathbf{H}_1 & \mathbf{O} \\ \mathbf{O} & \mathbf{H}_2 \end{bmatrix},$$

where $\mathbf{H}_1$ is the parity check matrix of Hamming code $\Psi(7, 4)$ given by (1) and $\mathbf{H}_2$ is the parity check matrix of NS code $\Psi(13, 7)$ expressed by (3). The generator matrix corresponding to the above 9 by 20 parity check matrix specifies the addresses of resource nodes, which amount to 2^{11} in total. If the strategies given in [10] are followed, one can show that the best result achievable for $r = 3$ in Q_{20} requires 2^{12} resource copies (an example such allocation is done by using Hamming code $\Psi(15, 11)$ plus the basic strategy for Q_5 with resource diameter 2 (= 3 − 1)), two times as many copies as in our allocation.

Algorithm 2 is implemented using C++ to check its execution efficiency in comparison with solving (7)-(9) directly without search space reduction. Execution times (in *ms* on a Sun SPARCstation 2) needed for different n and r values are presented in Table 2, where those included in brackets are times required if solution sets are obtained from (7)-(9) directly. Algorithm 2 arrives at the same solution set as its direct solving counterpart for any given n and r, but involves a substantially shorter execution time. It is possible to attack larger n cases without increasing execution times much following Algorithm 2. Conversely, exhaustive search makes the execution time grow rapidly as n increases.

5 MULTIPLE-CONNECTION RESOURCE ALLOCATION

The preceding section deals with the situation where a nonresource node is connected with one copy of the resource. Under this situation, excessive sharing of one resource copy may occur in a large system because every copy has to support an increasing number of nodes when the system dimension is enlarged, degrading overall system performance due to growing contention. Consequently, it might be desirable for a large system to consider the allocation of a resource in a way that every nonresource node is in connection with multiple resource copies. Such a resource allocation is fault-tolerant in the sense that the resource diameter of the allocation remains unchanged even after a fault arises. This kind of resource allocation is the focus of this section.

The problem to be dealt with here concerns efficient approaches to multiple-connection resource allocation. Optimal or near-optimal solutions which achieve a specified degree of connection (for any node without the resource) j under a resource diameter r, are to be found. They are based on the result obtained in Section 4.2 for given n' and r' so as to minimize the number of resource copies involved. The subsequent cases are treated separately to arrive at desired solutions.

Case 1. $j \le 2^{r-1}$

In this case, we focus on the situation of $r > 1$, as our concern lies in multiple-connection allocation, i.e., $j > 1$. Suppose that $\tau = \lceil \log_2 j \rceil$ and $\mathbf{H}'$ is a parity check matrix of linear code $\Psi(n', k')$ with covering radius r', where $n' = n - \tau$, $k' = k - \tau$ (to be decided), and $r' = r - \tau$ (which is ≥ 1). Matrix $\mathbf{H}'$ is obtained using the method described in Section 4.2. Then, the parity check matrix characterizing j-connection resource allocation with resource diameter r and minimized k for a given n is specified by

$$\mathbf{H}_1 = [\mathbf{0}\ \mathbf{0} \cdots \mathbf{0}\ \mathbf{0}\ \mathbf{H}']$$

where there are τ zero vectors $\mathbf{0}$ of size $(n - k)$ by 1 (as $n' - k' = n - k$). Since H' is an $(n' - k')$ by n' matrix, $\mathbf{H}_1$ is an $(n - k)$ by n matrix.

What remains to be proved is that any noncodeword with respect to $\mathbf{H}_1$ has at least j codewords within distance r or less, as stated in the following theorem.

THEOREM 4. *Parity check matrix $\mathbf{H}_1$ specifies a j-connection resource allocation with resource diameter r, $j \le 2^{r-1}$, in Q_n involving 2^k resource copies.*

PROOF. First, consider linear code $\Psi(n', k')$ and any of its noncodeword $\mathbf{a}' = (a_{n-\tau-1}a_{n-\tau-2} \ldots a_1a_0)$. Since $\mathbf{H}'$ is a parity check matrix of linear code $\Psi(n', k')$ with covering radius r', there is one codeword $\mathbf{c}' = (c_{n-\tau-1}c_{n-\tau-2} \ldots c_1c_0)$ such that the Hamming distance between $\mathbf{c}'$ and $\mathbf{a}'$ is no more than $r' = r - \tau$.

Next, consider any noncodeword, $\mathbf{a} = (a_{n-1}a_{n-2} \ldots a_{n-\tau}\ \mathbf{a}')$, and a codeword, $\mathbf{c} = (c_{n-1}c_{n-2} \ldots c_{n-\tau}\ \mathbf{c}')$, with respect to parity check matrix $\mathbf{H}_1$, where $\mathbf{a}'$ and $\mathbf{c}'$ are, respectively, the noncodeword and the codeword regarding $\Psi(n', k')$. Let $\mathbf{e}$ denote $(e_{n-1}e_{n-2} \ldots e_{n-\tau}\,00 \ldots 0)$, where e_h, $n - 1 \ge h \ge n - \tau$, can be 0 or 1; in other words, $\mathbf{e}$ is any one of the 2^τ distinct vectors, depending on the values of e_hs. Clearly, for any $\mathbf{e}$, $(\mathbf{c} + \mathbf{e})$ is a codeword

with respect to parity check matrix $\mathbf{H}_1$, and the Hamming distance between $(\mathbf{c} + \mathbf{e})$ and noncodeword $\mathbf{a}$ is no more than $\tau + r' = r$, as the distance between $\mathbf{c}'$ and $\mathbf{a}'$ is r' or less. Consequently, noncodeword $\mathbf{a}$ is connected with 2^{τ} $(\geq j)$ codewords whose distances are within r. □

EXAMPLE 3. Consider the case of $j = 4$ and $r = 3$ in Q_9. This case satisfies $j \leq 2^{r-1}$ and we have $\tau = 2$, $n' = n - \tau = 2$, $r' = r - \tau = 1$, indicating that a linear block code of length 7 with covering radius 1 should be found first. Hamming code $\Psi(7, 4)$ is such a code, whose parity check matrix is given in (1). The 4–connection resource allocation result is thus specified by a 3 by 9 parity check matrix:

$$\mathbf{H}_1 = \begin{bmatrix} 0\,0\,1\,0\,0\,1\,1\,0\,1 \\ 0\,0\,0\,1\,0\,1\,0\,1\,1 \\ 0\,0\,0\,0\,1\,0\,1\,1\,1 \end{bmatrix}.$$

Since Hamming code is an optimal code, the allocation so obtained is optimal, requiring $2^{9-3} = 64$ resource copies in total.

Case 2. $j > 2^{r-1}$ *and* $r > 1$

In this case, we cannot define τ as above because covering radius $r - \tau$ is then no longer ≥ 1, precluding the existence of any linear code $\Psi(n - \tau, k - \tau)$. Instead, let σ_χ be the smallest integer satisfying

$$\sum_{v=0}^{r-\chi} \binom{\sigma_\chi}{v} \geq j,$$

where

$$\binom{\sigma_\chi}{v}$$

equals

$$\frac{\sigma_\chi!}{v!\,(\sigma_\chi - v)!},$$

the total combinations of v out of σ_χ items, and $1 \leq \chi \leq r - 1$. Assume that $\mathbf{H}^*_\chi$ is a parity check matrix of linear code $\Psi(n - \sigma_\chi, k - \sigma_\chi)$ with covering radius χ derived using the procedure presented in Section 4.2. In other words, there is one $\mathbf{H}^*_\chi$ produced for each χ, $1 \leq \chi \leq r - 1$, when $n - \sigma_\chi$ and covering radius χ are given as inputs to Algorithm 2. A resource allocation with j-connection and resource diameter r in this case can be achieved according to a linear code whose parity check matrix is

$$\mathbf{H}_2 = [\mathbf{0}\ \mathbf{0} \cdots \mathbf{0}\ \mathbf{H}^*_\chi],$$

where there are σ_χ zero vectors $\mathbf{0}$ of size $(n - k)$ by 1. The next theorem expresses the correctness of using parity check matrix $\mathbf{H}_2$ in this case.

THEOREM 5. *Parity check matrix* $\mathbf{H}_2$ *specifies a j-connection resource allocation with resource diameter* r, $j > 2^{r-1}$ $(r > 1)$, *in* Q_n *involving* 2^k *resource copies.*

PROOF. Assume that $\mathbf{a}^* = (a_{n-\sigma_\chi-1} a_{n-\sigma_\chi-2} \cdots a_1 a_0)$ is not a codeword in $\Psi(n - \sigma_\chi, k - \sigma_\chi)$. There must be one codeword $\mathbf{c}^* = (c_{n-\sigma_\chi-1} c_{n-\sigma_\chi-2} \cdots c_1 c_0)$ such that the Hamming distance between $\mathbf{c}^*$ and $\mathbf{a}^*$ is χ, for the covering radius of $\Psi(n - \sigma_\chi, k - \sigma_\chi) = \chi$.

Next, consider any noncodeword,

$$\mathbf{a} = (a_{n-1} a_{n-2} \cdots a_{n-\sigma_\chi}\ \ \mathbf{a}^*)$$

and a codeword,

$$\mathbf{c} = (c_{n-1} c_{n-2} \cdots c_{n-\sigma_\chi}\ \ \mathbf{c}^*)$$

with respect to parity check matrix $\mathbf{H}_2$, where $\mathbf{a}^*$ and $\mathbf{c}^*$ are, respectively, the noncodeword and the codeword regarding $\Psi(n - \sigma_\chi, k - \sigma_\chi)$. Let $\mathbf{e}$ denote

$$(e_{n-1} e_{n-2} \cdots e_{n-\sigma_\chi}\ 0\ 0 \cdots 0),$$

where e_h, $n - 1 \geq h \geq n - \sigma_\chi$, can be 0 or 1 such that the total number of bits "1" is no more than $r - \chi$. In other words, $\mathbf{e}$ is any one of the

$$\sum_{v=0}^{r-\chi} \binom{\sigma_\chi}{v}$$

distinct vectors, relying on the values of e_hs. Suppose that

$$\mathbf{e}^+ = (a_{n-1} a_{n-2} \cdots a_{n-\sigma_\chi}\ 0\ 0 \cdots 0).$$

For any $\mathbf{e}$, it is clear that $(\mathbf{c} + \mathbf{e}^+ + \mathbf{e})$ is a codeword with respect to parity check matrix $\mathbf{H}_2$, and the Hamming distance between $(\mathbf{c} + \mathbf{e}^+ + \mathbf{e})$ and noncodeword $\mathbf{a}$ is no more than $(r - \chi) + \chi = r$, as the distance between

$$a_{n-1} a_{n-2} \cdots a_{n-\sigma_\chi}$$

and the leftmost σ_χ bits of $(\mathbf{c} + \mathbf{e}^+ + \mathbf{e})$ is less than or equal to $r - \chi$. Since there are

$$\sum_{v=0}^{r-\chi} \binom{\sigma_\chi}{v} \geq j$$

distinct $\mathbf{e}$ vectors, noncodeword $\mathbf{a}$ is connected with at least j codewords whose distances are within r. □

Every χ value results in one allocation specified by $\mathbf{H}_2$ which is constructed according to Algorithm 2, and such an allocation always meets the requirements of j and r, as indicated by Theorem 5. Our solution is the most efficient allocation involving fewest resource copies, selected from the $r - 1$ allocations, one corresponding to a χ value.

EXAMPLE 4. Assume that an allocation with $j = 6$ and $r = 3$ is to be realized in Q_{16} using as few resource copies as possible. Since this allocation satisfies $j > 2^{r-1}$ and $r > 1$, we get $\sigma_1 = 3$ when χ is 1 and $\sigma_2 = 5$ when χ is 2. For $\chi = 1$, the two inputs to Algorithm 2 are $n = 16 - \sigma_1$ and $r = 1$, yielding

$$\mathbf{H}^*_\chi = [\mathbf{0}\,\mathbf{0}\,\mathbf{0}\,\mathbf{0}\,\mathbf{0}\,\mathbf{0}\,\mathbf{H}^\dagger]$$

where $\mathbf{H}^\dagger$ is the parity check matrix of Hamming code $\Psi(7, 4)$ given in (1). On the other hand, the two inputs

to Algorithm 2 for $\chi = 2$ are $n = 16 - \sigma_2$ and $r = 2$, leading to

$$\mathbf{H}^*_\chi = [0\,0\,\mathbf{H}^\ddagger]$$

where $\mathbf{H}^\ddagger$ is the parity check matrix of NS code $\Psi(9, 4)$ [16]. Consequently, the most efficient allocation results from $\chi = 2$, taking $2^{16-(9-4)} = 2^{11}$ total resource copies. This same example was attempted in [14] by a hierarchical scheme, requiring a total of 2^{12} copies.

Case 3. $j > 1$ and $r = 1$

This case of multiple-connection allocation requires the resource diameter equal to 1. It becomes a multiple-adjacency problem treated in [14], where the parity check matrix of linear block codes for characterizing such placements is provided.

6 Conclusions

Systematic resource allocation in cube network systems that optimizes the measure of interest has been presented. This is made possible by constructing appropriately a parity check matrix of the desired linear code whose codewords specify the locations of resource copies, using the results of three types of known codes: Hamming codes, near-shortest codes with covering radius = 2, and the Golay code. Finding the best allocation with respect to a given measure of interest is translated to selecting a set of the codes whose covering radii satisfy two nonlinear inequalities and optimize the third one. Significant reduction in the time complexity of selecting such a set is attained by taking advantage of the basic properties of the three types of codes. For single-connection resource allocation, we have pursued the approach to achieving minimum resource diameter in a cube system involving a given number of resource copies as well as the approach to achieving fewest resource copies for a given resource diameter. Collected data show that execution times can be shortened by orders of magnitude due to search space reduction made by utilizing basic code properties. For multiple-connection resource allocation, we have derived optimal or near-optimal solutions which satisfy a given degree of connection for any nonresource node under a specified resource diameter. The allocation results we obtained appear interesting and are particularly useful for large-scale cube network systems.

Appendix A

The proof of Theorem 1 relies on Lemmas 2-5 below, which characterize how a solution can be refined for a given covering radius

$$f \triangleq \sum_{\forall\alpha} \chi_\alpha + \sum_{\forall\beta} 2y_\beta + 3z$$

expressed by (6). A set of solution, $\tilde{\Theta}$: $\tilde{x}_\alpha, \tilde{y}_\beta$, and z, for (4)-(5) is said to be better than another set of solution Θ: x_α, y_β, and z, for a given f, if

$$\sum_{\forall\alpha} \alpha\,\tilde{x}_\alpha + \sum_{\forall\beta} \beta\,\tilde{y}_\beta + 11z \geq \sum_{\forall\alpha} \alpha\,x_\alpha + \sum_{\forall\beta} \beta\ y_\beta + 11z \quad (21)$$

$$\sum_{\forall\alpha} \tilde{x}_\alpha(2^\alpha - 1) + \sum_{\forall\beta} \tilde{y}_\beta\ w(\beta) + 23z < \sum_{\forall\alpha} x_\alpha(2^\alpha - 1) + \sum_{\forall\beta} y_\beta\ w(\beta) + 23z\,. \quad (22)$$

The preceding inequalities serve as the basis of our refinement process.

LEMMA 2. *Let Θ be a solution for (4)-(5), with f minimized. For any $x_s \geq 2$, we have a refined solution $\tilde{\Theta}$: $\tilde{x}_s = x_s - 2$; $\tilde{y}_{2s} = y_{2s} + 1$; $\tilde{x}_\alpha = x_\alpha$, for all $\alpha \neq s$; and $\tilde{y}_\beta = y_\beta$, for all $\beta \neq 2s$. The refined solution keeps f unchanged.*

PROOF. It is trivial to show that solutions $\tilde{\Theta}$ and Θ have the same f. In order to check whether or not (21)-(22) hold for the two solutions, we consider an even s and an odd s separately.

(i) $s = 2k$: In this case,

$$\Delta \triangleq \left(\sum_{\forall\alpha} \alpha\,\tilde{x}_\alpha - \sum_{\forall\alpha} \alpha\,x_\alpha\right) + \left(\sum_{\forall\beta} \beta\,\tilde{y}_\beta - \sum_{\forall\beta} \beta\,y_\beta\right) = -2s + 2s = 0\,,$$

implying that (21) is satisfied. According to (2), $w(2s)$ equals $(2^k - 1)(2^{k+1} + 1) = 2^{2k+1} - 2^k - 1 < 2(2^s - 1)$, i.e, $w(2s) < 2(2^s - 1)$. Since

$$\Gamma \triangleq \left(\sum_{\forall\alpha} \tilde{x}_\alpha\left(2^\alpha - 1\right) - \sum_{\forall\alpha} x_\alpha\left(2^\alpha - 1\right)\right) + \left(\sum_{\forall\beta} \tilde{y}_\beta w(\beta) - \sum_{\forall\beta} y_\beta\ w(\beta)\right) = -2(2^s - 1) + w(2s)$$

which is less than 0, (22) holds.

(ii) $s = 2k + 1$: In this case, we may prove in the same way as above that $\Delta \geq 0$ and (21) is valid. From (2), one can similarly derive $w(2s) < 2(2^s - 1)$, which indicates that $\Gamma < 0$ and thus (22) is true. This completes the proof. □

LEMMA 3. *Let Θ be a solution for (4)-(5), with f minimized. For any x_s, $x_{s+1} \geq 1$, we have a refined solution $\tilde{\Theta}$: $\tilde{x}_s = x_s - 1$; $\tilde{x}_{s+1} = x_{s+1} - 1$; $\tilde{y}_{2s+1} = y_{2s+1} + 1$; $\tilde{x}_\alpha = x_\alpha$, for all $\alpha \neq s$ or $s + 1$; and $\tilde{y}_\beta = y_\beta$, for all $\beta \neq 2s + 1$. The refined solution keeps f unchanged.*

PROOF. It is clear that solutions $\tilde{\Theta}$ and Θ have the same f. Fol-

lowing the same way as given in the proof of Lemma 2, one can easily arrive at Δ (defined above) ≥ 0 and thereby (21). Next, if we prove $w(2s+1) < (2^{s+1}-1) + (2^s - 1)$, (22) follows immediately because Γ (defined above) is then < 0. Let $s = 2k$ first. From (2), $w(2s+1)$ is $(2^k - 1)(2^{k+1} + 1) + (2^{2k} - 1)$, which equals $(2^{2k+1} - 2^k - 1) + (2^{2k} - 1) < 2^{2k+1} - 1 + 2^{2k} - 1 = 2^{s+1} - 1 + 2^s - 1$. When $s = 2k+1$, the inequality of $w(2s+1) < (2^{s+1}-1) + (2^s - 1)$ can likewise be obtained. As a result, (22) is valid for any s. □

LEMMA 4. *Let Θ be a solution for (4)-(5), with f minimized. For any x_s, $x_{s+h} \geq 1$, and $h \geq 2$, we have a refined solution $\tilde{\Theta}$: $\tilde{x}_s = x_s - 1$; $\tilde{x}_{s+h} = x_{s+h} - 1$; $\tilde{x}_{s+l} = x_{s+l} + 2$, for any*

$$\left\lceil \frac{h}{2} \right\rceil \leq l < h;\ \tilde{x}_\alpha = x_\alpha,$$

for all $\alpha \neq s, s+l, s+h$; and $\tilde{y}_\beta = y_\beta$, for all β. The refined solution keeps f unchanged.

PROOF. It is obvious that both solutions have the same f. To prove (21), we examine Δ, which is equal to

$$-s - (s+h) + 2(s+l) = -h + 2l \geq 0 \left(\text{as} \left\lceil \frac{h}{2} \right\rceil \leq l \right).$$

Consequently, we have (21). Since $2(2^{s+l} - 1) = 2^s(2^{l+1}) - 2 < 2^s(2^h + 1) - 2$ (for $l < h$), which equals $(2^s - 1) + (2^{s+h} - 1)$, (22) follows. □

LEMMA 5. *Let Θ be a solution for (4)-(5), with f minimized. For any y_s, $y_{s+h} \geq 1$, and $h \geq 4$, we have a refined solution $\tilde{\Theta}$: $\tilde{y}_s = y_s - 1$; $\tilde{y}_{s+h} = y_{s+h} - 1$; $\tilde{y}_{s+l} = y_{s+l} + 2$ for any*

$$l = \left\lceil \frac{h}{2} \right\rceil;\ \tilde{y}_\beta = y_\beta$$

for all $\beta \neq s, s+1, s+h$; and $\tilde{x}_\alpha = x_\alpha$, for all α. The refined solution keeps f unchanged.

PROOF. It is trivial to see that both solutions have the same f. We may arrive at (21) due to $\Delta \geq 0$, based on a similar examination as given in the proof of Lemma 4. Next, the inequality of $2w(s) < w(s+2)$ is to be shown. For $s = 4k+2$, we have $2w(s) = 2w(4k+2) = 2(2^{2k+1} - 2^k - 1 + 2^{2k+1} - 1)$, according to (2). The preceding expression is less than $2^{2k+3} - 2^{k+1} - 1$, which equals $w(4k+4) = w(s+2)$. The same result of $2w(s) < w(s+2)$ can be obtained for $s = 4k$, $4k+1$, and $4k+3$. From the above result, $2w(s+l) < w(s+l+2)$, which is $\leq w(s+h)$, for

$$h \geq 4,\ l = \left\lceil \frac{h}{2} \right\rceil \geq 2$$

and $w(s)$ is a monotonically increasing function. This means that $2w(s+1) < w(s+h) < w(s+h) + w(s)$, implying $\Gamma < 0$, which leads to (22) directly. □

PROOF OF THEOREM 1. This theorem is proved by applying the results given in Lemmas 2-5 repeatedly, as follows. A given solution is refined using Lemma 2 repeatedly until $x_s = 0$, or 1, for all s. Lemma 3 is then applied repeatedly to make every pair of $x_s = x_{s+1} = 1$ become 0, yielding a solution in which there is no consecutive x_ss being nonzero. The next step is to have every two nonzero x_s and x_{s+h}, with $h \geq 2$, decremented by 1 each, and x_{s+l} incremented by 1 for any

$$\left\lceil \frac{h}{2} \right\rceil \leq l < h,$$

according to Lemma 4. In case any x_{s+l} becomes 2 in the course of the above step, Lemma 2 is employed to make it become zero. At the end of this step, there is at most one nonzero x_s left.

From Lemma 5, any nonzero y_s and $y_{s+h} \geq 1$, with $h \geq 4$, can be refined by decrementing them by 1 each and by incrementing y_{s+l} by 2, for any

$$l = \left\lceil \frac{h}{2} \right\rceil.$$

Repeated application of Lemma 5 results in that no more four y_βs are nonzero and they are consecutive, as described in the theorem. □

PROOF OF THEOREM 2. We first prove that

$$\Xi(\beta) \triangleq \frac{w(\beta)}{\beta + 3}$$

is a strictly increasing function. Let $\beta = 4k+3$, then $(\beta+3)w(\beta+1) - (\beta+4)w(\beta) = (\beta+3)(w(\beta+1) - w(\beta)) - w(\beta) = (4k+6)((2^{k+1} - 1)(2^{k+2} + 1) - (2^k - 1)(2^{k+1} + 1) - (2^{2k+2} - 1)) - (2^k - 1)(2^{k+1} + 1) - (2^{2k+2} - 1)$, from (2). This expression equals

$$\begin{aligned}
&(4k+6)(2^{2k+3} - 2^{k+1} - 1 - 2^{2k+1} + 2^k + 1 - 2^{2k+2} + 1) - \\
&(2^{2k+1} - 2^k - 1) - (2^{2k+2} - 1) \\
&= (4k+6)(2^{2k+1} - 2^k + 1) - (2^{2k+1} - 2^k - 1) - (2^{2k+2} - 1) > \\
&(4k+5)(2^{2k+1} - 2^k + 1) - (2^{2k+2} - 1) > \\
&(4k+5)(2^{2k+1} - 2^k + 1) - 2(2^{2k+1} - 2^k + 1) \\
&= (4k+3)(2^{2k+1} - 2^k + 1) > 0.
\end{aligned}$$

This means that

$$\frac{w(\beta)}{\beta+3} < \frac{w(\beta+1)}{\beta+4},$$

or equivalently, $\Xi(\beta)$ is a strictly increasing function. Along the same line, it can be easily proved that $\Xi(\beta)$ is a strictly increasing function for the cases of $\beta = 4k$, $4k+1$, and $4k+2$.

Now that $\Xi(\beta)$ is a strictly increasing function and β_u is the largest integer satisfying (14), no β value greater than β_u needs to be considered in the search of a solution. β_u fulfills (12)-(13). For this β_u, (12)-(13) become

an integer *linear program*, called *LP*, and suppose that $\{y_{\beta_u+i} | 0 \le i \le 3\}$ (which involves elements y_{β_u}, y_{β_u+1}, y_{β_u+2}, y_{β_u+3}) is a set of solution for *LP*. We then show by contradiction the subsequent inequality regarding this solution set.

$$\sum_{i=0}^{3}(\beta_u+i)y_{\beta_u+i} \le \begin{cases} m'+\beta_u+2, & \text{if } y_{\beta_u}=y_{\beta_u+1}=y_{\beta_u+2}=0,\ y_{\beta_u+3}\neq 0 \\ m'+\beta_u+1, & \text{if } y_{\beta_u}=y_{\beta_u+1}=0, y_{\beta_u+2}\neq 0 \\ m'+\beta_u, & \text{if } y_{\beta_u}=0, y_{\beta_u+1}\neq 0 \\ m'+\beta_u-1, & \text{if } y_{\beta_u}\neq 0 \end{cases} \quad (23)$$

Let us examine the case of $y_{\beta u}=y_{\beta u+1}=y_{\beta u+2}=0, y_{\beta u+3} \neq 0$ first. Assume

$$\sum_{i=0}^{3}(\beta_u+i)y_{\beta_u+i} > m'+\beta_u+2,$$

and consider the set:

$$\tilde{y}_{\beta_u+3}=y_{\beta_u+3}-1, \tilde{y}_{\beta_u}=\tilde{y}_{\beta_u+1}=\tilde{y}_{\beta_u+2}=0, \tilde{\beta}_u=\beta_u.$$

The above set is then a refined solution because it satisfies (12)-(13), as below

$$\sum_{i=0}^{3}(\beta_u+i)\,\tilde{y}_{\beta_u+i}=(\beta_u+3)\,\tilde{y}_{\beta_u+3}=(\beta_u+3)\left(y_{\beta_u+3}-1\right)$$

which $> (m'+\beta_u+2)-(\beta_u+3) \ge m'$ due to our assumption, and

$$\sum_{i=0}^{3}(\tilde{\beta}_u+i)\,w(\tilde{\beta}+i) < \sum_{i=0}^{3}(\beta_u+i)\,w(\beta+i) \le n'.$$

However, the refined solution set yields covering radius

$$\sum_{i=0}^{3}\tilde{y}_{\beta_u+i} < \sum_{i=0}^{3}y_{\beta_u+i},$$

suggesting that $\{y_{\beta_u+i} \mid 0 \le i \le 3\}$ is not a solution set for *LP*, a contradiction, which results from our earlier assumption. This means that

$$\sum_{i=0}^{3}(\beta_u+i)y_{\beta_u+i} \le m'+\beta_u+2,$$

if

$$y_{\beta_u}=y_{\beta_u+1}=y_{\beta_u+2}=0, y_{\beta_u+3}\neq 0.$$

We may show (23) for the other three cases using a similar argument.

Now, we want to prove our result by contradiction, i.e., assume that any solution set $\{\beta^*, y_{\beta^*+i} \mid 0 \le i \le 3\}$ for the nonlinear program given in (12)-(13) satisfies $\beta^* \le \beta_u - 3$. Consider the two expressions:

$$E^*=\sum_{i=0}^{3}y_{\beta^*+i} \text{ and } E_u=\sum_{i=0}^{3}y_{\beta_u+i},$$

where $\{y_{\beta_u+i} \mid 0 \le i \le 3\}$ is a solution set for *LP*.

(i) If $E^* > E_u$, then E^* is not minimized, implying that $\{\beta^*, y_{\beta^*+i} \mid 0 \le i \le 3\}$ is not a solution set, a contradiction.

(ii) If $E^* = E_u$, then $\{\beta_u, y_{\beta_u+i} \mid 0 \le i \le 3\}$ is also a solution set for the nonlinear program (because $\{y_{\beta_u+i} \mid 0 \le i \le 3\}$ satisfies (12)-(13) when $\beta^* = \beta_u$); but according to the above assumption, every solution set satisfies β^* (which is β_u) $\le \beta_u - 3$, a contradiction.

(iii) If $E^* < E_u$, or equivalently,

$$\sum_{i=0}^{3}y_{\beta^*+i} \le \sum_{i=0}^{3}y_{\beta_u+i}-1,$$

then we have the following contradiction:

$$\begin{aligned} m' &\le \sum_{i=0}^{3}(\beta^*+i)y_{\beta^*+i} && \text{(from (12))} \\ &\le \sum_{i=0}^{3}(\beta_u-3+i)y_{\beta^*+i} && \text{(as } \beta^* \le \beta_u-3 \text{ from assumption)} \\ &= \beta_u(y_{\beta^*}+y_{\beta^*+1}+y_{\beta^*+2}+y_{\beta^*+3})-3y_{\beta^*}-2y_{\beta^*+1}-y_{\beta^*+2} \\ &\le \beta_u(\sum_{i=0}^{3}y_{\beta_u+i}-1)-3y_{\beta^*} \\ &\quad -2y_{\beta^*_u+1}-y_{\beta^*+2} && \text{(from (iii) above)} \\ &= \sum_{i=0}^{3}(\beta_u+i)y_{\beta_u+i}-\beta_u-y_{\beta_u+1} \\ &\quad -2y_{\beta_u+2}-3y_{\beta_u+3}-3y_{\beta^*}-2y_{\beta^*+1}-y_{\beta^*+2}, \end{aligned}$$

which according to (23) results in

(i) for $y_{\beta_u}=y_{\beta_u+1}=y_{\beta_u+2}=0, y_{\beta_u+3}\neq 0$:

$$\begin{aligned} &\le m'+\beta_u+2-\beta_u-y_{\beta_u+1}-2y_{\beta_u+2} \\ &\quad -3y_{\beta_u+3}-3y_{\beta^*}-2y_{\beta^*+1}-y_{\beta^*+2} \\ &= m'+2-3y_{\beta_u+3} < m', \end{aligned}$$

(ii) for $y_{\beta_u}=y_{\beta_u+1}=0, y_{\beta_u+2}\neq 0$:

$$\begin{aligned} &\le m'+\beta_u+1-\beta_u-y_{\beta_u+1} \\ &\quad -2y_{\beta_u+2}-3y_{\beta_u+3}-3y_{\beta^*} \\ &\quad -2y_{\beta^*+1}-y_{\beta^*+2} < m', \end{aligned}$$

(iii) for $y_{\beta_u}=0, y_{\beta_u+1}\neq 0$:

$$\begin{aligned} &\le m'+\beta_u-\beta_u-y_{\beta_u+1} \\ &\quad -2y_{\beta_u+2}-3y_{\beta_u+3}-3y_{\beta^*} \\ &\quad -2y_{\beta^*+1}-y_{\beta^*+2} < m', \end{aligned}$$

(iv) for $y_{\beta_u}\neq 0$:

$$\begin{aligned} &\le m'+\beta_u-1-\beta_u-y_{\beta_u+1} \\ &\quad -2y_{\beta_u+2}-3y_{\beta_u+3}-3y_{\beta^*} \\ &\quad -2y_{\beta^*+1}-y_{\beta^*+2} < m'. \end{aligned}$$

As a result, the preceding assumption always leads to a contradiction and this theorem is thus proved. □

PROOF OF THEOREM 3. Suppose that Θ is a solution for (7)-(8), with

$$v = \sum_{\forall\alpha} \alpha x_\alpha + \sum_{\forall\beta} \beta y_\beta + 11z$$

maximized. It is easy to see that a set of lemmas identical to Lemmas 2-5 can be obtained for solution Θ, because during the refining steps,

(i) the covering radius is kept unchanged and thus (7) is satisfied;
(ii) (22) is ensured, so (8) holds, and
(iii) v remains maximized, as according to (21), v is never reduced. This theorem can be proved by applying repeatedly, in the same way as in Theorem 1, the results of the set of lemmas obtained. □

PROOF OF (19). (By contradiction.) Assume

$$\sum_{i=0}^{3} y_{\beta^*+1} w(\beta^* + i) \le n' - \delta\ w(\beta^*),$$

and let the solution set be refined as: $\beta^0 = \beta^*$, $y^0 = y_{\beta^*} + \delta$, $y^1 = y_{\beta^*+1}$, $y^2 = y_{\beta^*+2}$, $y^3 = y_{\beta^*+3}$. We then have

$$y^0 + y^1 + y^2 + y^3 = y_{\beta^*} + \delta + y_{\beta^*+1} + y_{\beta^*+2} + y_{\beta^*+3} = r'$$

$$\sum_{i=0}^{3} y^i\ w(\beta^0 + i) = (y_{\beta^*} + \delta)\ w(\beta^*) + \sum_{i=1}^{3} y_{\beta^*+i}\ w(\beta^* + i) = \sum_{i=0}^{3} y_{\beta^*+i}\ w(\beta^* + i) + \delta\ w(\beta^*),$$

which is $\le n' - \delta\ w(\beta^*) + \delta\ w(\beta^*) = n'$, according to our assumption. This means that the refined set: β^0, y^0, y^1, y^2, y^3, satisfies the nonlinear program given by (17)-(18). However,

$$m^0 \triangleq \sum_{i=0}^{3} (\beta^0 + i) y^i = \beta^* (y_{\beta^*} + \delta) + \sum_{i=1}^{3} (\beta^* + i) y_{\beta^*+i} > \sum_{i=0}^{3} (\beta^* + i) y_{\beta^*+i} = m',$$

implying that β^*, y_{β^*}, y_{β^*+1}, y_{β^*+2}, y_{β^*+3} are not a solution set (as m' is not maximized). This is a contradiction, resulting from our assumption. Hence (19) holds.

Appendix B

Algorithm 1

Input: n, k (n: system dimension; k: 2^k resource copies)
Output: f, α^*, β^*, z^*, x_{α^*}, y_{β^*}, y_{β^*+1}, y_{β^*+2}, y_{β^*+3}

$f := n$; (initial covering radius)
for α ranging from 0 to int($\log_2(n+1)$) do (int(x) is the integer part of x)
 for z ranging from 0 to int(n/23) do
  for x_α ranging from 0 to 1 do
  {
   $m' := n - k - \alpha x_\alpha - 11z$;
   $n' := n - x_\alpha(2^\alpha - 1) - 23z$;
   if $m' > 0$ & $n' > 0$ then
    {
     β_u := the maximum β such that $\frac{w(\beta)}{\beta+3} < \frac{n'}{m'}$;
     for β ranging from $\beta_u - 2$ to β_u do
      for $y_{\beta+i}$, $0 \le i \le 3$, ranging from 0 to int($n'/(w(\beta+i)+1)$) do
       if $\sum_{i=0}^{3}(\beta+i)\, y_{\beta+i} \ge m'$ & $\sum_{i=0}^{3} y_{\beta+i}\, w(\beta+i) \le n'$ & $x_\alpha + 2\sum_{i=0}^{3} y_{\beta+i} + 3z < f$ then
        {
         $f := x_\alpha + 2\sum_{i=0}^{3} y_{\beta+i} + 3z$;
         $\alpha^* := \alpha$; $\beta^* := \beta$; $z^* := z$; $x_{\alpha^*} := x_\alpha$; $y_{\beta^*+i} := y_{\beta+i}$, for $0 \le i \le 3$;
        }
    }
  }

Algorithm 2

Input: n, r (n: system dimension; r: resource diameter)
Output: $m, \alpha^*, \beta^*, z^*, x_{\alpha^*}, y_{\beta^*}, y_{\beta^*+1}, y_{\beta^*+2}, y_{\beta^*+3}$

$m := 0$; (initial m value)
for α ranging from 0 to int($\log_2(n+1)$) do
 for z ranging from 0 to int($n/23$) do
  for x_α ranging from 0 to 1 do
  {
  $r' := (r - x_\alpha - 3z)/2$;
  $n' := n - x_\alpha(2^\alpha - 1) - 23z$;
  if $r' > 0$ & $n' > 0$ then
  {
  $\beta_u :=$ the maximum β such that $w(\beta) < \frac{n'}{r'}$;
  for β ranging from 2 to β_u do
  for $y_{\beta+i}$, $0 \le i \le 3$, ranging from 0 to int($n'/(w(\beta+i)+1)$) do
  if $\sum_{i=0}^{3} y_{\beta+i} = r' \,\&\, \sum_{i=0}^{3} y_{\beta+i}\, w(\beta+i) \le n' \,\&\, x_\alpha + \sum_{i=0}^{3}(\beta+i)\, y_{\beta+i} + 11z > m$ then
  {
  $m := \sum_{i=0}^{3}(\beta+i)\, y_{\beta+i} + 11z$;
  $\alpha^* := \alpha$; $\beta^* := \beta$; $z^* := z$; $x_{\alpha^*} := x_\alpha$; $y_{\beta^*+i} := y_{\beta+i}$, for $0 \le i \le 3$;
  }
  β_s (or β_u) := minimum (or maximum) β such that $w(\beta+3) > \frac{n'}{r'}$ and $w(\beta) \le n'$;
  for β ranging from max(β_s, 2) to β_u do
  for $y_{\beta+i}$, $0 \le i \le 3$, ranging from 0 to int($n'/(w(\beta+i)+1)$) do
  if $\sum_{i=0}^{3} y_{\beta+i} < r' \,\&\, \sum_{i=0}^{3} y_{\beta^*+i}\, w(\beta^*+i) \le n' \,\&\, x_\alpha + \sum_{i=0}^{3}(\beta+i)\, y_{\beta+i} + 11z > m$ then
  {
  $m := \sum_{i=0}^{3}(\beta+i)\, y_{\beta+i} + 11z$;
  $\alpha^* := \alpha$; $\beta^* := \beta$; $z^* := z$; $x_{\alpha^*} := x_\alpha$; $y_{\beta^*+i} := y_{\beta+i}$, for $0 \le i \le 3$;
  }
  }
  }

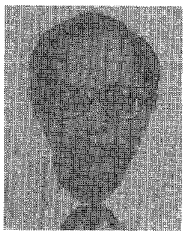
ACKNOWLEDGMENTS

Nian-Feng Tzeng was supported in part by the U.S. National Science Foundation under Grant No. MIP-9201308 and by the State of Louisiana under Contract No. LEQSF(1992-94)-RD-A-32. Gui-Liang Feng was supported in part by the U.S. National Science Foundation under Grant No. NCR-9305038 and by the State of Louisiana under Contract No. LEQSF(1994-96)-RD-A-36.

REFERENCES

[1] Y. Saad and M.H. Schultz, "Topological Properties of Hypercubes," *IEEE Trans. Computers*, vol. 37, no. 7, pp. 867-872, July 1988.
[2] D.A. Reed and R.M. Fujimoto, *Multicomputer Networks: Message-Based Parallel Processing*. Cambridge, Mass.: MIT Press, 1987.
[3] C. L. Seitz, "The Cosmic Cube," *Comm. ACM*, vol. 28, no. 1, pp. 22-33, Jan. 1985.
[4] J.C. Peterson et al., "The Mark III Hypercube-Ensemble Concurrent Computer," *Proc. 1985 Int'l Conf. Parallel Processing*, pp. 71-73, Aug. 1985.
[5] R. Arlauskas, "iPSC/2 System: A Second Generation Hypercube, *Proc. Third Conf. Hypercube Concurrent Computers and Applications*, vol. 1, pp. 38-42, Jan. 1988.
[6] NCube Corp., "*n*-Cube 2 Processor Manual," 1990.
[7] W.D. Hillis, *The Connection Machine*. Cambridge, Mass.: MIT Press, 1985.
[8] M. Livingston and Q.F. Stout, "Distributing Resources in Hypercube Computers," *Proc. Third Conf. Hypercube Concurrent Computers and Applications*, pp. 222-231, Jan. 1988.
[9] A.L.N. Reddy, "Parallel Input/Output Architectures for Multiprocessors," PhD dissertation, Dept. of Electrical and Computer Engineering, Univ. of Illinois, Urbana, 1990.
[10] G.-M. Chiu and C.S. Raghavendra, "Resource Allocation in Hypercube Systems," *Proc. Fifth Distributed Memory Computing Conf.*, pp. 894-902, Apr. 1990.
[11] R.E. Blahut, *Theory and Practice of Error Control Codes*. Reading, Mass.: Addison-Wesley, 1983.
[12] S. Lin and D.J. Costello, *Error Control Coding Fundamentals and Applications*. Englewood Cliffs, N.J.: Prentice Hall, 1983.
[13] G.D. Cohen et al., "Covering Radius—Survey and Recent Results," *IEEE Trans. Information Theory*, vol. 31, pp. 328-343, May 1985.
[14] H.-L. Chen and N.-F. Tzeng, "Fault-Tolerant Resource Placement in Hypercube Computers," *Proc. 20th Int'l Conf. Parallel Processing*, vol. 1, pp. 517-524, Aug. 1991.
[15] F.J. MacWilliams and N.J.A. Sloane, *The Theory of Error-Correcting Codes*. New York: North-Holland, 1977.
[16] R.A. Brualdi, V.S. Pless, and R.M. Wilson, "Short Codes with a Given Covering Radius," *IEEE Trans. Information Theory*, vol. 35, pp. 99–109, Jan. 1989.

[17] P. Ramanathan and S. Chalasani, "Resource Placement in k-Ary n-Cubes," *Proc. 21st Int'l Conf. Parallel Processing*, vol. 2, pp. 133–140, Aug. 1992.
[18] H.-L. Chen and N.-F. Tzeng, "Efficient Resource Placement in Hypercubes Using Multiple-Adjacency Codes," *IEEE Trans. Computers*, vol. 43, no. 1, pp. 23-33, Jan. 1994.



Nian-Feng Tzeng (S'85-M'86-SM'92) received the BS degree in computer science from the National Chiao Tung University, Taiwan, the MS degree in electrical engineering from the National Taiwan University, Taiwan, and the PhD degree in computer science from the University of Illinois at Urbana-Champaign in 1978, 1980, and 1986, respectively.

Dr. Tzeng is currently an associate professor in the Center for Advanced Computer Studies at the University of Southwestern Louisiana, Lafayette, where he has been on the faculty since June 1987. From 1986 to 1987, he was a member of the technical staff at AT&T Bell Laboratories, Columbus, Ohio. His research interests include parallel and distributed processing, high-performance computer systems, fault-tolerant computing, and high-speed networking. He is on the editorial board of *IEEE Transactions on Computers*, has served on program committees of several conferences, and is a distinguished visitor of the IEEE Computer Society. In 1995, he was co-guest editor of a special issue of the *Journal of Parallel and Distributed Computing* on distributed shared memory systems.

Dr. Tzeng is a senior member of the IEEE, a member of Tau Beta Pi and the Association for Computing Machinery, and was the recipient of the outstanding paper award at the 10th International Conference on Distributed Computing Systems in May 1990.

Gui-Liang Feng (S'89-M'92-SM'95) received the BS degree from Fudan University, Shanghai, China, in 1968 and the MS degree from Jiaotong University, Shanghai, in 1982, both in applied mathematics. He received the PhD degree in computer science from Lehigh University, Bethlehem, Pennsylvania, in 1990.

From 1970 to 1979, Dr. Feng was an electrical engineer with the Hudong Shipyard, Shanghai. He was an assistant researcher at the Shanghai Institute of Computer Technology from 1982 to 1986. Since 1991, he has been with the Center for Advanced Computer Studies, University of Southwestern Louisiana, Lafayette, where he is currently an assistant professor. His research interests include error-correcting codes, fault-tolerant computing, cryptography, and computational algebra. He has published more than 40 papers in journals and proceedings of international symposia and conferences. He has received research grants from the NSF, ONR, and LEQSF.

Dr. Feng is a former member of the governing board of the Information Theory Society of the Chinese Institute of Electronics. He received the 1994 IEEE Information Theory Society Paper Award and the Outstanding Paper Award of the Chinese Institute of Electronics for 1984–1987. He is a senior member of the IEEE and a member of the Editorial Board of *IEEE Transactions on Computers*.