Subcube Allocation Strategies in a K-ary N-Cube

Vikas Gautam and Vipin Chaudhary

TR-93-15-15

Subcube Allocation Strategies in a K-ary N-Cube

Vikas Gautam and Vipin Chaudhary

Abstract

This paper proposes extensions of two previous schemes as well as a novel scheme called the sniffing strategy, to recognize m-ary R-subcubes (of base-m and dimension R, henceforth called the subcube) in a k-ary N-cube (of base-k and dimension N, hereby called the hypercube) where m is less than or equal to k. Note that the dimension of the subcube, R, is less than or equal to the dimension of the hypercube, N. The important aspect of the node configuration in a hypercube is that each contiguous pair of nodes differ from the other in one digit only and the difference is either unity (1) or (k-1) for that particular digit. A set of codes having excellent properties needed for node utilization is the Reflected k-ary Gray Codes where each succeeding codeword differs from its predecessor in one digit by unity or, in the cyclic case, by k-1. This paper utilizes a new algorithm to recursively generate these Gray codes and identify a subcube from it. We initially address the problem of subcube allocation in two ways. First the Buddy strategy and later the Gray Code (GC) strategy are considered and extended to the k-ary hypercube from the trivial case of the binary hypercube. It is proved, for the specific case of all subcubes requested being of base k, that the number of subcubes recognized by the Extended GC strategy is k times greater than that of the Extended Buddy Strategy. A minimum (2) and maximum (k) bound on the improvement in these two strategies have been found. It has been shown that using either of these strategies, complete subcube recognition cannot be accomplished, and also, more importantly, they realize subcubes which actually do not exist in the hypercube. Further, it is shown for base-2 subcubes that the Extended Buddy and Extended GC strategies fail in the sense that they recognize non-realizable subcubes. Our **sniffing strategy** on the other hand is foolproof in that each subcube recognized by it actually exists and so can be allocated to the job in hand. This paper also conjectures that for any hypercube of base k, $k > 2$ it is impossible to recognize a subcube of base, m such that $2 < m < k$.

I. INTRODUCTION

Considerable activity has recently been generated in the study of hypercube topology in interconnecting multiple processors because it has become one of the most popular architectures for parallel processing. Popular machines such as the Intel iPSC-2 and Ametek 2010 are examples of k-ary hypercube [8], [1]. The hypercube has many desirable properties like low degree and diameter, strong heirarchy, easy construction and structural regularity, etc.[5], [2]. This paper is concerned with the general form of binary hypercubes called the k-ary hypercube which differs in that its base is not 2 but some other integer k.

The reason of studying k-ary hypercube topology is to determine some additional aspects of the binary hypercubes which have not yet come to light. A hypercube is a network of computers in which the nodes form the processors and the edges constitute the communication channel. A k-ary N-cube is a N-dimensional, base-k hypercube defined as an undirected graph, G=(V,E) with V = 0,1, ...,$k^N - 1$ and $\forall\, i,j \in V,\ (i,j) \in$ E iff the k-ary representation of i and j differ in a single digit by unity or k-1. More formally, we define a hypercube in the following manner. Let

$$< Z >= \{0, 1, 2, 3, ..., Z - 1\}\ for\ some\ integer\ Z \geq 1$$

A n-node base-k hypercube of dimension N called $H_{k,N}$ for short is a graph $G = (V, E)$ where, $V = \{x | x = x_N x_{N-1} \ldots x_2 x_1\ where\ x_i \in < k >,\ i = 1\ to\ N\}$ and $E = \{(x, y) | x, y \in V,\ x = x_N x_{N-1} \ldots x_2 x_1,\ y = y_N y_{N-1} \ldots y_2 y_1$ and there exists j, $1 \leq j \leq k$, such that $y_j = (x_j + 1)\ mod\ k$ and $x_i = y_i$ for i $\neq j$ }. A k-ary hypercube represented by $H_{k,N}$ having base-k and N dimension consists of n=k^N nodes (processors) and each processor is uniquely addressed by a base-k integer. The address scheme is such that the difference in the adjacent nodes is in a single digit only and differs by unity or k-1. The addresses lie between 0 and

$k^N - 1$ and the representation is in N digits (dimension).

In such a hypercube any incoming job may require a subcube to process it and so many such jobs can be executed in parallel using arbitrary dimensions and bases for the subcube. Thus, the problem of subcube allocation assumes importance so as to enable the most efficient distribution of jobs. The major focus of this paper is in the subcube recognition ability of different approaches and a comparison of the same and to the authors' knowledge, this is the first such attempt for k-ary N-cubes. A subcube is defined as a hypercube having base m and dimension R both of which are less than or equal to the base-k and dimension-N of the k-ary hypercube, respectively. A subcube denoted by $S_{m,R}$, of the above mentioned hypercube, $H_{k,N}$, is a contiguous collection of m^R processors which forms a base-m hypercube. Note that the address scheme for every subcube follows the $H_{k,N}$ addresses and $m \leq k, R \leq N$. Each subcube $S_{m,R}$ can be uniquely addressed by a combination of strings from the set $\{*_p, 0, 1, 2, \ldots, m-1\}$, $p \in <k>$, and the number of $*_p$'s in the address equals the dimension R of $S_{m,R}$. A $*_p$ denotes any string from the set $<k>,\ where <k>= \{0, 1, 2, \ldots, k-1\}$. Also note an important fact that whereever there is a $*_p$, the values of p, taken in order are, $(p \oplus_k 0), (p \oplus_k 1), \ldots, (p \oplus_k i), \ldots, (p \oplus_k (m-1))$, where, $\oplus_k$ denotes addition in base k of the hypercube and *not base m, of the subcube.* The address of the subcube $S_{3,2}$ in $H_{4,4}$ is of the form, say $(*_0\ 2\ 1 *_2)_3$, i.e., it includes all nodes with addresses (0212, 0213, 0210, 1212, 1213, 1210, 2212, 2213, 2210). Another example is a subcube, $S_{2,2}$, whose address can be $(2\ *_1\ 1\ *_2)_2$, i.e., it contains the nodes having addresses { 2112, 2113, 2212, 2213 }. For hypercube node identification one can use simple k-ary codes or a special class of codes called the Gray Codes. We have used the simple codes in the Extended Buddy strategy and the Gray Codes in the Extended GC strategy. In this paper we are going to utilize a new algorithm to generate the special class of Codes called the k-ary Reflected Gray Codes (KRGC). This algorithm is a part of another paper [6].

In any multi-user system there is an active request and release of subcubes depending on whether there is an incoming request for job allocation or the job is completed. The dimension, R, of the allocated subcube depends on the minimum number of processors required for that specific job and is numerically equal to $\lceil log_m(I) \rceil$ where m is the base of the subcube and I is the required number of processors for the job. $\lceil x \rceil$ denotes the greatest integer $\geq$ x. Whether a subcube is actually allocated to that request is a different aspect altogether. It is contingent on the fact that an exact number (m^R) of a contiguous set of processors starting from a particular address are simultaneously free or not. This set of nodes should form the nodes of a subcube and this fact is of prime importance. Which starting address is to be chosen depends on which strategy we are working on.

We consider two common strategies for subcube allocation in a binary hypercube and extend them to the case of the k-ary hypercube. The first strategy is the Buddy strategy and the second is the Gray Code strategy (GC strategy), [7], [3], [4] and their extensions to the kary hypercube are respectively called the Extended Buddy strategy and the Extended GC strategy. A new algorithm [6] for the generation of k-ary Gray Codes is also covered which has been used in the Extended GC strategy. This strategy has been shown to be much better in comparison with the Extended Buddy strategy in the following way. For every k-ary hypercube, complete subcube recognition cannot be accomplished using only a single Code. Thus, there is a maximum limit on the number of subcubes which can be allocated using any particular strategy. This limit is greater in the case of the Extended GC strategy than the Extended Buddy strategy for the case when all incoming subcubes request only a base k. In fact, for the particular case of the binary hypercube (k = 2) the number of recognisable subcubes in the Extended GC strategy is twice the number in the Extended Buddy Strategy [3]. We have shown that when the base of the requested subcube is 2 then both these strategies show some serious defects and so we have come up with another strategy called the **Sniffing Strategy** which is a more suitable subcube allocation scheme for a k-ary hypercube.

II. PRELIMINARIES

The generation of Gray Codes of arbitrary base-k is the central theme of the Gray Code strategy. So we came up with an algorithm to generate them in an efficient manner. This algorithm is part of [6] but will be discussed briefly in this paper for completeness. We start by stating a few definitions used for the same.

Definition 1 **Hamming Distance**

We have, x & y as two integers represented by m digits of some base k. The base-k hamming distance between x & y is defined as $h_k(x, y) = m$ *if there exists* $i_1 < i_2 < i_3 \ldots < i_m$ *such that* $x_{i_j} \neq y_{i_j}$*, j lies between 1 to m and* $x_p = y_p,\ p \notin \{i_1, i_2, \ldots, i_m\}$*. In other words* $h_k(x, y)$ *is the number of base-k digit positions in which x & y differ e.g.,* $h_4(1230, 0321) = 4$.

Definition 2 **Base-k N-digit Gray Code** $G^k(N)$

Let n=k^N*. A base-k Gray code is an ordered sequence {* $G^k_0(N)$, $G^k_1(N)$,. . . ,$G^k_{n-1}(N)$ *} of n, N-digit base-k integers, where* $G^k_i(N)$ *denotes the* i^{th} *codeword and the ordering is such that*

$$h_k(G^k_i(N), G^k_{i+1}(N)) \ = 1$$

i.e., each succeeding element differs from its previous code word in just one (1) digit and the difference is either (k-1) or unity (1).

Definition 3 **Partial rank**

Let $S = \{g_1, g_2, \ldots, g_N\}$ *be a sequence of distinct integers. For* $1 \leq i \leq N$*, the partial rank* r_i *of* g_i *is defined as the rank of* g_i *in the set* $\{g_1, g_2, \ldots, g_i\}$ *when the set is rearranged in the ascending order. Thus, to get the partial rank of* g_i*, sort all elements in S from* g_1 *to* g_i *in ascending order and the index of* g_i *in this order is the partial rank* r_i *of* g_i*. For example, the partial rank,* r_1 *of the sequence S = {* $\underline{3}$*,1, 2 } is 1, the partial rank* r_2 *of the sequence S = {3,* $\underline{1}$*, 2 } is again 1 and similarly, the partial rank,* r_3 *of the sequence S = {3,1,* $\underline{2}$ *} is 2*

Definition 4 **Direction i**

Let a sequence of strings have N digits. Then the rightmost (LSB) digit is called the direction 1, second to the rightmost will be called the direction 2, and so on. Consider the string 234$\underline{1}$*01. The direction of the underlined digit is 3.*

Definition 5 **Symbol :** $A^{d\backslash m}$

Let A be a sequence of k-ary strings e.g. 301, 21 etc. of length x-1, where $x > 1$*. Then a* *sequence of k-ary strings of length x, denoted by* $A^{d\backslash m}$*,where* $d \in \{0, 1, \ldots, k-1\}$ *is obtained by the following rules*

1. If $1 \leq m \leq N-1$

Inserting the digit, d into the position immediately right of the digit in the m^{th} *direction of every string in A.*

2. if m=N

Prefixing the digit d to every string in A .

For example Let A = 3121, N=4; then $A^{2\backslash 2} = 312\underline{2}1$ *and* $A^{2\backslash N} = \underline{2}3121$

Definition 6 **<u>Inverse mapping : A*</u>**

For any sequence of k-ary strings, A, let A denote the sequence of k-ary strings obtained from A by reversing the order of strings in A. e.g. Let A =* $\{20,21,01\}$ *Then, A*=* $\{01,21,20\}$ *and,* $(A^*)^{2\backslash 2} = \{\ 0\underline{2}1, 2\underline{2}1, 2\underline{2}0, \}$

A. *ALGORITHM*

Let G_m be a GC (Gray Code) with parameters, g_i, and $S = \{g_1, g_2, \ldots, g_N, \}$ where S is a permutation of $< Z_N >$, and $< Z_N >= \{0, 1, \ldots, Z-1\}$. We also have $G_1 =< k >= \{0, 1, \ldots, k-1\}$. Then, G_N is defined recursively for $2 \leq m \leq N$ as the following:

1. FOR ODD BASE - k

$$G_m = \{G_{m-1}^{0\backslash r_m}, (G_{m-1}^*)^{1\backslash r_m} \ldots G_{m-1}^{k-1\backslash r_m}\}$$

2. FOR EVEN BASE -k

$$G_m = \{G_{m-1}^{0\backslash r_m}, (G_{m-1}^*)^{1\backslash r_m} \ldots (G_{m-1}^*)^{k-1\backslash r_m}\}$$

where r_m as before is the partial rank of g_k. For example, let k=3 and N=3. Also let, S={ 3,1,2 } = $\{\ g_1, g_2, g_3\ \}$. Then, $r_1 = 1, r_2 = 1$ *and* $r_3 = 2$. We have, $G_1 = \{0, 1, 2\}$ And also $G_1^* = \{2, 1, 0\}$.Then we get $G_2 = \{G_1^{0\backslash 1}, (G_1^*)^{1\backslash 1}, G_1^{2\backslash 1}\}$ because, $r_2 = 1$. This implies that $G_2 = \{0\underline{0}, 1\underline{0}, 2\underline{0}, 2\underline{1}, 1\underline{1}, 0\underline{1}, 0\underline{2}, 1\underline{2}, 2\underline{2}\}$ *and* $G_2^* = \{22, 12, 02, 01, 11, 21, 20, 10, 00\}$.Again, since k can go upto N = 3 we have, $G_3 = \{G_2^{0\backslash 2}, (G_2^*)^{1\backslash 2}, G_2^{2\backslash 2}\}$ because, $r_2 = 2$. From this we infer that $G_3 = \{0\underline{0}0, 1\underline{0}0, 2\underline{0}0, 2\underline{0}1, 1\underline{0}1, 0\underline{0}1, 0\underline{0}2, 1\underline{0}2, 2\underline{0}2, 2\underline{1}2, 1\underline{1}2, 0\underline{1}2, 0\underline{1}1, 1\underline{1}1, 2\underline{1}1, 2\underline{1}0, 1\underline{1}0,$
$0\underline{1}0, 0\underline{2}0, 1\underline{2}0, 2\underline{2}0, 2\underline{2}1, 1\underline{2}1, 0\underline{2}1, 0\underline{2}2, 1\underline{2}2, 2\underline{2}2\}$

Lemma 1 *If S is the identity permutation of $< k >$ i.e. $S = \{0, 1, ..., k-1\}$ then the generated Gray Codes are k-ary REFLECTED Gray Codes. Thus if $g_i = i$ then the generated codes are called reflected codes.*

Lemma 2 *Let n=k^N. Then the base-k* **cyclic** *Gray code $G^b(N)$ is a base-k Gray code such that $h_k(G_0^k(N), G_{n-1}^k(N)) = 1$ or $k-1$, which means that the last codeword in the whole code differs from the first one by one single digit.*

We now state two theorems whose proof can be found in [6],[2]

Theorem 1 *If the base b of the codes is even then G_N is cyclic*

Theorem 2 *If the base b of the codes is odd then G_N is not cyclic.*

Note: *A method to generate cyclic Gray Codes for odd bases is given in [6],[2]*

A set of contiguous integers is called a region and we define $<< a, b >>= \{k | a \leq k \leq b, k \in I^+\}$ where I^+ is the set of positive integers. We denote $D_N(q)$ as the general base k representation of the integer q and $G_N(q)$ as the k-ary Gray Code representation for the integer, q. It is obvious that the notation is in N digits, N being the dimension of the hypercube.

III. SUBCUBE ALLOCATION STRATEGIES

Our main emphasis in this paper is to address the problem of subcube allocation so as to have the most efficient scheme. Processors (nodes) must be allocated to incoming tasks in a hypercube so as to enable maximum subcube recognition and minimize system fragmentation within the framework of the selected strategy. We start off by assuming that all incoming request are of same base, k, of the hypercube and consider initially the buddy strategy which has been expounded a number of times for the binary hypercube case. This strategy has been extended to the k-ary hypercube in the form of an Extended Buddy strategy. Another strategy called the Gray Code (GC) strategy is then explained as well as extended and applied to the problem of subcube allocation in a kary hypercube. This Extended GC strategy has been proved to be better than the Extended Buddy Strategy in the sense that more number of subcubes are recognizable. The principle underlying both the strategies is the linear search of an array called the Allocation array in which each entry corresponds to a 0 or a 1 depending on whether the node corresponding to that allocation bit is free or not. Thus a one-to-one mapping exists between the nodes and the Allocation array. As stated earlier, the number of nodes in a hypercube is k^N which means that the size of the array is k^N. To get the dimension of the subcube in such a case (of same base, k) the only parameter required is the dimension, R of the subcube.

In the next section we prove that both these strategies fail for a subcube request of base 2. We show that the Extended GC and Extended Buddy strategies recognize some of those cubes which **cannot exist in that particular hypercube**. This feature is absent in the Sniffing strategy and is an important factor in proving that our scheme is better than those two for the case when m = 2. Now, to recognize an arbitrary subcube, two parameters are required, which are the base and dimension of the subcube. The user can be asked to provide both the parameters or the user can give the number of nodes required for the problem and so the algorithm may have to decide which base and dimension is to be allocated for that problem. But, in this case the decision to find the base and dimension of the subcube, $S_{m,R}$, becomes very complex and so some priority has to be predefined. At this point we state without proof that the choice should be made so as to make the value of R, the dimension of the subcube, as small as possible and also the value m^R must be as close to the required number of nodes. It is obvious that sometimes an exact combination of m and R cannot be determined to exactly equal the requred number of nodes and so a heurestic approach has to be undertaken. We have assumed in this paper that the user knows both the dimension and the base of the required subcube to be allocated.

A. *All Subcubes of same base-k*

We assume for all examples, nowonwards, that the hypercube is absolutely clear at time = 0, i.e., all allocation bits are initially reset (0). We have also assumed that the incoming subcube allocation request is for the base k of $H_{k,N}$. Any incoming request is allocated to a

Figure 1: Connections for a $H_{3,3}$ hypercube. The wraparound condition is shown only for the node 020, but such connections exist for every node. The shaded area shows one of several 3-ary 2-subcubes that can be realized. This particular subcube has the address $(*_0 2 *_0)_2$

subcube $S_{k,R}$ and the required parameter is either R, the dimension of the required subcube, or the number of nodes requested from which R can be obtained with the help of the simple equation $R = \lceil log_k(I) \rceil$, where I is the number of nodes requested.

Theorem 3 *All subcubes of base k and dimension less than or equal to N, where k and N are the base and the dimension of the hypercube, will always* **exist**.

PROOF *Any subcube of same base but different dimension is equivalent to removing a single dimension from the hypercube for each unit difference in N from R, the base of the subcube, but the topology required for the subcube would be exactly like the hypercube. For example, let both N and R be equal to 3, i.e., consider $H_{3,3}$ as shown in the Fig.1. Then, a subcube $S_{3,2}$ is simply one of the planes (1 dimension less because N-R=1) and the conditions required from* *the hypercube of wraparound and hamming distances between nodes are met. The subcube $S_{3,1}$ is two(2) dimensions less than the hypercube but in itself forms a subcube as shown in fig. Extending the same logic we infer that since the essential topology of any subcube of base k, is same as the hypercube, for all R less than or equal to the N, therefore, such a subcube will exist. The proof can also be given by induction. Let the case be true for R=0. This is the obvious case as one can always find a single node. Now let us assume that it is true for some R i.e. $S_{k,R}$ exists and is recognizable. Adding another dimension to this subcube, $S_{k,R}$, implies that we add another degree to each node. Thus a symmetry would exist even after addition of another dimension to this subcube. We know that the hypercube is strongly hierarchial which means that if we disconnect the edges across the same dimension for all the nodes then we obtain a hypercube of dimension R-1 and same base where R is the dimension of the original hypercube. Thus if a hypercube of dimenision R greater than 0 exists then it implies that a subcube of dimension R-1 also exists. Reversing the logic implies that if any hypercube of dimension R exists then a hypercube of dimension R+1, but same base also exists. Or if $S_{k,R}$ exists, then $S_{k,R+1}$ also exists. Thus the proof.*

For this section, wherever we take up an example, the hypercube is to be taken of the form $H_{4,4}$, i.e., of dimension and base, both equal to 4. This assumption, however, is only for illustration and does not restrict our discussion.

1) *Extended Buddy Strategy:* STEP 1

Set $x :=\mid I_j \mid$, where $\mid I_j \mid$ is the dimension of the subcube required to accomodate the j^{th} request, I_j.

STEP 2

Determine the least integer z such that all allocation bits in the Allocation array in $<< zk^x, (z+1)k^x - 1 >>$ are 0's, and set these allocation bits to 1.

STEP 3

Allocate nodes with addresses $D_N(i)$ to the request, I_j, $\forall i \in << zk^x, (z+1)k^x - 1 >>$

Processor Relinquishment:

Once the task is complete, reset every allocation bit, i, in $<< zk^x, (z+1)k^x - 1 >>$ to 0. The nodes with addresses $D_N(i)$ are henceforth deallocated from the request and are free to be used again.

To recognize $S_{4,2}$ in the assumed hypercube $H_{4,4}$, the Extended Buddy strategy would choose the subcube with address (starting from node 0000) in this way: x=2 implies that 0000, 0001, 0002, 0003, 0010, 0011, 0012, 0013, 0020, 0021, 0022, 0023, 0030, 0031, 0032, 0033 would be selected which is the subcube $(00 *_0 *_0)_4$. After this selection let us assume that $S_{4,3}$ is desired. Now the strategy has to find the lowest z to meet its criteria. x=3 implies that searches from allocation bit number $z4^3 = 64z$. It will find z=1 and starting from 1000 it goes on till 128-1=127, i.e., node 1333. Thus, it forms the subcube $(1 *_0 *_0 *_0)_4$. But the important thing to note is that it missed out the nodes from locations 0100 to 0333 which remain as a hole in the hypercube.

Similarly the Extended GC strategy can be explained in terms of the allocation bits by the following methodology,

2) *Extended Gray Code (GC) Strategy:* Here the utlilisation of our algorithm to generate KRGC is used for node identification and the steps involved are:

STEP 1

Set $x := \mid I_j \mid$, where $\mid I_j \mid$ is the dimension of the subcube required to accomodate the j^{th} request, I_j.

STEP 2

Determine the least integer i such that all $(i \; mod \; k^N)th$ allocation bits in the Allocation array in the region $<< zk^{x-1}, (z+k)k^{x-1} - 1 >>$ are 0's. Set all the allocation bits in this region to 1.

STEP 3

Allocate nodes with addresses $G_N(i \; mod \; k^N)$ to the request I_j, $\forall i \in << zk^{x-1}, (z+k)k^{x-1} - 1 >>$.

Processor Relinquishment:

Once the task is complete, reset every allocation bit in $<< zk^{x-1}, (z+k)k^{x-1} - 1 >>$ to zero (0).

The reason for having the mod k^N term is to allow a circular search for a free subcube of the required dimension and base. It is stated that the set of Gray Codes we have used are the Reflected Gray Codes being, obviously, of base k.

Taking the previous example, for the Extended GC strategy. to recognize $S_{4,2}$ the Extended GC strategy would choose (starting from node 0000) in this way: x=2 implies that nodes having allocation bits in the region $<< 0, 15 >>$ would be chosen and the addresses would be 0000, 0001, 0002, 0003, 0013, 0012, 0011, 0010, 0020, 0021, 0022, 0023, 0033, 0032, 0031, 0030. After this selection let us assume that $S_{4,3}$ is desired. Now the strategy has to find the lowest z to meet its criteria. x=3 implies that searches from allocation bit number $z4^2 = 16z$. It will find z=1 and starting from allocation bit No.16 it would search for the next 64 nodes till bit # 79 i.e. the region $<< 16, 79 >>$. The addresses are 0130,0131, ..., 1331. Remember that the nodes are addressed using Gray Codes and that too the reflected ones which have been generated using our algorithm for the generation. Refer to Fig.2 for a set of Gray Codes for the case of $H_{3,2}$. We observe that after the two subcubes have been allocated there is no 'hole' observable as was the case for the Extended Buddy Strategy. This phenomenon will be there for any sequence of subcube requests and the number and lengths of holes will be much more for the case of the Extended Buddy strategy as compared to the Extended GC strategy even though both the strategies do not show complete subcube recognition.

Theorem 4 *The number of subcubes recognized by the Extended GC strategy is at most k times the number of subcubes recognizable by the Extended Buddy strategy where k is the base of the hypercube.*

PROOF *The Extended GC strategy searches for a subcube starting from the location y^{x-1}, y is the base and x is the required dimension of the subcube. The Extended Buddy strategy starts its search from the location y^x. Thus, if all required bases are k i.e. if y=k then k* *times the number of subcubes will be recognized by the Extended GC strategy as compared to the Extended Buddy Strategy. For all possible combinations of any base $\leq k$, the exact number of recognizable subcubes by the Extended GC strategy, though varying on the pattern requested, will always be less than k times the number of recognizable subcubes by the Extended Buddy strategy. This implies that the maximum improvement in subcube recognition for the Extended GC strategy would be k times the subcube recognition ability for the Extended Buddy strategy.*

B. *Subcube of bases 0,1 or 2*

All incoming requests have to be allocated to a subcube, $S_{m,R}$ and the difference from the previous case is that now $R = \lceil log_m(I) \rceil$, where I is the number of nodes requested. The case when m=0 is trivial. For the case when m=1, the case is again trivial because it means that only one node is to be allocated. The third case is for the request of a subcube with base 2. We state a theorem for such a case,

Theorem 5 *If the base of the hypercube is greater than or equal to 2, then there will always exist a recognizable subcube of base 2 and dimension R, i.e., $S_{2,R}, R \leq N$.*

PROOF *There would be at least one recognizable subcube of base-2 whose nodes have addresses differing in a single digit with the difference being 1 or (k-1) for that particular digit. This is nothing but the criteria for a binary hypercube and so we infer that at least one base-2 subcube exists in any hypercube of base greater than 2.*

1) *Extended Buddy Strategy:* STEP 1

Set $x :=\mid I_j \mid$, where $\mid I_j \mid$ is the dimension of the subcube required to accomodate the j^{th} request, I_j. Also, y:= required base of the subcube for the request I_j.

STEP 2

Determine the least integer z such that all allocation bits in the Allocation array in $<< zy^x, (z+1)y^x - 1 >>$ are 0's, and set these allocation bits to 1.

STEP 3

Allocate nodes with addresses $D_N(i)$ to the request, I_j, $\forall i \in << zy^x, (z+1)y^x - 1 >>$

Processor Relinquishment:

Once the task is complete, reset every allocation bit in $<< zy^x, (z+1)y^x - 1 >>$ to 0. The nodes with addresses $D_N(i)$ are henceforth deallocated from the request and are free to be used further.

2) *Extended Gray Code (GC) Strategy:* STEP 1

Set $x := | I_j |$, where $| I_j |$ is the dimension of the subcube required to accomodate the j^{th} request, I_j. Also, y:= required base of the subcube for the request I_j.

STEP 2

Determine the least integer z such that all $(i \; mod \; k^N)th$ allocation bits in the Allocation array, $i \in << zy^{x-1}, (z+y)y^{x-1} - 1 >>$ are 0's. Set all the allocation bits in this region to 1.

STEP 3

Allocate nodes with addresses $G_N(i \, mod \, k^N)$ to the request I_j, $\forall i \in << zy^{x-1}, (z+y)y^{x-1} - 1 >>$.

Processor Relinquishment:

Once the task is over reset every allocation bit in $<< zy^{x-1}, (z+y)y^{x-1} - 1 >>$.

Theorem 6 *The number of subcubes recognizable by the Extended GC strategy is at least twice the number of subcubes recognizable by the Extended Buddy strategy.*

PROOF *The base of any required subcube will be at least 2. The Extended GC strategy searches for a subcube starting from the location y^{x-1}, y is the base and x is the required dimension of the subcube. The Extended Buddy strategy starts its search from the location y^x. Thus, if all required bases are 2, i.e., if y=2 then a minimum of twice the number of subcubes will be recognized by the Extended GC strategy. For all possible combinations of any base ≥ 2 the exact number of recognizable subcubes by the Extended GC strategy, though varying on the pattern requested, will always be greater than twice the number of recognizable subcubes by the Extended Buddy strategy.*

Theorem 7 *The Extended Buddy strategy as well as the Extended GC strategy are erroneous for m=2.*

PROOF *We give the proof using an example. Let us assume that we have a $H_{3,2}$. Now let us assume that the hypercube is entirely free at initial time so any subcube can be allocated. Now, let a request come in for a subcube $S_{2,2}$. With the Extended buddy and the Extended GC strategies, the nodes chosen would be those having addresses (00,01,02,10) and (00,01,02,12) respectively. These are shown in the Fig.2 and it is obvious that they do not form a subcube at all. Thus the Extended Buddy as well as the Extended GC strategy fail for the case and so they are erroneous for m=2.*

3) *Sniffing Strategy:* Before we explain this strategy we would like to review some important notation. $< k >$ denotes the set { 0, 1, . . . , k-1 } and $p, *_p, \# \in < k >$. A $*_p$ implies that the integer p is the starting integer for the address at the digit where $*_p$ is located. For example, the starting integer for $*_2$ is 2. Given a subcube, say $S_{3,2}$, its address can be $(\#\# *_3 *_0)_3$ which means that the direction 1 will have values $0 \oplus_4 0,\ 0 \oplus_4 1,\ 0 \oplus_4 2,\ i.e.,\ 0, 1$ *and* 2 because the hypercube considered is $H_{4,4}$ (of base 4.) Also, direction 2 will have values $3 \oplus_4 0,\ 3 \oplus_4 1,\ 3 \oplus_4 2$, *i.e.*, $3, 0$ *and* 1. We notice that p can be incremented only in steps of one till the increment equals m-1, m is the base of the subcube. A # means that any integer from the set $< k >$ can be placed at that location. Thus, in our example, directions 3 and 4 can have any integer from $< 4 >$, *i.e.*, $\{0, 1, 2, 3\}$. Thus, the subcube can be addressed by $(00 *_3 *0)_3$, $(01 *_3 *0)_3$, $(30 *_3 *0)_3$, $(23 *_3 *0)_3$, etc. In general, we can address any $S_{m,R}$ *in* $H_{k,N}$ with the form

$$(\#_N \#_{N-1} \cdots \#_{R+1} *_{p_R} *_{p_{R-1}} \cdots *_{p_2} *_{p_1})_m,$$

where p_i *and* $\#_i$ denote the integer value of p and # in direction i.

Algorithm used by the strategy

Set $R :=| I_j |$, where $| I_j |$ is the dimension of the subcube required to accomodate the j^{th} request, I_j. Also, m:= required base of the subcube for the request I_j. The number of $*_p$ is equal to the dimension of the subcube which is R and in our strategy the subcube will take

Figure 2: Comparison of strategies

the form $(\#_N \#_{N-1} \ldots \#_{R+1} *_{p_R} *_{p_{R-1}} \cdots *_{p_2} *_{p_1})_m$.

```
BEGIN
  0. for i := R+1 to N do $\#_i := 0$  ; initialising all $\#_i$ variables to 0;
  1. found := false;
  2. for i := 1 to R do   $p_i := 0$  ; initialising all $p_i$ variables to 0
  3. whilenot found do
          begin
               4.checkstars
               5.if $\#_R = \#_{R-1} = \ldots = \#_2 = \#_1 = k$ then goto 8
               else if $\#_1 \leq k-1$ then $\#_1 = \#_1 + 1$
               else if $\#_2 \leq k-1$ then $\#_2 = \#_2 + 1$
               ⋮
               else if $\#_{R-1} \leq k-1$ then $\#_{R-1} = \#_{R-1} + 1$
               else if $\#_R \leq k-1$ then $\#_R = \#_R + 1$
               6.if found then goto 11
          7.end { of while }
  8. if not found then goto 11
  9. Subcube is $(\#_N \#_{N-1} \ldots \#_{R+1} *_{p_R} *_{p_{R-1}} \cdots *_{p_2} *_{p_1})_m$. Allocate the nodes corresponding to these
     addresses to the request.
  10.goto 13
  11.No free subcube exists.
  12.If found then allocate nodes with addresses corresponding to the subcube address to the job else got
  13.END { of program }
procedure checkstars
  BEGIN
     0. while not found do
          begin
               1. if $p_R = p_{R-1} = \ldots = p_2 = p_1 = k$ then goto 4
```

```
                else if $p_1 \leq k-1$ then $p_1 = p_1 + 1$
                else if $p_2 \leq k-1$ then $p_2 = p_2 + 1$
                ⋮
                else if $p_{R-1} \leq k-1$ then $p_{R-1} = p_{R-1} + 1$
                else if $p_R \leq k-1$ then $p_R = p_R + 1$
            2. found:=( Allocation bits for the addresses $(\#_N \#_{N-1} \ldots \#_{R+1} *_{p_R} *_{p_{R-1}} \ldots *_{p_2} *_{p_1})_m$ ar
    3.  end {of while}
  4. END {of procedure}
```

We find that for the example of the $H_{3,2}$ if the request comes for a $S_{2,2}$ then the Sniffing Strategy will find nodes corresponding to the address $(0 *_0 *0)_2$ which are 00,01,10,11 and these form a subcube. This is shown in the Fig.2. Thus, we find that our strategy is error free as compared to the Extended Buddy and the Extended GC strategy.

C. *Subcube with* $3 \leq m \leq k-1$

Conjecture 1 *In any hypercube,*$H_{k,N}$*, a subcube with base, m, such that* $3 \leq m \leq k-1$ **does not exist**.

PROOF *We have verified (that will be a part of our future research) this result by extensive use of examples. Also, with the wraparound necessity in hypercubes, it is a must for the subcube nodes to have addresses (for the nodes having dissimilar digits in only one direction i) such that the maximum difference is 1 or k-1 for that particular direction, k being the base of the hypercube. Consider for example a* $H_{4,2}$*. To find a subcube of the form,* $S_{3,1}$ *let us take the nodes 00,01,02. These will not form a subcube, (refer to the fig.) as the wraparound joining for 00 and 02 is non-existent. Also note that the maximum difference in direction 1 is 2 which is neither 1 nor 4-1=3. Thus, by our conditions as stated in the underlined statement, we conjecture that no subcube will exist.*

Theorem 8 *The Sniffing strategy, can be used for the case when m=k. Thus, as a generalization, the Sniffing strategy is better than the Extended Buddy and Extended GC strategy.*

PROOF *From Theorems 3,4 and 7 and Conjecture 1 we infer that the above statement is true.*

CONCLUSION

In this paper a new algorithm for the faster generation of k-ary Gray Codes has been explained and the set of parameters required for Reflected Gray Codes given. Three strategies which have been explained for each of the three different cases of subcube bases. It has been proved that even though all the strategies do not show complete subcube recognition, yet in terms of sheer number of recognizable strategies, the Extended GC strategy is at least two times and at most k times (k is the base of the hypercube) better than the Extended Buddy Strategy. Our third strategy called the Sniffing strategy is shown to be the best as it is error free for different bases and its virtues are specially clear in the case when there is no restriction of base on the request for subcubes except that $m < 3$ *or* $m = k$, m is the base of the subcube which is requested. We have not proved the optimality of our strategy in the sense that the maximum number of subcubes are being recognized but the important fact is that it does not recognize non-realizable subcubes. This is a study for future research. Also, in the Sniffing strategy, we have not taken into consideration the case when the $\#$ *and* $*_p$ interchange their positions or, in other words, permute. Thus, incomplete subcube recognition is being done, for example in $H_{4,4}$ a subcube, $S_{2,2}$ is searched in the Sniffing strategy in the manner,

$$(\#_4 \#_3 *_{p_2} *_{p_1})_2.$$

We should also search in this fashion: $(\#_4 *_{p_3} \#_2 *_{p_1})_2$ and also as $(*_{p_4} *_{p_3} \#_2 \#_1)_2$, *etc.* We have not incorporated such permutations in our strategy to avoid complicating the algorithm at the cost of lower subcube recognization power. Also, we have not used the case in our algorithm when the request is given in terms of the number of nodes and no specific base or dimension is given. The redressal of such issues will again be a basis for future research.

References

[1] C.L.Seitz et al., " The architecture and programming of the Ametek Series 2010 multi-

computer," Proceedings of the Third Conference on Hypercube Concurrent Computers and Applications, pp. 33-37, Jan. 1988.

[2] L.S. Barasch, S.Lakshmivarahan and S.K.Dhall, "Generalized Gray Codes and their Properties," Lecture Notes in Pure & Applied Mathematics, Vol. 120, Dekker Inc., NY, March, 1989.

[3] Ming-Syan Chen and Kang G. Shin, "Processor Allocation in an N-cube Multiprocessor Using Gray Godes," IEEE Trans. on Computers, Vol. C-36, No.12, pp. 1396-1407, Dec, 1987.

[4] Ming-Syan Chen and Kang G. Shin, "Subcube Allocation and Task Migration in Hypercube Multiprocessor," IEEE Trans. on Computers, Vol. C-39, No.9, pp. 1146-1155, Dec, 1990.

[5] S.Lakshmivarahan and S.K.Dhall, " Analysis and Design of Parallel Algorithms," McGRAW Hill pp. 53-71.

[6] Vikas Gautam and V. Chaudhary, " An Efficient Algorithm for the Recursive Generation of k-ary Gray Codes," Technical Report: PDCL 93-01-01, Wayne State University, Detroit, MI 48202, 1993.

[7] Young Man Kim, Ten-Hwang Lai, Yu Chee Tseng, "Compacting Free Buddy Subcubes in a Hypercube," International Conference on Parallel Processing, 1992.

[8] Y.Saad and M.H. Schulz, "Topological Properties of hypercubes," IEEE Trans. on Computers, Vol.37, No.7, July 1988.