A FAST AND EFFICIENT ISOMORPHIC TASK ALLOCATION SCHEME FOR K-ARY N-CUBE SYSTEMS

D.Doreen Hephzibah Miriam [1], T.Srinivasan [2]
[1] *Postgraduate Student, Sri Venkateswara College Of Engineering*
[2] *Assistant Professor, Department of Computer Science and Engineering*
Sri Venkateswara College Of Engineering, Sriperumbudur, India– 602 105.
doreenhm@gmail.com, tsrini@svce.ac.in

Abstract: A good task allocation algorithm should find available processors for incoming jobs, if they exist, with minimum overhead. Due to its topological generality and flexibility the k-ary n-cube architecture has been chosen for the task allocation problem. We propose a fast and efficient isomorphic processor allocation scheme for k-ary n-cube systems by using isomorphic partitioning where the processor space is partitioned into higher dimensional isomorphic subcube and by using Subcube recognition ability algorithm (SRA) which uses simple coordinate calculation and spatial subtraction. Thus the proposed scheme seeks to reduce the search space drastically, and hence can locate a free subcube very quickly providing scalable, faster, processor allocation, complete recognition ability with minimal overhead and minimizes the fragmentation

Keywords: Full subcube recognition, k-ary n-cube systems, isomorphic partitioning, processor allocation.

1. INTRODUCTION

As the incoming tasks to a system can involve different topologies, the k-ary n-cube is desirable to accept and execute topologically different tasks. An exact number of processors with a particular topology must be allocated to an incoming task. That is, processor allocation in a k-ary n-cube system concerns not only about the dimension (m, $0 \leq m \leq n$) but also about the amount of processors in each dimension (r, $2 \leq r \leq k$), apparently more complicated than that in a binary system.

Please use the following format when citing this chapter:

Miriam, D.D.H., Srinivasan, T., 2006, in IFIP International Federation for Information Processing, Volume 225, From Model-Driven Design to Resource Management for Distributed Embedded Systems, eds. B. Kleinjohann, Kleinjohann L., Machado R., Pereira C., Thiagarajan P.S., (Boston: Springer), pp. 165–174.

The processor allocation problem is much more challenging for k-ary n-cube networks than hyper cubes or meshes because reducing fragmentation within the system involves recognizing both the dimension of the network (as in hypercubes) and the number of processors in each dimension(as in meshes). However, existing processor allocation strategies for the k-ary n-cube [6] - [9] system either recognize only the dimensionality of the subcubes or allow arbitrary partition sizes at the cost of complex search operations.A critical attribute of a processor allocation algorithm is its ability to find available subcubes for incoming requests is called the *subcube recognition ability*. An allocation algorithm is said to have complete subcube recognition ability when it always find a free subcube for an incoming job if one is available.

The rest of the paper is organized as follows: Section 2 presents the pertinent preliminaries. Section 3 presents earlier allocation algorithms in Mesh systems, Hypercube systems, k-ary n-cube systems. Section 4 presents the proposed isomorphic allocation strategy. Section 5 reports the experimental results. Finally, concluding remarks are discussed in Section 6.

2. PRELIMINARIES

2.1 NOMENCLATURE

A k-ary n-cube denoted by Q_k^n, has k^n nodes each of which is identified by n tuple $(a_{n-1},\ldots,a_1,a_0)$ of radix k. where a_i represents the node's position in the i^{th} direction.

Definition 1: A (two-dimensional) **subcube** S(p,q) in the k-ary 2 -cube M(w,h) such that $1\leq p\leq w$ and $1\leq q\leq h$.A subcube is identified by its base and end and is denoted by S[base, end]. Figure 1 depicts a 8 ary 2 cube systems, two busy subcube are indicated in green circles S_1 [(3,4),(6,6)] and S_2 [(0,0),(1,3)].

Definition 2: The **coverage** of a busy subcube β with respect to a job J, denoted as $\xi_{\beta,J}$ is a set of processors such that the use of any node in $\xi_{\beta,J}$ as the base of a free subcube for the allocation of J will make the job J to be overlapped with β.The coverage set with respect to J denoted C_J is the set of the coverages of all the busy subcubes.

Definition 3: The **reject area** with respect to a job J, R_J is a set of processor such that the use of any node in R_J as the base of a free subcube for the allocation of J will make the job J cross the boundary of the k ary n-cube systems.

Definition 4: The **base block** with respect to a job J is a subcube whose nodes can be used as the bases of free subcubes to accommodate the job J. The base set with respect to a job J,B_J is a set of disjoint base blocks with

respect to J. The coverage of two allocated subcubes, the reject area are shown in Figure 2.

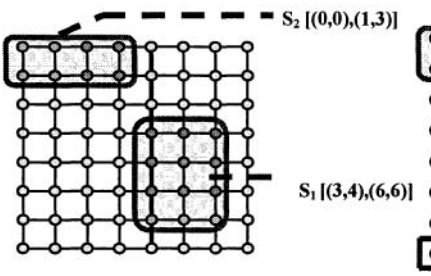

Fig 1: A 8-ary 2- cube system with allocated sub cube

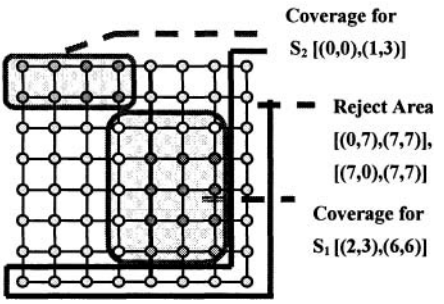

Fig. 2: Coverage of two allocated subcubes, the reject area of 8- ary 2 –cube systems

3. PREVIOUS WORKS

3.1 MESH SYSTEMS

3.1.1 FIRST – FIT (FF)/ BEST – FIT (BF):

These were proposed in [1] to improve the FS Strategy. It maintains a busy array representing the allocation status of the mesh The BF is identical to FF except that BF selects the base in such a way that it has maximum number of busy neighbors. This scheme does not have recognition complete algorithm and excessive runtime overhead due to the manipulation of the array.

3.1.2 FREE LIST:[FL]

The key idea of this scheme proposed in [2] is to maintain a FL list of free sub meshes in the system. This scheme is that it is complicated to update the free list when a sub mesh is released in order to maintain a list of largest possible disjoint free sub mesh and recognition completeness is not achieved.

3.1.3 QUICK ALLOCATION

The basic idea of QA proposed in [3] provides complete sub mesh recognition ability with minimal overhead. For each row it defines covered segment. Column wise scan is avoided. Problems in Quick Allocation are the construction of Covered segment requires a series of row wise scan operation which can be a performance bottleneck.

3.2 K –ary n- cube systems

3.2.1 SLICE PARTITIONING:

Slice partitioning scheme proposed in [4],[5],[6] are shown in Figure 3(a).Jobs requesting different sizes are allocated to one or more partitions of base k and the remaining nodes will be wasted results in internal Fragmentation.

3.2.1.1 EXTENDED BUDDY [EB] AND EXTENDED GRAY CODE [EGC]

EB and EGC as proposed in [4], [5] in which higher dimensions partitioned into lower dimensional subcube. Links as well as the processors are underutilized and have longer internodes distance.

3.2.1.2 K-ARY PARTNER AND MULTIPLE GC

K-ary Partner and Multiple GC as proposed in [6], [5] enhances Sub cube recognition ability over EB and EGC. They utilize the fragmented nodes to form a slice along the other dimension. All these strategies limit the job size to be base k.

3.2.2 JOB PARTITIONING

Job partitioning as proposed in [7],[8] are shown in Figure 3(b).It address the internal fragmentation problem by allowing arbitrary partition sizes rather than restricting them to base – k. The allocation algorithm searches the processor space to find an available subcube for the job by sliding a window frame.

3.2.2.1 EXTENDED FREE LIST (EFL) AND EXTENDED TREE COLLAPSING (ETC)

EFL and ETC as proposed in [7],[8].These algorithm improves subcube recognition ability compacted to Sniffing Strategy. For Allocating 4 ary 2 cube job request in an 8 ary cube includes the cases along the other dimensions. Here, the total number of window positioning is n times of the sniffing strategies.

4. PROPOSED WORK

4.1 THE MAIN APPROACH

The processor allocation algorithm proposed in this paper has complete subcube recognition. This scheme achieves recognition completeness by isomorphically partitioning the k –ary n – cube Systems and manipulating the orientation of the subcube request using SRA algorithm which even reduces both internal and external fragmentation.

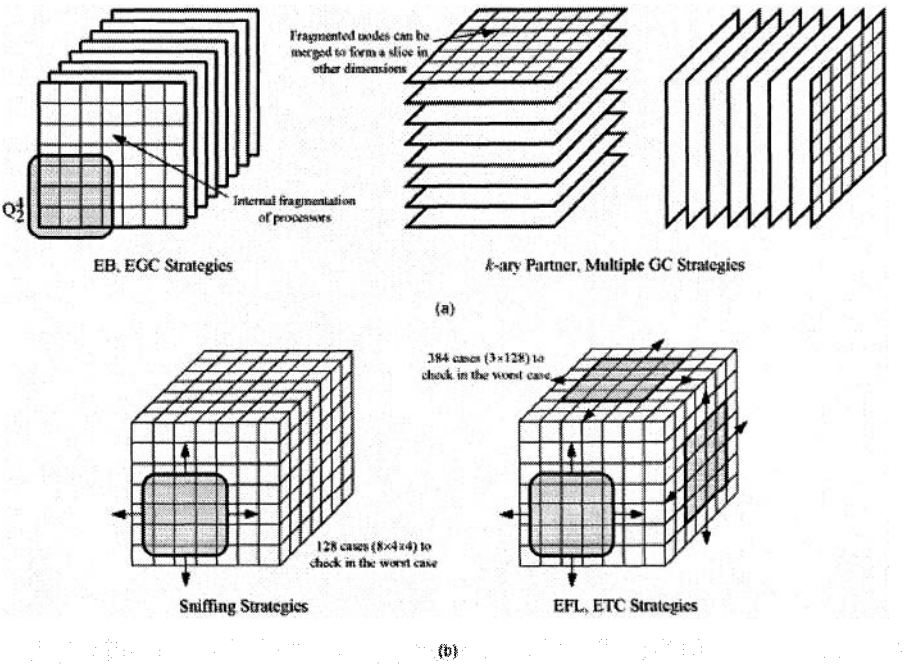

Fig 3: Slice and job-based allocation strategies on an 8-ary 3-cube system. (a) With slice partitioning. (b) With job-based partitioning

4.2 ISOMORPHIC ALLOCATION STRATEGY

We now introduce the isomorphic allocation strategy for Q^k_n. The basic idea is Isomorphic partitioning recursively partitions a k-ary n-cube into 2^n number of $k/2^i$ ary n-cubes where i is the partition step. Processor space is partitioned into higher dimensional isomorphic subcubes and keeping the same order of dimension. Graphical representation of isomorphic partitioning is shown in Figure 4. The basic allocation strategy in produces isomorphic subcube partitions and the job requests to be isomorphic (Q^a_n). Isomorphic partitioning improves Subcube recognition capability, Fragmentation reduction, Complexity compared to other methods.

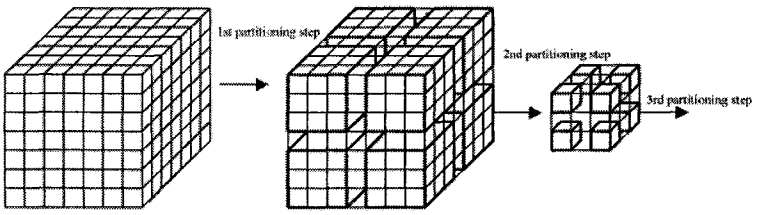

Fig 4: Isomorphic partitioning of an 8-ary 3-cube (8 x 8 x 8).

The resulted partitioned subcubes are said to be "isomorphic" (i-e) n-cube thus they retain properties of k-ary n-cube network Symmetry, Low node degree (2n),Low diameter (kn)

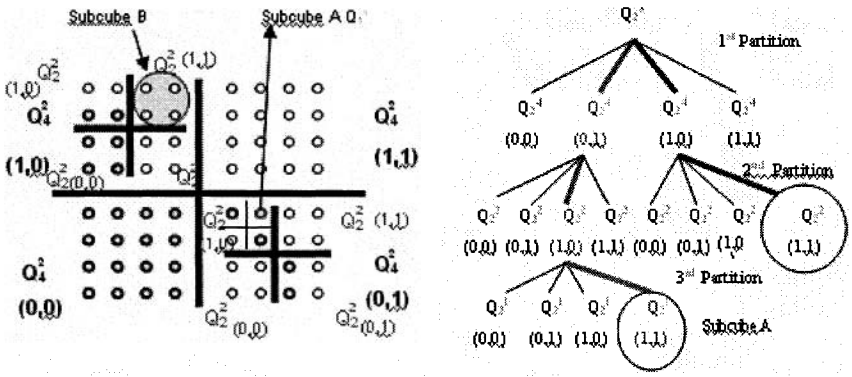

Fig 5 Representation of the isomorphic partitioning of Q^8_2. (a) Subcubes in Q^8_2.

(b) 4-ary tree representation of Q^8_2.

Isomorphic allocation strategy assigns a subcube partition to a job requesting a 2^a-ary n-cube in a 2^k-ary n-cube system, where $a \leq k$.Figure 5a shows the subcubes in an 8-ary 2-cube system (8 x 8 mesh). They can also be described by a 2^n-ary tree (4-ary tree in this example) with k partition steps (three steps in this case), as in Figure 5b.

Consider a subcube A consisting of one node whose address is (3, 5) in Figure 5. (Note that the digits are ordered from right to left, i.e., (a_1, a_0).) .A binary representation of the node is (011, 101). Since $Q^8_2 = H_2 \ \Theta \ H_2 \ \Theta \ H_2$, the node can alternatively be represented by ((0, 1), (1, 0),(1, 1)), where (0, 1) is the address of the node in the first sub graph H_2,(1, 0) is the one in the second H_2, and (1 ,1) is the one in the third H_2. In other words, the subcube A can be addressed by selecting (0, 1) Q^4_2 subcube after the first partition step, (1, 0). Q^2_2 subcube after the second partition step, and, finally, (1,1) Q^1_2 subcube after the third partition step. Similarly, subcube B in Figure 5 can be identified by ((1 , 0),(1 , 1))=(11,01)=(11*,01*). In general, a node in a k - ary n-cube, Q^k_n , is denoted by an n-tuple $(a_{n-1},\ldots..,a_1,a_0)$ where $a_i \in \Sigma k$. We can also denote the node in a full binary representation as

$$(a_{n-1}{}^{(1)},a_{n-1}{}^{(2)}\ldots a_{n-1}{}^{(k/2)},\ldots,a_1{}^{(1)}a_1{}^{(2)}\ldots a_1{}^{(k/2)},a_0{}^{(1)}a_0{}^{(2)}..a_0{}^{(k/2)})$$

where $a_i{}^{(j)} \in \Sigma^2$ .The superscript j in each binary number denotes the partition step.

4.3 SUBCUBE RECOGNITION ABILITY (SRA) ALGORITHM.

The proposed scheme achieves recognition completeness by manipulating the orientation of the subcube request. The proposed scheme quickly finds the base set through simple coordinate calculation and spatial subtraction. First using simple coordinate calculation determine the reject area (R_J) and coverage (C_J) with respect to job that is to be allocated using the busy list, job size, and the system size. The allocation scheme first determines U - R_J and inserts the set difference into the candidate list as initial candidate block. Given an R_J with a sink $<x_s, y_s>$,the initial candidate block is a subcube I_f [<0,0>,< x_s -1,y_s -1>.The initial candidate block along with the coverage with which it intersects is placed in the tray. The coverage in C_J is then spatially subtracted from the initial candidate block.

A spatial subtraction between 2 subcubes is illustrated in Figure 6.The pink and the white rectangles in the figure represent the subtrahend and minuend subcubes respectively.

The newly created candidate block after a spatial subtraction replaces the old block in the candidate list. When more than one candidate block is generated after the spatial subtraction, the length of the candidate list increases and so does the search space. The key idea is to implement the candidate list as a tray. A candidate block on the top of the tray is always compared with the next coverage in the coverage set to see if they intersect with each other. When a new set of candidate blocks is generated after the spatial subtraction, these new candidate blocks are pushed on to the tray replacing the top element. Associated with each candidate block is a pointer to the next coverage that the candidate block should be compared with. When a candidate block is compared with a null pointer appears on the top of the tray, the desired base block is obtained. The node at the top – left corner of the base block is returned as the base of a free subcube.If no such candidate block is found, then the allocation fails. We call this scheme **Subcube Recognition Ability (SRA)** Algorithm.

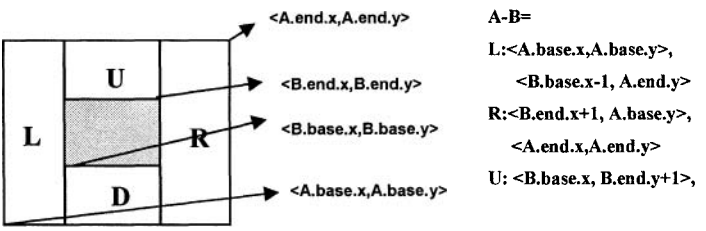

Fig 6:Spatial subtraction between two 2D subcubes *(A-B)*

Subcube Recognition Ability (SRA) Algorithm

Notations

1. C_J [i] denotes the i^{th} coverage set C_J ,where $i \geq 1$.
2. $f_{coverage}$ denotes the coverage that the candidate block *f* that the is to be compared with.
3. *traytop* represent the candidate block on top of the tray.
4. next(k) returns k + 1 if C_J [k+1]exists.Otherwise,it returns null.

Allocation

1. Construct the coverage set C_J w.r.t.job J
2. Determine the initial candidate block *initial.*
3. *Initial* ← next(0).
4. Push *initial* on to the tray.
5. While the tray is not empty do
   If $traytop_{coverage}$ is **null** then return the base of *traytop*
   Else $k \leftarrow traytop_{coverage}$
   If *traytop* intersects with C_J *[k]* then
      Pop up *traytop* from the stack.
      Spatially subtract C_J *[k]* from *traytop*
      For each new candidate block *f* created by the spatial subtraction,
         $f_{coverage}$ ← next(k)
         push *f* onto the tray.
   Else $next(traytop_{coverage}) \leftarrow traytop_{coverage}$
6. If all the orientation of J are considered then return fail.

The allocation of J(2,2) using the SRA algorithm is illustrated .,where a S_1 (4,3) and S_2(2,4) are allocated in 8 –ary 2 – cube systems. The coverages of two subcubes S_1 [(3,4) ,(6,6)] and S_2 [(0,0),(1,3)] are C_1 [(2,3),(6,6)] and C_2 [(0,0),(1,3)] respectively. First the Reject area R_J [(0,7),(7,7)],[(7,0),(7,7)] is calculated then sink with the <7,7> is determined. Then the initial candidate block I_1 [<0,0>,<6,6>] is obtained using the sink and pushed on to the tray with C_1 is shown in Figure 7.

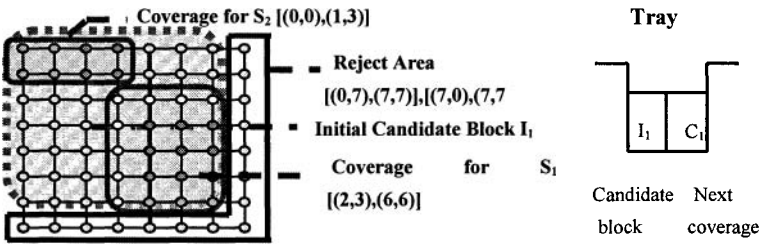

Fig 7: Initial candidate block in first step

Since I_1 and C_1 intersect a spatial subtraction is carried out and two new candidate blocks , I_2 [<0,0>,<6,2>] and I_3 [<0,3>,<1,6>] with the coverages C_2 are pushed onto the tray replacing I_1. I_2 intersects with C_2 and hence a new candidate block I_4 [<(2,0),(6,2)>] is created after spatially subtracting C_2 from B_2 .Since there are no more coverages to compare with, a null pointer is pushed with I_4 indicating that I_4 is the first base block named as B_1 .Any node in B_1 can be used as a base for J. Similarly we find B_2 [<0,4>,<1,6>] are shown in fig 8. An obvious advantage of this versatility is that we can reduce

the allocation overhead significantly while preserving the behavior of the allocation scheme being evaluated.

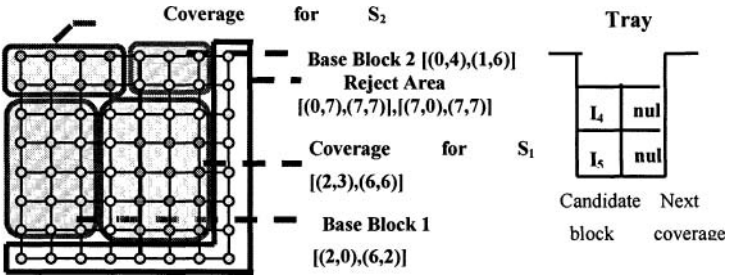

Fig 8: Allocation of 2 x 2 jobs in 8 –ary 2 – cube systems.

5. EXPERIMENTAL RESULTS

The experimental results of Subcube recognition ability algorithm SRA, Emulated Quick allocation EQA, and Emulated First fit EFF are compared with that of a QA for uniform job size distribution. The allocation overhead and average job response time of the four policies are plotted with respect to system loads are shown in Figure 9.

Table 1. Performance evaluation of Four allocation scheme

SYSTEM LOAD	EFF	EQA	QA	SRA
1	0.0222	0.0228	0.0200	0.0170
2	0.0353	0.0369	0.0284	0.0247
3	0.0489	0.0523	0.0378	0.0338
4	0.0656	0.0688	0.0486	0.0458
5	0.084	0.0989	0.0692	0.0592
6	0.1116	0.1221	0.0850	0.0785

Figure 10. shows the variations in mean response time w.r.t system utilization for a 4-ary 2-cube (Q^4_2).The proposed scheme outperforms.

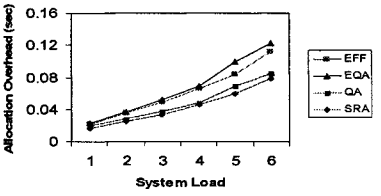

Fig 9: Performance evaluation of four allocation schemes for allocation overhead

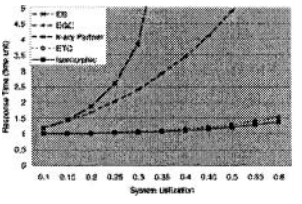

Fig 10.Comparison of mean response time in a Q^4_2systems

6. CONCLUSIONS

This paper addresses the processor allocation problem for k-ary n-cube systems. Our proposed isomorphic partitioning efficiently divides a system into the same dimensional subcubes so that the external as well as internal fragmentation is minimized. Moreover, the resulting partitions are characterized by the same order of dimension as the whole system and, thus, retain the advantages of high order architecture. The isomorphic partitioning mechanism is a novel method for partitioning a k-ary n-cube topology. The proposed allocation algorithm Subcube Recognition ability (SRA) finds a free subcube quickly by reducing the search space drastically through the use of simple coordinate calculation and spatial subtraction. The main objective in developing the algorithm is to speed the allocation process while achieving the maximum attainable performance by guaranteeing complete subcube recognition ability. Experimental results shows that the Subcube Recognition ability algorithm is faster than any allocation scheme reported in literature by showing in the graph plotted System load vs. Allocation overhead.

REFERENCES

[1] Y. Zhu, "Efficient Processor Allocation Strategies for Mesh-Connected Parallel Computers," *J. Parallel and Distributed Computing*, vol. 16, pp. 328-337, 1992

[2] T. Liu, W. Huang, F. Lombardi and L.N. Bhuyan, "A Submesh Allocation Scheme for Mesh Connected Multiprocessor Systems," *Proc.International Conference on Parallel Processing*, vol. II, pp. 193-200, 1995.

[3] S. Yoo, H.Y. Youn and B. Shirazi , "An Efficient Task Allocation Scheme for 2D Mesh Architectures," *IEEE Transactions on Parallel and Distributed Systems*, vol. 8, no. 9, pp. 934-938, 1997.

[4] V. Gautam and V. Chaudhary, "Subcube Allocation Strategies in a K-Ary N-Cube," *Proc. Int'l Conf. Parallel and Distributed Computing and Systems*, pp. 141-146, 1993.

[5] G. Dommety, V. Chaudhary, and B. Sabata, "Strategies for Processor Allocation in k-Ary n-Cubes," *Proc. Int'l Conf. Parallel and Distributed Computing and Systems*, pp. 216-221, 1995.

[6] K. Windisch, V. Lo, and B. Bose, "Contiguous and Non-Contiguous Processor Allocation Algorithms for k-Ary n-Cubes," *Proc. Int'l Conf. Parallel Processing*, 1995.

[7] H.L. Chen and C.T. King, "Efficient Dynamic Processor Allocation for k-Ary n-Cube Massive Parallel Processors," *Computers Math. Applications*, pp. 59-73, 1997.

[8] P.J. Chuang and C.M. Wu, "An Efficient Recognition-Complete Processor Allocation Strategy for k-Ary n-Cube Multiprocessors," *IEEE Trans. Parallel and Distributed Systems*, vol. 11, no. 5, pp. 485-490, May 2000.

[9] *T.Srinivasan, PJS.Srikanth, K.Praveen, L.Harish Subramaniam,*"Parallel AI Game Playing Approach for Faster Processor Allocation in Hypercube Systems using Veitch diagram", *Proc. of 11th IEEE International Conference on Parallel and Distributed Systems - ICPADS 2005*, pp.536-542, July 2005.