Journal of Parallel and Distributed Computing **60**, 998–1027 (2000)
doi:10.1006/jpdc.2000.1638, available online at http://www.idealibrary.com on IDEAL®

Optimal Mappings of q-ary and Binomial Trees into Parallel Memory Modules for Fast and Conflict-Free Access to Path and Subtree Templates[1]

Sajal K. Das

Department of Computer Science and Engineering, The University of Texas at Arlington, Arlington, Texas 76019-0015
E-mail: das@cse.uta.edu

and

M. Cristina Pinotti

Istituto di Elaborazione della Informazione, National Research Council, Pisa, Italy
E-mail: pinotti@iei.pi.cnr.it

Received June 11, 1999; revised January 6, 2000; accepted February 9, 2000

The main memory access latency can significantly slow down the overall performance of a computer system due to the fact that average cycle time of the main memory is typically a factor of 5–10 times higher than that of a processor. To cope with this problem, in addition to the use of caches, the main memory of a multiprocessor architecture is usually organized into multiple *modules* or *banks*. Although such organization enhances *memory bandwidth*, the amount of data that the multiprocessor can retrieve in the same memory cycle, *conflicts* due to simultaneous attempts to access the same memory module may reduce the effective bandwidth. Therefore, efficient mapping schemes are required to distribute data in such a way that regular patterns, called *templates*, of various structures can be retrieved in parallel without memory conflicts. Prior work on data mappings mostly dealt with conflict-free access to templates such as rows, columns, or diagonals of (multidimensional) arrays, and only limited attention has been paid to access templates of nonnumeric structures such as trees. In this paper, we study optimal and balanced mappings for accessing path and subtree templates of trees, where a mapping will be called *optimal* if it allows conflict-free access to templates with as few memory banks as possible. An optimal mapping will also be

[1] This work is partially supported by Texas Advanced Technology Program Grant TATP-003594031, Texas Advanced Research Program Grant TARP-003594013, and Italian Government Progetto Speciale "Sistemi Mobili" (Com. 12).

0743-7315/00 $35.00

called *balanced* if it distributes as evenly as possible the nodes of the entire tree among the memory banks available. In particular, based on *Latin squares*, we propose an optimal and balanced mapping for leaf-to-root paths of q-ary trees. Another (recursive) mapping for leaf-to-root paths of binary trees raises interesting combinatorial problems. We also derive an optimal and balanced mapping to access complete t-ary subtrees of complete q-ary trees, where $2 \leqslant t \leqslant q$, and an optimal mapping for subtrees of binomial trees.

Key Words: binomial tree; complete q-ary tree; conflict-free access; optimal mapping scheme; parallel memory system; path template; subtree template.

1. INTRODUCTION

The CPU speed has traditionally been much faster than the memory access rate, and this divergence in speeds has been constantly increasing in recent years. Typically, the average cycle time of the main memory is higher than the cycle time of a processor by a factor of 5 to 10 and hence the memory access latency can significantly slow down the overall system performance. Therefore, an efficient organization of main memory is important for high-speed computations, particularly if the hardware parallelism in multiprocessor architectures is to be exploited to the fullest extent.

There are two complementary approaches to achieve this goal. The first approach involves the use of a high-speed buffer or *cache* memory, which has about the same cycle time as the processors. Since the cost of cache memory is high, its size is limited and normally cannot replace the entire main memory. The second approach organizes the main memory as a *parallel* or *multibank* memory system. This latter approach has mostly been adopted in multiprocessors which often consist of more memory banks (or modules) than processors. For example, the shared-memory multiprocessor machines NEC SX-3 and CRAY J90, with 4 and 16 processors, respectively, have 1024 memory banks; whereas the Tera MTA has 256 processors and 2^{15} memory banks. Although the ability of a multiprocessor architecture to fetch data in parallel increases with multibank systems, the contentions at the banks may degrade the overall performance and hence the *memory bandwidth*. A bank contention, or *memory conflict*, occurs when two or more processors simultaneously attempt to access the same memory bank.

To avoid or minimize memory conflicts, various mapping schemes that decluster memory references have been proposed in the literature. One of the oldest effective schemes is *M-way interleaving* which uses $M = 2^m$ memory banks. Assuming that there is a total of 2^w words in the parallel memory system, m bits of address suffice to select a bank and the remaining $w - m$ bits can be used to select a word within a bank. In the *low-order interleaving-scheme*, all subsets of 2^m consecutive memory words belong to different banks and can be accessed without conflicts. For example, this scheme can be very useful for conflict-free access to rows of an $M \times M$ matrix stored in M memory banks in a row-major order. However, the M elements of each column can be accessed only sequentially since they are all stored in the same memory bank. Thus, access to different templates (e.g., rows and columns of a

matrix) need different mappings which makes it hard to solve the problem in general by a unified scheme.

The relevance of the memory bank contention problem is such that recently Blelloch *et al.* [3] extended Valiant's *Bulk Synchronous Parallel* (BSP) model with two additional parameters—the bank delay and the expansion factor. The *bank delay* (d) is the throughput at a memory bank and the *expansion factor* (x) is the ratio of the number of memory banks to the number of processors. This new model, called the (d, x)-BSP, can be used to predict the performance of parallel machines with fairly good accuracy in the presence of a large bank delay or a large number of processors (i.e., high contentions). Experimental results in [3] show that if the expansion factor is sufficiently large and the memory access pattern is irregular, a random mapping of the memory locations to banks is enough to balance the memory references across all the banks and therefore to limit the contentions. Nevertheless in all cases where memory is accessed with regular patterns and the expansion factor is not sufficiently large, special data storage schemes can be designed to avoid or minimize bank contentions and thus to increase the memory bandwidth.

In the literature, most of the attention has been paid to designing efficient storage schemes to access arrays and matrices by regular patterns arising in scientific computations (see [17] for a complete list of references). Budnik and Kuck [5] introduced *linear* schemes, which provide conflict-free access to several templates including rows, columns, diagonals, and subarrays of an $N \times N$ matrix using $M > N$ memory banks. Balakrishnan *et al.* [4] proposed a combinatorial approach leading to new *nonlinear* mapping schemes based on *magic* squares, which provide conflict-free access to rows, columns, and diagonals (but not subarrays) of $N \times N$ matrices using exactly N memory banks. This mapping guarantees optimal memory utilization, but it is not *direct* in the sense that it does not compute quickly and locally the memory bank to which an array element is assigned. Kim and Prasanna [17] used the well-known combinatorial objects called the *Latin squares* [14] to tackle this problem. They introduced a new class of Latin squares, called *perfect Latin squares*, which allow conflict-free access to rows, columns, major and minor diagonals, and $\sqrt{N} \times \sqrt{N}$ subarrays within $N \times N$ matrices using N memory banks. These mappings are direct. With the help of the multiplication table of *quasi-groups*, Das and Sarkar [7] also proposed efficient schemes for conflict-free access to array templates such as rows, columns, diagonals, and distributed subarrays using as many memory banks as the template size.

Additionally, some efforts have been made to devise mappings for nonnumeric data structures such as binary and q-ary trees [6, 7, 15, 19]. These results will be summarized in Section 4.

In this paper, we propose new storage schemes for q-ary and binomial trees such that path and subtree templates can be accessed without memory conflicts. Our mapping schemes have the following desirable properties.

- *optimality*: a mapping is *optimal* if it guarantees conflict-free access using as few memory banks as possible. Unlike the arrays, for trees this minimum number not only depends on the template size but also on the structure of the template instances.

• *optimal memory utilization* (or, *balanced memory load*): if the *load* of a memory bank is defined as the number of nodes of the data structure assigned to it, an optimal mapping is *balanced* if it distributes the load as evenly as possible among the available memory banks, preserving the optimality of mapping.

• *direct bank allocation*: a mapping is *direct* if the memory bank to which each tree node is assigned can be computed locally, without global knowledge of assignment of other nodes.

The rest of the paper is organized as follows. Section 2 formalizes the template access problem in data structures. Section 3 presents two optimal and balanced mappings for conflict-free access to leaf-to-root paths. The first mapping is designed only for complete binary trees and poses some intriguing (open) combinatorial problems regarding the memory load at each bank. The second mapping applies to complete q-ary trees, where $q \geqslant 2$, and is also direct because the memory bank to which a tree node is assigned can be obtained from a Latin square. Section 4 deals with the access to complete q-ary subtrees of q-ary trees. It first surveys known results from the literature and then proposes a new mapping which is optimal, direct, and balanced. Such a mapping is also *flexible* in the sense that it optimally solves the conflict-free access problem for any complete t-ary subtree, where $2 \leqslant t < q$, of q-ary trees. Finally, Section 5 presents optimal mappings for accessing subtrees of binomial trees. Section 6 concludes the paper.

2. TEMPLATES OF DATA STRUCTURES

In an idealized shared-memory model, such as the parallel random access machine (PRAM), the memory bandwidth is equal to the number of processors. This is because all the processors can simultaneously access the memory and obtain the required data. This ideal model can be (approximately) realized if the memory is organized as a parallel memory or multibank system and if the data simultaneously requested by the processors belong to distinct memory banks. Consider, for example, a step of a parallel algorithm running on an SIMD (single instruction and multiple data stream) architecture, in which all processors fetch a distinct memory reference. Such references will be available to the processors simultaneously only if they belong to distinct memory banks. Therefore, to achieve optimal performance of a parallel algorithm implemented on a real multiprocessor machine, it is not only enough to distribute the workload evenly among the available processors (such that no processor is idle at any time), but also the bandwidth of the parallel memory system must be large enough to serve all the processor requests simultaneously. To maximize the memory bandwidth, special data mappings that avoid bank conflicts must be designed when the memory is accessed through regular patterns, called *templates*. Ideally, for a multiprocessor system with P processors working on a data structure D, it is desired to have a parallel memory system consisting of at least $M \geqslant P$ memory banks such that any subset of P arbitrary elements of D when mapped into P different banks would guarantee conflict-free access. This memory mapping can be viewed as a *coloring* problem in which the distribution of nodes of the data structure D among M banks is the same

as coloring the nodes from the color-set $\{0, 1, 2, ..., M-1\}$ such that two nodes belonging to the same template are assigned different colors (i.e., memory banks).

Let $G_D = (V_D, E)$ be the graph underlying the data structure D, where a node of V_D is a data element or a node of D and an edge of E is associated to each pair of data elements adjacent in D. A *template*, $\mathscr{A}$, is a subgraph[2] of G_D. Each occurrence of $\mathscr{A}$ in D will be called a *template instance*. For example, if D is a complete binary tree, a leaf-to-root path is an example of a template, and all the paths from the leaves to the root are instances of this template. After coloring the nodes of V_D, a *conflict* is said to occur if two nodes of a template instance are assigned the same color.

An access to a template instance A_i results in k_j conflicts for memory bank j if $k_j + 1$ nodes of A_i are mapped to the bank j. The *cost* of accessing a template instance A_i is given by the maximum number of conflicts that occur over all the memory banks. The cost of accessing the template $\mathscr{A}$ of D is then defined as the maximum cost over all its instances.

Thus, for a parallel memory system consisting of M banks, the goal of the *data access problem* for a given template $\mathscr{A}$ in D, is to find a *mapping* $U^*: V_D \to M$ that minimizes, over all possible memory mappings, the cost of accessing the template $\mathscr{A}$. A mapping $U: V_D \to M$ for template $\mathscr{A}$ of D is *conflict-free* if its cost is equal to 0. Among all possible conflict-free mappings, we are interested in those that use the minimum possible number of memory banks. Thus,

DEFINITION 1. For a given template $\mathscr{A}$ and data structure D, a memory mapping $U: V_D \to M$ is *optimal* if both the following conditions hold: (i) it is a conflict-free mapping: and (ii) there exists no conflict-free mapping that uses less than M memory modules.

Recall that it is desired that an optimal memory mapping is *balanced* and *direct*. In order to determine the lower bound on the number of memory banks required for conflict-free access to the template $\mathscr{A}$, observe that at least Γ memory banks are needed where Γ is the size (the number of nodes) of $\mathscr{A}$. However, this is a necessary but not sufficient condition. In fact, not only the template size but also the overlapping of the template instances in the data structure D is a contributing factor, as justified below.

Let the *associated graph* $G_{D,\mathscr{A}} = (V_D, E^+)$ be obtained from the data structure graph $G_D = (V_D, E)$ by adding to E an edge between every pair of nodes r and s of D if there exists a template instance A_i of $\mathscr{A}$ such that $r, s \in A_i$. Now, to find an optimal mapping, colors must be assigned to the nodes of $G_{D,\mathscr{A}}$ so that every pair of nodes connected by an edge is assigned two distinct colors and the minimum number of colors is used. In other words, the problem of finding an optimal memory mapping can be reduced to the classical coloring problem on the associated graph $G_{D,\mathscr{A}}$. Clearly, the size of the largest *clique* in $G_{D,\mathscr{A}}$ is a lower bound on the number of memory banks required for conflict-free access to the template $\mathscr{A}$ in D, where a clique is a maximal subset of nodes having pairwise adjacency.

[2] In this paper, we only consider *connected* subgraphs of D as templates.

Let us introduce a few notations and terminology related to a rooted tree, T. The *level* of a node $v \in T$ is defined as the number of edges on the path from v to the root which is assumed to be at level 0. The maximum level of the nodes is the *height*, h, of T. Counting from left to right, the jth node at level i will be denoted as (i, j) where $i, j \geqslant 0$.

A *complete q-ary tree*, T^q, is a rooted tree having all its leaves at the same level and all the internal nodes with exactly q children. A complete q-ary tree of height h, denoted as T^q_h, has $(q^{h+1}-1)/(q-1)$ nodes. Level i of T^q contains q^i nodes, numbered from left to right $(i, 0), (i, 1), ..., (i, q^i-1)$. The rth *ancestor* of a node (i, j) is the node $(i-r, \lfloor j/q^r \rfloor)$, while its children in order from left to right are the nodes $(i+1, qj), (i+1, qj+1), ..., (i+1, qj+q-1)$.

3. LEAF-TO-ROOT PATH TEMPLATES OF COMPLETE TREES

This section proposes optimal mappings to access leaf-to-root paths in complete trees.

We define a *leaf-to-root path template*, P^{h+1}, of a tree T^q_h as an ascending path of $h+1$ nodes starting from a leaf and going up to the root. The leaf-to-root path access problem was first studied by Das *et al.* [10] while designing cost-optimal parallel algorithms for implementing insert and delete operations on heap data structures on the hypercube multicomputers. In order to minimize the communication overhead among the hypercube processors, it was necessary to find an efficient mapping of nodes lying on the paths of the heap represented as a complete q-ary tree.

In the following we design two optimal and balanced mappings that guarantee conflict-free access to all the leaf-to-root paths of a complete tree of height h, using $h+1$ memory banks. The proposed mappings balance the load on all the banks, except for the one storing the root of the tree. Since the root belongs to every leaf-to-root path, it must be stored in a separate bank to guarantee conflict-free path access. Therefore, for a path template, a mapping will be called *balanced* if all the nodes of the tree T^q_h, except the root, are evenly distributed among the remaining h banks.

Note that it is easy to design an optimal mapping for conflict-free access to paths by assigning a distinct memory bank (color) to each level of the tree such that all nodes belonging to the same level are assigned the same color. However, since the number of nodes at level i is q times the number of nodes at level $i-1$, the load at memory banks will be highly unbalanced.

Our first mapping, called PATH-COLORING-RECURSIVE, works on complete binary trees and leads to some interesting combinatorial problems. The second mapping, called PATH-COLORING-DIRECT, can be applied to complete trees of any arity and is based on the combinatorial objects known as the *Latin squares*. A preliminary version of these results appeared in [8].

3.1. Recursive Coloring of T^2_h

For simplicity of presentation, we present a mapping scheme that recursively colors binary trees, T^2_h, of height h. Supposing that T^2_{h-1} has already been colored

using colors from the set $S=\{0, 1, ..., h-1\}$, the tree T_h^2 is recursively colored by decomposing it into three parts: the root of T_h^2, the left subtree LT_{h-1}, and the right subtree RT_{h-1}. The color assignment of LT_{h-1} is the same as that of T_{h-1}^2. A new color h is assigned to the root of T_h^2. To color RT_{h-1}, we first count how many times (frequency) each color has been used in LT_{h-1}, and then we reuse the colors from $S=\{0, 1, ..., h-1\}$ exactly the same way as in LT_{h-1} except that the least frequently used color in LT_{h-1} is replaced with the most frequently used color, the second least frequently used color is exchanged with the second most frequently used one, and so on.

Figure 1a illustrates a coloring of the binary tree T_2^2 of height two, which forms the basis of our recursive scheme. Color 2 is used only once for the root, while colors 0 and 1 are used three times each. To color the tree T_3^2 of height three, as depicted in Fig. 1b, the assignment of the left subtree LT_2 is copied from Fig. 1a and the root is assigned the new color 3. The assignment of the right subtree RT_2 is obtained from LT_2 such that color 0 is replaced with color 2 and vice versa.

The coloring in Fig. 1b is then used in Fig. 2 to color the nodes of T_4^2, in which color 4 is used only once for the root node and colors $\{0, 1, 2, 3\}$ are used 8, 7, 8, 7 times, respectively. Keeping aside the new color for the root, colors 0 and 2 are thus the most frequently used while colors 1 and 3 are the least frequently used. The mapping scheme is formally described below.

PROCEDURE *PATH-COLORING-RECURSIVE* (P^{h+1}, $h+1$, T_h^2),

1. Decompose the T_h^2 into the root, left subtree LT_{h-1}, and right subtree RT_{h-1}.
2. If $LT_{h-1} \neq empty$ then PATH-COLORING-RECURSIVE (P^h, h, LT_{h-1}).
3. Assign the new color h to the root of T_h^2.
4. If $LT_{h-1} \neq empty$ then
   (a) Let $f_i =$ frequency of color i in LT_{h-1}, where $0 \leqslant i \leqslant h-1$.
   (b) Sort f_i's in the increasing order.
   Let $\mathscr{C}(i)$ be the color corresponding to the ith element in this sorted sequence.
   (c) Define a reverse mapping Φ: $S \to S$, such that $\Phi(\mathscr{C}(i)) = \mathscr{C}(h-i)$.
   (d) Color the right subtree RT_{h-1} with the help of LT_{h-1} but using color $\Phi(\mathscr{C}(i))$ wherever color $\mathscr{C}(i)$ was used in LT_{h-1}.

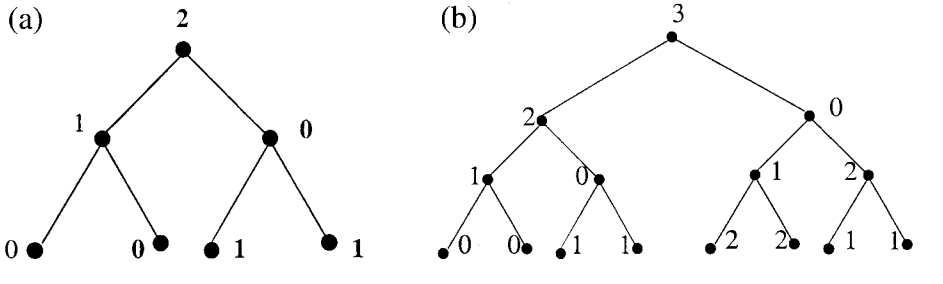

FIG. 1. (a) Coloring of T_2^2. (b) Coloring of T_3^2.

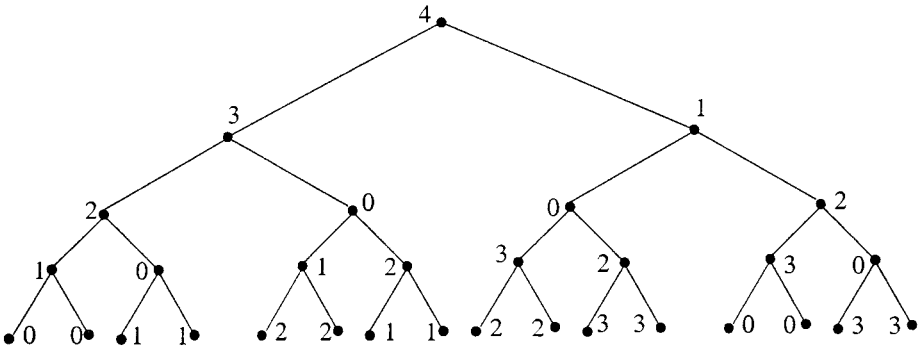

FIG. 2. Coloring T_4^2 using the assignment of T_3^2.

Clearly, the nodes on the leftmost path of the tree T_h^2 are colored 0, 1, 2, ..., h starting from the leaf up to the root. By induction on the height h, it is easy to prove that the procedure *PATH-COLORING-RECURSIVE* assigns distinct colors to all the nodes in any instance of the leaf-to-root path template. Therefore, this mapping scheme is optimal.

Let us now analyze the *load* on each memory bank by determining bounds on the maximum and minimum frequency of use of each color (except the root color h). Let $f_{h,j}$ denote the frequency of the color j used in a complete tree T_h^2. Let $x_h = \max_{0 \leqslant j \leqslant h-1} \{f_{h,j}\}$ and $y_h = \min_{0 \leqslant j \leqslant h-1} \{f_{h,j}\}$ be respectively the maximum and minimum load on the memory banks. According to our coloring scheme, $x_h \leqslant 2x_{h-1}$ for $h \geqslant 2$. Moreover,

LEMMA 1. *It holds* $y_h = x_{h-1} + 1$ *for* $h \geqslant 2$.

Proof (by induction on h). It is easy to verify that the claim holds for $h = 2$, which forms the basis. Assume the lemma is true for trees of height $\leqslant h-1$, but not true for height h. In other words, $y_h < x_{h-1} + 1$ where $x_{h-1} + 1$ certainly belongs to the sequence $\{f_{h,j} \mid 0 \leqslant j \leqslant h\}$ since there exists a color which is used $x_{h-1} + 1$ times according to the PATH-COLORING-RECURSIVE procedure. In such a case, y_h is the sum of two elements in the sequence $\{f_{h-1,j}\}$ such that $y_h = f_{h-1,q} + f_{h-1,l}$ for some $q, l \leqslant h-1$. Using the fact that $2x_{h-2} \geqslant x_{h-1}$, we get

$$\begin{aligned} f_{h-1,q} + f_{h-1,l} &\geqslant (x_{h-2} + 1) + (x_{h-2} + 1), \qquad \text{by induction hypothesis} \\ &> x_{h-1} + 1. \end{aligned}$$

Hence, $y_h = x_{h-1} + 1$ for $h \geqslant 2$. ∎

Applying this lemma, it immediately follows:

LEMMA 2. $(x_h / y_h) \leqslant (2 \cdot x_{h-1})/(x_{h-1} + 1) < 2$.

THEOREM 1. *The PATH-COLORING-RECURSIVE mapping is optimal for accessing leaf-to-root paths* P^{h+1} *of complete binary trees* T_h^2. *The ratio between the maximum to minimum loads is less than* 2.

Following our scheme, we computed the minimum and maximum loads for binary trees of height $h \leqslant 30$, which are shown in Table 1. This tabular data also

TABLE 1

Maximum and Minimum of $\{f_{h,j}\}$'s

h	x_h	y_h	h	x_h	y_h
3	6	4	17	17800	9661
4	8	7	18	32844	17801
5	15	9	19	61866	32845
6	24	16	20	117520	61867
7	46	25	21	222729	117521
8	78	47	22	422998	222730
9	130	79	23	820008	422999
10	237	131	24	1562374	820009
11	446	238	25	2972179	1562375
12	786	447	26	5665146	2972180
13	1442	787	27	10931673	5665147
14	2653	1443	28	21084763	10931674
15	5202	2654	29	40426936	21084764
16	9660	5203	30	78013980	40426937

implies that $x_h/y_h \geqslant 3/2$ for $h \geqslant 5$, which we have yet to prove analytically for all values of h. Additional properties of this sequence $\{f_{h,j} \mid 0 \leqslant j \leqslant h\}$ of color frequencies (see Table 2) include: (a) $x_h - y_h \leqslant x_{h-1}$ and $y_h \leqslant 2y_{h-1} - 1$; (b) it is symmetric; (c) if h is even, the maximum load, x_h, appears at least twice in the sequence. The last property leads to the proof that $(x_{h+1})/(y_{h+1}) \geqslant 3/2$ when the height h is even. However, a straightforward inductive proof cannot be applied for odd h because sorting of colors based on their frequencies followed by a reverse mapping (as followed in the proposed algorithm) destroys the induction properties and complicates the analysis. Therefore, we leave the following as an open combinatorial problem.

Open Problem. Prove that $x_h/y_h \geqslant 3/2$ for $h \geqslant 5$.

We believe an accurate characterization of the sequence $\{f_{h,j}\}$, say in terms of a generating function, would help settle the problem.

3.2. Direct Coloring of T_h^q

The PATH-COLORING-RECURSIVE algorithm just described must know the coloring of the entire left subtree in order to color the nodes of the right subtree. Hence, by definition, such a mapping is not direct.

In this section, we present a direct mapping scheme to access leaf-to-root paths in a complete q-ary tree T_h^q of height $h = q^k$, where $k \geqslant 2$. The case for trees of other heights can he handled with minor modifications. For convenience, we count the levels of T_h^q from -1 (root) to $h-1$ (leaves) and use the set of memory banks (colors) $\{-1, 0, ..., h-1\}$. As before, the root gets a unique color, say -1, since it is common to every path.

TABLE 2

Sequence of Elements $\{f_{h,j}\}$, for $0 \leqslant j \leqslant h$ and $0 \leqslant h \leqslant 14$

$h \cdot \cdot . j$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	1														
1	2	1													
2	3	3	1												
3	4	6	4	1											
4	7	8	8	7	1										
5	9	15	14	15	9	1									
6	16	24	23	23	24	16	1								
7	25	40	39	46	39	40	25	1							
8	47	65	65	78	78	65	65	47	1						
9	79	125	112	130	130	130	112	125	79	1					
10	131	209	209	237	237	237	237	209	209	131	1				
11	238	368	368	446	418	418	418	446	368	368	238	1			
12	447	684	656	786	786	736	736	786	786	656	684	447	1		
13	787	1233	1233	1442	1392	1420	1368	1420	1392	1442	1233	1233	787	1	
14	1443	2229	2207	2653	2625	2625	2601	2601	2625	2625	2653	2207	2229	1443	1

Intuitively, the mapping *PATH-COLORING-DIRECT* perceives the tree T_h^q as an array such that: (i) each path belongs to a column of the array, and (ii) each diagonal of the array stores almost the same number of tree nodes.

We first partition the tree T_h^q (excluding the root) into q parts, horizontally as well as vertically (Fig. 3). This divides the tree into q^2 cells, organized as a $q \times q$ matrix $\mathscr{G}$. Then we form q groups of colors from the set $\{0, ..., h-1\}$ such that each group has $\frac{h}{q}$ colors. The groups are represented as G_γ, where $0 \leqslant \gamma \leqslant q-1$. Roughly speaking, to every cell of $\mathscr{G}$ is assigned one of the groups, G_γ, such that every group of colors occurs exactly once in each row and each column of the matrix G, as in a *Latin square matrix* [14]. Precisely, if the first row of the matrix $\mathscr{G}$ is assigned the groups of colors in the order $\{G_0, G_1, ..., G_{q-1}\}$, every other row is formed by shifting the previous row cyclically one position to the right. Subsequently, and in some way recursively, the nodes in the cell $\mathscr{G}[i, j]$, where $0 \leqslant i, j \leqslant q-1$, are partitioned in a matrix C of size $\frac{h}{q} \times \frac{h}{q}$, which is colored as a Latin square using the group of colors assigned to that cell in the previous step.

PROCEDURE *PATH-COLORING-DIRECT* $(P^{h+1}, h+1, T_h^q)$,

1. Assign color -1 to the root of T_h^q.
2. Divide the tree horizontally into q rows, each consisting of $\frac{h}{q}$ consecutive levels of the tree. The ith row consists of levels $\{i(\frac{h}{q}), ..., (i+1)\frac{h}{q}-1\}$, for $0 \leqslant i \leqslant q-1$.
3. Divide the tree into q columns vertically. The jth column consists of the entire subtree rooted at the jth child of the root of the tree T_h^q, where $0 \leqslant j \leqslant q-1$. Nodes with indices $\{jq^{i-1}, ..., (j+1)\, q^{i-1}-1\}$ in level i are assigned the jth column.

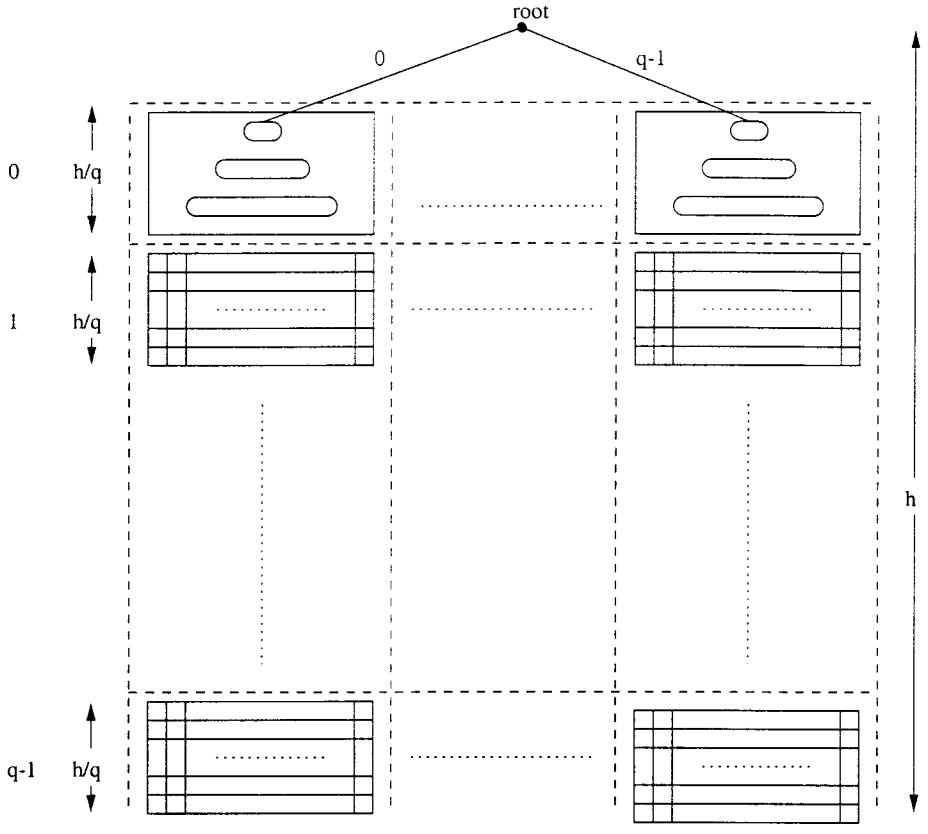

FIG. 3. A sketch of the algorithm PATH-COLORING-DIRECT.

4. Divide the h colors $\{0, ..., h-1\}$ into groups of size $\frac{h}{q}$. Let G_γ be the group containing the set of colors $\{\gamma(\frac{h}{q}), ..., (\gamma+1)\frac{h}{q}-1\}$ where $0 \leqslant \gamma \leqslant q-1$.

5. The tree T_h^q has been partitioned into q^2 cells arranged as a $q \times q$ grid $\mathscr{G}$. Let $\mathscr{G}[i, j]$ denote the (i, j)th cell, which is assigned using the set of colors in the group $\mathscr{G}_{(j-i) \bmod q}$. The assignment of colors to the individual nodes in a cell is described in Steps 6 and 7 below. This is equivalent to creating a $q \times q$ Latin square with symbols from the set $\{G_0, G_1, ..., G_{q-1}\}$.

6. Color the nodes in the cell $\mathscr{G}[0, j]$ level by level, for $0 \leqslant j \leqslant q-1$. The nodes in level r in $\mathscr{G}[0, j]$ are assigned the color $j(\frac{h}{q})+r$, belonging to the group G_j, where $0 \leqslant r \leqslant \frac{h}{q}-1$.

7. To assign the nodes in $\mathscr{G}[i, j]$, $1 \leqslant i \leqslant q-1$ and $0 \leqslant j \leqslant q-1$ using the set of colors assigned previously, we partition $\mathscr{G}[i, j]$ into h^2/q^2 subcells arranged as a $\frac{h}{q} \times \frac{h}{q}$ grid. Let $C[r, s]$ be the (r, s)th subcell in $\mathscr{G}[i, j]$, where $0 \leqslant r, s \leqslant \frac{h}{q}-1$. The cell $C[r, s]$ will contain nodes (u, v) such that $u = i(\frac{h}{q})+r$ and $v \in \{jq^u + s(q^{u+1}/h), ..., jq^u + (s+1)(q^{u+1}/h)-1\}$. Each cell in level u contains q^u nodes and each subcell contains q^{u+1}/h nodes.

8. The subcell $C[r, s]$ in the matrix cell $\mathscr{G}[i, j]$ is assigned the memory bank $M[r, s] = [(j-i) \bmod q]\frac{h}{q} + (s-r) \bmod \frac{h}{q}$. This is the same as creating a

Latin square with the colors from the set $\mathscr{G}_{(j-i) \bmod q}$ and distributing them to the subcells.

THEOREM 2. *The* PATH-COLORING-DIRECT *mapping is optimal for accessing leaf-to-root path template* P^{h+1} *of complete q-ary trees* T_h^q *of height* $h = q^k$, *where* $k \geqslant 2$.

Proof. To prove optimality, we show that no conflict occurs on any leaf-to-root path. Each such path P belongs to the jth array column of the grid $\mathscr{G}$, for some j, where $0 \leqslant j \leqslant h/q - 1$. Moreover, the nodes of P at levels $i(\frac{h}{q})$, $i(\frac{h}{q}) + 1$, ..., $(i+1)\frac{h}{q} - 1$ belong to $\mathscr{G}[i, j]$ and form a column of the grid arranged in the matrix cell $\mathscr{G}[i, j]$. Since, by construction, the grid arranged in each matrix cell and the grid $\mathscr{G}$ are Latin squares, no memory conflicts can arise on a path. Finally, since at least $h+1$ colors are needed for conflict-free access to the path P^{h+1}, the PATH-COLORING-DIRECT mapping is optimal. ▌

THEOREM 3. *The* PATH-COLORING-DIRECT *mapping is direct.*

Proof. Given a tree node (u, v) such that $u = i(\frac{h}{q}) + r$ and $v = jq^u + s(q^{u+1}/h) + t$, where $0 \leqslant i, j \leqslant q-1$, $0 \leqslant r, s \leqslant \frac{h}{q} - 1$, and $0 \leqslant t \leqslant (q^{u+1}/h) - 1$, we know that the node belongs to the subcell $C[r, s]$ of the cell $\mathscr{G}[i, j]$. According to the mapping scheme, the node (u, v) is assigned to the memory bank

$$M[u, v] = \begin{cases} jq^{k-1} + r = j\left(\dfrac{h}{q}\right) + r & \text{if} \quad i = 0 \\ ((j-i) \bmod q)\dfrac{h}{q} + \left((s-r) \bmod \dfrac{h}{q}\right) & \text{otherwise.} \end{cases}$$

Since this is a closed form formula, the memory assignment can be computed locally for any node. Hence the theorem. ▌

For the analysis of the memory load, let the frequency f_i, where $0 \leqslant i \leqslant h-1$, be the number of nodes having color i in the q-ary tree of height $h = q^k$. Let $x_i = \max\{f_i \mid 0 \leqslant i \leqslant h-1\}$ and $y_i = \min\{f_i \mid 0 \leqslant i \leqslant h-1\}$ be, respectively, the maximum and minimum load on memory banks.

THEOREM 4. *The PATH-COLORING-DIRECT mapping of complete q-ary trees of height* $h = q^k$, *with* $k \geqslant 2$, *is balanced for all the memory banks but the one storing the root, with the asymptotic maximum-to-minimum load ratio as* 1.

Proof. Distinct sets of colors are assigned to the cells of the first row of the grid $\mathscr{G}$. Each cell $\mathscr{G}[0, j]$, for $0 \leqslant j \leqslant q-1$, is a q-ary subtree of height $(\frac{h}{q}) - 1$ and colored level-by-level. Thus, after coloring the first row of $\mathscr{G}$, the maximum and minimum loads are, respectively, 1 and $(q)^{(h/q)-1}$.

Since the cells and the subcells are both assigned with the help of Latin squares, every memory bank appears the same number of times in every row. So the load of a bank in the remaining parts of the tree is given by

$$\sum_{z=(h/q)+1}^{h} \frac{q^z}{h} = \frac{(q)^{(h/q)+1}}{h}\left(\sum_{z=0}^{h-(h/q)-1} q^z\right)$$
$$= \left(\frac{(q)^{(h/q)+1}}{h}\right)\frac{(q)^{h-(h/q)}-1}{h}$$
$$= \frac{q^{h+1}-(q)^{(h/q)+1}}{h(q-1)}.$$

Therefore, the minimum memory load is

$$y_i = \frac{q^{h+1}-(q)^{(h/q)+1}}{h(q-1)} + 1$$

and the maximum memory load is

$$x_i = \frac{q^{h+1}-(q)^{(h/q)+1}}{h(q-1)} + q^{(h/q)-1} = y_i + q^{(h/q)-1} - 1.$$

For $q \ll h$, the ratio of the maximum-to-minimum load is $x_i/y_i \approx 1$ asymptotically, implying that the load is equally distributed among various memory banks. ∎

In the case when h is not a power of q but a multiple of q, our mapping scheme can be generalized by simply changing the partitioning of the cells $\mathscr{G}[i, j]$, into h^2/q^2 subcells arranged as a square grid, where $1 \leqslant i \leqslant q-1$ and $0 \leqslant j \leqslant q-1$. In the first row of the cell $\mathscr{G}[i, j]$, we create $(\frac{h}{q}-1)$ subcells each of size $\lfloor \frac{(q)^{i(h/q)}}{h/q} \rfloor$ and by putting the remaining elements in the last subcell, the subcells in the subsequent rows are formed by the children of the nodes from the corresponding cells in the previous row. However, this does not change the asymptotic load distribution among various memory banks. In the case when h is not even a multiple of q, we color the tree as if it were a tree of an extended height $h' = \lceil \frac{h}{q} \rceil q$ with empty nodes.

Finally, given fewer memory modules than the number of nodes on a path P^{h+1}, i.e., $M < h+1$, the results in this section can be extended as follows. We partition the tree T_h^q into rows, each of height $M-1$. Then we color each T_{M-1}^q for conflict-free access to its leaf-to-root paths P^M. In this way, exactly $\lceil \frac{h+1}{M} \rceil - 1$ conflicts occur on each path of T_h^q, which is the optimal.

4. SUBTREE TEMPLATES OF COMPLETE q-ARY TREES

This section is devoted to the subtree access problem in complete q-ary trees, T^q, of any height. A comprehensive summary of previous research reveals that none of the existing schemes is optimal as well as direct. After reviewing the literature, we describe a mapping for accessing complete q-ary subtrees of T^q. Then, we extend it to optimally access subtrees of any arity, that is, complete t-ary subtrees of T^q where $2 \leqslant t \leqslant q$.

We define a *subtree template* $S^{K,t}$ in the tree T^q as a complete t-ary subtree of size K and height $h = \lfloor \log_q K \rfloor$, where $2 \leqslant t \leqslant q$. Also let $S^{K,q}(i, j)$ denote the instance of the template $S^{K,q}$ rooted at node (i, j) of T^q.

4.1. Previous Work

In the following we review various existing mapping schemes and analyze their merits and demerits.

4.1.1. *Regular mappings.* Wijshoff [19] extended the class of *regular* mappings, which were earlier known to work efficiently for template access in d-dimensional arrays, to access templates of tree structures. This class of mappings is defined as follows.

DEFINITION 2 [19]. Given two nodes x and y of a data structure mapped to the sane memory bank, a conflict-free mapping is *regular* if it implies that any two nodes which are in the same relative position to each other as x and y are also mapped to the same bank.

A regular mapping can be compactly described by a *basis*, B, which lists the minimal set of node relations (i.e., rules) needed to represent all the nodes assigned to the same memory bank. Figure 4 shows an example of an optimal, regular mapping of binary trees for conflict-free access to binary subtrees $S^{3,2}$ (of size 3) using three memory banks $\{0, 1, 2\}$. Given a node α of a tree, the basis for the mapping is given by:

$$B = \{[\alpha, \mathit{Left}(\mathit{Left}(\alpha))], [\alpha, \mathit{Right}(\mathit{Right}(\alpha))], [\alpha, \mathit{Left}(\mathit{Right}(\mathit{Left}(\alpha)))], [\alpha, \mathit{Right}(\mathit{Left}(\mathit{Right}(\alpha)))]\}.$$

To the best of our knowledge, there exists no algorithm in the literature on how to construct such regular mappings. As will be explained later, our proposed mapping for subtree access belongs to this class. However, our algorithm description does not rely on the concept of *basis*.

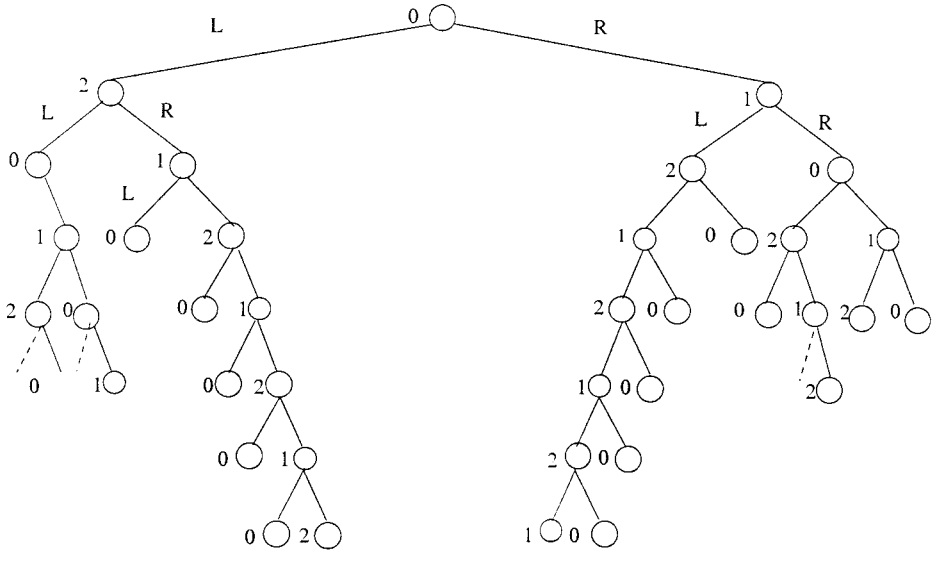

FIG. 4. A regular mapping to access $S^{3,2}$.

4.1.2. *Isotropic mappings.* Gössel and Rebel [15] first studied how to store binary trees into parallel memory systems with the help of a class of mappings called *isotropic mappings* which are defined as follows.

DEFINITION 3 [15]. Given a data structure D and V_D as the set of its elements which are related to each other by a set of functions F: $V_D \to V_D$, a memory mapping U: $V_D \to M$ is called *isotropic* if $U(k) = U(k') \Rightarrow U(f(k)) = U(f(k'))$ for every pair of elements $k, k' \in V_D$ and for all $f \in F$.

Considering the set of functions $F = \{Left, Right\}$ to describe binary trees, Table 3 shows an isotropic mapping that uses $M = 9$ memory banks for conflict-free access to complete binary subtrees, $S^{7,2}$, of size 7 (i.e., height two). Figure 5 shows the coloring of the tree T_4^2 with the help of such a mapping.

It has been proved in [15] that there exists no isotropic conflict-free mapping for $S^{7,2}$ that uses less than nine memory banks. Moreover, as proved by Creutzburg [6], all the isotropic mappings for $S^{7,2}$ are either isomorphic to that in Table 3 or to the mapping given in Table 4. However, we are not aware of any existing algorithm for constructing such isotropic mappings.

As will be shown in Section 4.2, it is possible to access conflict-free the template $S^{K,q}$ in T_h^q using exactly K memory bans. Therefore, the described isotropic mappings are not optimal.

It is worth pointing out that there is no relationship between regular and isotropic mappings. In fact, it can be shown that these two classes of mappings are not equivalent, nor one does imply the other in general. Indeed, there exist mappings which are isotropic but not regular. Let us refer to the isotropic mapping given in Table 5. The nodes assigned to the memory bank 0 follow the rules:

$$B = \{[\alpha, Left(Right(Right(\alpha)))], [\alpha, Left(Left(Right(Left(\alpha)))).\}$$

Now, let us apply the first rule to the the node $\alpha = (1, 1) = Right(\text{root})$. The two nodes $[Right(\text{root}), Left(Right(Right(Right(\text{root}))))]$, which are in the same relative position as the node-pair $[\text{root}, Left(Right(Right(\text{root})))]$, are assigned to the memory banks 2 and 7.

Similarly, there exist regular but nonisotropic mappings. Consider the mapping of Fig. 4 in which the two nodes, $Left(root)$ and $Left(Right(root))$, are assigned to the same bank. Their left children are assigned to different banks and the same holds for their right children too. Therefore, the isotropic condition is not satisfied.

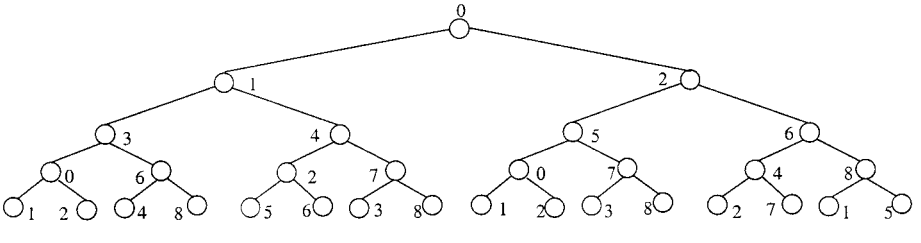

FIG. 5. Coloring of T_4^2 according to the mapping in Table 1.

TABLE 3

An Isotropic Mapping U for Conflict-Free Access to $S^{7,2}$ in Binary Trees

$\ddots$ $U(k)$	0	1	2	3	4	5	6	7	8
$U(Left(k))$	1	3	5	0	2	0	4	3	1
$U(Right(k))$	2	4	6	6	7	7	8	8	5

4.1.3. *Recursively linear mapping.* Creutzburg [6] proposed a proper subclass of isotropic mappings, called the *recursively linear* mappings. Using $M = q^{h-2}[q(h-1)+1]$ memory banks, conflict-free access to complete q-ary subtrees $S^{K,q}$ of height h and size $K = (q^{h+1}-1)/(q-1)$ can be guaranteed by adopting the following recursively linear mapping U for the ith child of node k (counting the children from left to right, starting from 1),

$$U(qk+i) = aU(k) + i \bmod M \qquad \text{for} \quad i = 1, ..., q,$$

where a is given by

$$a \equiv \begin{cases} 1 \bmod (q+1) & \text{for} \quad h = 1, \\ -2q \bmod [q(2q+1)] & \text{for} \quad h = 2, \\ 1q^{h-2} \left| \dfrac{1}{q^{h-2}} \right|_{\bmod(q(h-1)+1)} & \\ \quad + 2(q(h-1)+1) \left| \dfrac{1}{q(h-1)+1} \right|_{\bmod q^{h-2}} & \text{for} \quad h > 2, \end{cases}$$

where $|\frac{1}{x}|_{\bmod y}$ is the multiplicative inverse of x mod y. That is, the number z such that $xz \equiv 1 \bmod y$. For example, $|\frac{1}{3}|_{\bmod 5} = 2$ since $3 * 2 \equiv 1 \bmod 5$.

This class of mappings requires a large number of memory banks, as illustrated in Table 6.

Table 7 describes the recursively linear mapping $U(4k+i) = (28U(k)+i) \bmod 36$ for conflict-free access to 4-ary subtrees $S^{21,4}$ of size 21 in 4-ary trees.

4.1.4. *Breadth-first mapping.* So far we have discussed conflict-free, but nonoptimal, mappings for accessing subtree templates in binary or q-ary trees. Das and

TABLE 4

Another Isotropic Mapping for Conflict-Free Access to $S^{7,2}$

$\ddots$ $U(k)$	0	1	2	3	4	5	6	7	8
$U(Left(k))$	1	3	5	0	2	1	3	5	0
$U(Right(k))$	2	4	6	7	8	8	4	6	7

TABLE 5

Isotropic but Not Regular Mapping for Template $S^{7,2}$

$\ddots$ $U(k)$	0	1	2	3	4	5	6	7	8
$U(Left(k))$	1	3	5	5	7	7	0	0	1
$U(Right(k))$	2	4	6	6	8	8	4	3	2

Sarkar [7] proposed an optimal mapping for conflict-free access to templates $S^{K,q}$ of height $h = \lfloor \log_q K \rfloor$. Their mapping colors level-by-level the tree T^q of any height as follows. First, the subtree $S^{K,q}(0,0)$ is colored level-by-level with all distinct memory banks. Thus, the q^{h+1} nodes at level $h+1$ of T^q are numbered consecutively from left to right and partitioned into q blocks such as $B_{h+1,0}$, ..., $B_{h+1,q-1}$, each of size q^h. The leftmost block $B_{h+1,0}$ consists of the leaves of the subtree $S^{K,q}(1,0)$ rooted at node $(1,0)$. The nodes in $B_{h+1,0}$ are assigned those colors which are used in $S^{K,q}(0,0)$, but not yet used in $S^{K,q}(1,0)$. Block $B_{h+1,1}$ consists of the leaves of the subtree $S^{K,q}(1,1)$. The nodes of $B_{h+1,1}$ are assigned to the colors used in $S^{K,q}(0,0)$, but not yet used in $S^{K,q}(1,1)$, and so on. In general, assuming that the tree has been colored up to level i, the nodes at level $i+1$ are partitioned, from left to right, in q^{i+1-h} consecutive blocks, each of size q^h. Block $B_{i+1,j}$, with $0 \leqslant j \leqslant q^{i+1-h}-1$, consists of the leaves of the subtree $S^{K,q}(i+1-h, j)$. The nodes of $B_{i+1,j}$ are then assigned the colors already used in $S^{K,q}(i-h, \lfloor \frac{j}{q} \rfloor)$, but not yet used in $S^{K,q}(i-h+1, j)$.

The breadth-first mapping applied to color the ternary tree T_3^3 of height 3 for optimally accessing the subtree template $S^{4,3}$ is shown in Fig. 6. Although this mapping scheme is optimal, it is not direct. This is because to color a new node, we must know how a subtree of size K is colored.

4.2. New Results on Accessing Subtree Templates

Now we propose an optimal and direct mapping scheme to access subtrees of any arity in complete q-ary trees. In fact, the memory bank to which a tree node is

TABLE 6

Recursively Linear Mappings for Conflict-Free Access to $S^{K,q}$ in q-ary Trees

Subtree height	3-ary tree	4-ary tree	5-ary tree	6-ary tree	7-ary tree	8-ary tree
	$K, M, \frac{K}{M}$	$K, M, \frac{K}{M}$	$K, M, \frac{K}{M}$	$K, M, \frac{K}{M}$	$K, M, \frac{K}{M}$	$K, M, \frac{K}{M}$
1	4, 4, 1	5, 5, 1	6, 6, 1	7, 7, 1	8, 8, 1	9, 9, 1
2	13, 21, 1.42	21, 36, 1.71	31, 55, 1.77	43, 78, 1.81	57, 105, 1.84	73, 126, 1.86
3	40, 90, 2.25	85, 208, 2.44	156, 400, 2.56	259, 684, 2.64	400, 1078, 2.69	594, 1600, 2.70
4	121, 351, 2.90	341, 1088, 3.19				
5	364, 1296, 3.56					

TABLE 7

The Isotropic and Recursively Linear Mapping $U(4k+i)=(28U(k)+i) \bmod 36$

$\ddots\ U(k)$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
$U(4k+1)$	1	29	21	13	5	33	25	17	9	1	29	21	13	5	33	25	17	9
$U(4k+2)$	2	30	22	14	6	34	26	18	10	2	30	22	14	6	34	26	18	10
$U(4k+3)$	3	31	23	15	7	35	27	19	11	3	31	23	15	7	35	27	19	11
$U(4k+4)$	4	32	24	16	8	0	28	20	12	4	32	24	16	8	0	28	20	12
$\ddots\ U(k)$	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
$U(4k+1)$	1	29	21	13	5	33	25	17	9	1	29	21	13	5	33	25	17	9
$U(4k+2)$	2	30	22	14	6	34	26	18	10	2	30	22	14	6	34	26	18	10
$U(4k+3)$	3	31	23	15	7	35	27	19	11	3	31	23	15	7	35	27	19	11
$U(4k+4)$	4	32	24	16	8	0	28	20	12	4	32	24	16	8	0	28	20	12

assigned depends only on the position of the node in that tree. A preliminary version of these results appeared in [18].

Let us consider complete q-ary subtrees $S^{K,q}$ of height $h=\lfloor \log_q K \rfloor$ in the q-ary trees T^q. The mapping COLOR-SUBTREE presented below uses K memory banks, the same as the template size. For each node $(i, j) \in T^q$, let $j=j_{i-1}\cdots j_0$ be the q-weighted representation of j, that is, $j=\sum_{r=0}^{i-1} j_r q^r$, and let $M[i, j]$ be the memory bank assigned to the node (i, j).

PROCEDURE *COLOR-SUBTREE* ($S^{K,q}$, K, T^q)

- $M[(0, 0)]=0$
- for all $(i, j) \in T^q$, where $i \geqslant 1$,

$$M[(i, j)]=\left(\sum_{r=0}^{i-1} (j_r+1)\, q^{i-1-r}\right) \bmod K$$

To simplify the calculation of the memory bank to which a tree node (i, j) is assigned, the 3-weighted representation of j is given in Fig. 7. The edge between a node and its $(s+1)$th child, where $0 \leqslant s \leqslant q-1$, is labeled s in this figure. Figure 8 illustrates the coloring of T_3^3 according to the procedure *COLOR-SUBTREE*.

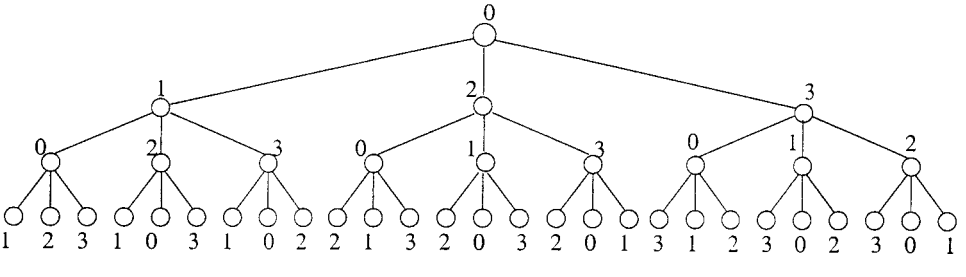

FIG. 6. The breadth-first mapping for conflict-free access to $S^{4,3}$ of T_3^3.

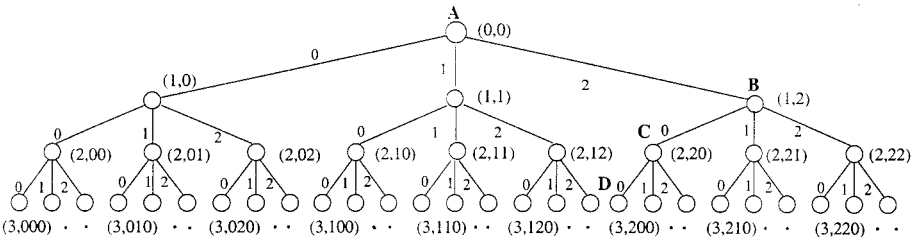

FIG. 7. On the right of each node (i, j) is shown the 3-ary weighted representation of position j.

Before showing that the COLOR-SUBTREE mapping is optimal, let us first prove the following result.

LEMMA 3. *The* COLOR-SUBTREE *mapping assigns all the K colors from the closed interval* $[0, ..., K-1]$ *to the nodes of the subtree template instance* $S^{K,q}(0,0)$ *of height* $h = \lfloor \log_q K \rfloor$. *In particular, the leaves of* $S^{K,q}(0,0)$ *are colored with the colors in* $[(q^h-1)/(q-1), ..., K-1]$.

Proof. To show that all the nodes of $S^{K,q}(0,0)$ are assigned distinct colors, we use an inductive reasoning on the level of the subtree template. The basis for induction is $h=1$. The root is colored 0, and the nodes at level 1 are assigned to the colors $[1, ..., q=(q^2-1)/(q-1)-1]$.

By inductive hypothesis, we assume that all the nodes of $S^{K,q}(0,0)$ up to level $h-1$ are assigned to the memory banks $[0, ..., (q^h-1)/(q-1)-1]$ and all the q^{h-1} nodes at level $h-1$ are assigned the colors from the interval $[(q^{h-1}-1)/(q-1), ..., (q^h-1)/(q-1)-1]$.

Now the COLOR-SUBTREE algorithm colors the nodes at level h as follows. Those q^{h-1} nodes at level h, which are the leftmost $(j_0=0)$ children of the nodes at level $h-1$, are colored from the interval

$$\left[\frac{q^{h-1}-1}{q-1}+q^{h-1}, ..., \frac{q^{h-1}-1}{q-1}+q^{h-1}+q^{h-1}-1\right];$$

that is,

$$\left[\frac{q^h-1}{q-1}, ..., \frac{q^h-1}{q-1}+q^{h-1}-1\right].$$

Similarly, those q^{h-1} nodes at level h, which are the second children from the left $(j_0=1)$ of the nodes at level $h-1$, are colored

$$\left[\frac{q^{h-1}-1}{q-1}+2q^{h-1}, ..., \frac{q^{h-1}-1}{q-1}+q^{h-1}+2q^{h-1}-1\right];$$

that is,

$$\left[\frac{q^h-1}{q-1}+q^{h-1}, ..., \frac{q^h-1}{q-1}+2q^{h-1}-1\right].$$

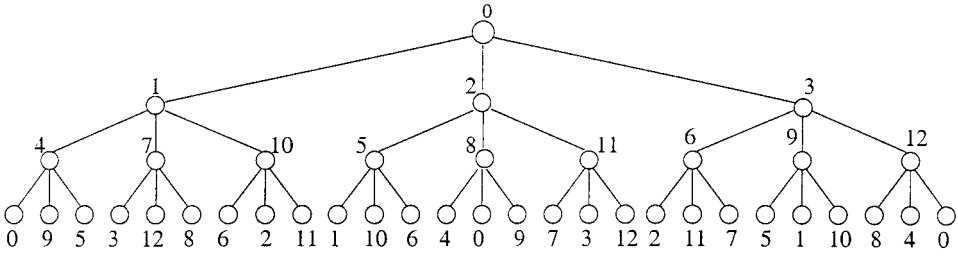

FIG. 8. The COLOR-SUBTREE mapping applied to T_3^3 to optimally access the $S^{13,3}$ template.

In general, the q^{h-1} nodes at level h, which are the sth children of the nodes at level $h-1$ ($j_0 = s-1$), are colored

$$\left[\frac{q^h-1}{q-1} + sq^{h-1}, ..., \frac{q^h-1}{q-1} + (s+1)\, q^{h-1} - 1\right],$$

and the q^{h-1} nodes at level h, which are the rightmost children of the nodes at level $h-1$ ($j_0 = q-1$), are colored

$$\left[\frac{q^h-1}{q-1} + (q-1)\, q^{h-1}, ..., \frac{q^{h+1}-1}{q-1} - 1\right].$$

Thus, altogether the nodes at level h are assigned to the memory banks $[(q^h-1)/(q-1), ..., (q^h-1)/(q-1)+q^h-1]$. Since, by inductive hypothesis, the nodes up to level $h-1$ are assigned to the memory banks $[0, ..., (q^h-1)/(q-1)-1]$, the lemma follows. ▌

To prove that the above mapping is optimal, let us count the children of a node in the left-to-right order, where the edge from a node to its sth child is labeled with s, for $0 \leqslant s \leqslant q-1$, as shown in Fig. 7. Then, the path from the root to any node (i, j) contains i edges, and the labels of such edges are uniquely identified on the tree T^q by the q-ary representation $(j_{i-1}, ..., j_0)$ of j. Let the $(i-m)$th ancestor of the node (i, j) be denoted as the node $(m, n) = (m, \lfloor j/q^{i-m} \rfloor)$. Then, the ascending path of $i-m$ edges from the node (i, j) to node (m, n) is identified by the $i-m$ least significant digits of the q-weighted representation of j, that is, $j \bmod q^{i-m}$. And, the ascending path from (m, n) to the root is uniquely identified by the most significant m digits of the q-weighted representation of j, that is, by the value $\lfloor j/q^{i-m} \rfloor$ expressed in q-ary radix.

For example, in Fig. 7, the top-down edge labels on the path from node $D = (3, 18)$ to the root $A = (0, 0)$ are 2, 0, 0. The labels on the path from $B = (1, 2)$ to $D = (3, 18)$ coincide with the two least significant digits of the 3-ary representation 2, 0, 0 of the position 18 at level 3 of D, while the label on the path from $A = (0, 0)$ to $B = (1, 2)$ coincides with the most significant digit.

THEOREM 5. *The* COLOR-SUBTREE *mapping is optimal for accessing complete q-ary subtree template $S^{K,q}$ of complete q-ary trees.*

Proof. For a given node (i, j) in the tree T^q, consider its $(i-m)$th ancestor $(m, n) = (m, \lfloor j/q^{i-m} \rfloor)$. Recall that the q-ary subtree template of size K rooted at the node (m, n) is denoted as $S^{K,q}(m, n)$. Let W be the set of q-ary representations of the positions j of all the K nodes of T^q that belong to $S^{K,q}(m, n)$. Note that:

(a) the most significant m digits of all the representations in W are identical, since the ancestor of all the nodes in $S^{K,q}(m, n)$ is the node (m, n);

(b) the representations in W have different lengths since the nodes of $S^{K,q}(m, n)$ belong to different levels.

Now, let W_R be a set containing the q-ary representations of the nodes of W, each truncated of the m most significant digits. The truncated q-ary representation of $j' = j \bmod q^{i-m}$ of position j belonging to W_R represents the position of the node (i, j) relative to the template instance $S^{K,q}(m, n)$. In order to prove that any template instance is colored without conflicts, we consider the nodes of $S^{K,q}(m, n)$ in relation with the nodes of $S^{K,q}(0, 0)$. Let us associate the node (i, j) of T^q, whose relative position in $S^{K,q}(m, n)$ is $(i, j' = j \bmod q^{i-m})$, with the node $(i-m, j' = j \bmod q^{i-m})$ of $S^{K,q}(0, 0)$. Clearly, $(i-m, j' = j \bmod q^{i-m})$ has the same position in the subtree $S^{K,q}(0, 0)$ as the node (i, j) has in $S^{K,q}(m, n)$. Finally, the color of the node (i, j) can be derived from the color assigned to its associated node $(i-m, j' = j \bmod q^{i-m})$ in $S^{K,q}(0, 0)$ as follows.

$$
\begin{aligned}
M[(i, j)] &= \left(\sum_{r=0}^{i-m-1} (j_r + 1)\, q^{i-1-r} + \sum_{r=i-m}^{i-1} (j_r + 1)\, q^{i-1-r} \right) \bmod K \\
&= \left(\sum_{r=0}^{i-m-1} (j_r + 1)\, q^{i-1-r} + \sum_{w=0}^{m-1} (j_{w+i-m} + 1)\, q^{m-1-w} \right) \bmod K \\
&= \left(\sum_{r=0}^{i-m-1} (j_r + 1)\, q^{i-1-r} + M[(m, n)] \right) \bmod K \\
&= \left(M[(m, n)] + q^m \sum_{r=0}^{i-m-1} (j_r + 1)\, q^{i-r-1-m} \right) \bmod K \\
&= (M[(m, n)] + q^m M[(i-m,\ j \bmod q^{i-m})]) \bmod K. \qquad (1)
\end{aligned}
$$

From Eq. (1), it is now clear that two arbitrary nodes of $S^{K,q}(m, n)$ are assigned to the same memory bank if and only if their two associated nodes in $S^{K,q}(0, 0)$ are colored the same. Since we have proved in Lemma 3 that the COLOR-SUBTREE mapping colors $S^{K,q}(0, 0)$ is without conflicts, any template instance is thus conflict-free.

Therefore, the COLOR-SUBTREE mapping is optimal because it guarantees conflict-free access using a number of memory banks equal to the template size. ∎

COROLLARY 1. *The COLOR-SUBTREE mapping for accessing templates $S^{K,q}$ of height $h = \lceil \log_q K \rceil$ of the q-ary tree T^q is also balanced.*

Proof. Given a complete q-ary T_v^q of height v, for $0 \leqslant h \leqslant v$, its lower-most $v-(v \bmod h)$ levels can be partitioned into $\lfloor (q^{v+1}-1)/(q^{h+1}-1) \rfloor$ complete disjoint instances of $S^{K,q}$, while the uppermost $v \bmod h$ levels belong to an incomplete instance of $S^{K,q}$. By Theorem 5, each template instance is conflict-free. Therefore, at least $\lfloor (q^{v+1}-1)/(q^{h+1}-1) \rfloor$ and no more than $\lceil (q^{v+1}-1)/(q^{h+1}-1) \rceil$, nodes are assigned to each memory bank. Hence, the ratio between the maximum and the minimum memory loads is asymptotically 1. ▮

By the same argument, accessing templates $S^{Z,q}$, where $Z > K$, of a complete q-ary tree colored by the procedure COLOR-SUBTREE ($S^{K,q}$, K, T^q), a total of $\lceil Z/K \rceil - 1$ conflicts occur which is again the minimum number of conflicts that can occur while accessing $S^{Z,q}$ using exactly K memory banks.

At this point it is important to note that our proposed COLOR-SUBTREE mapping is not *isotropic*. For example, both the nodes $(1, 0)$ and $(2, 3)$ of T_3^3 in Fig. 9 are assigned the color 1, but their children, from left to right, are colored with $\{0, 3, 2\}$ and $\{2, 3, 0\}$, respectively. However this mapping is *regular* as proved below.

THEOREM 6. *The COLOR-SUBTREE mapping belongs to the class of* regular *mappings.*

Proof. Consider two nodes A and B which are assigned the same color. Let the q-ary representations of positions in their levels be $a_{i-1}, ..., a_0$ and $b_{w-1}, ..., b_{i-1}, ..., b_0$, where $w > i$. That is, the assigned memory banks satisfy $M[A] = M'[A] \bmod K = M[B] = M'[B] \bmod K$. Then, it must hold for $M'[B] = \sum_{r=0}^{w-1} (b_r + 1)\, q^{w-1-r}$ and $M'[A] = \sum_{r=0}^{i-1} (a_r + 1)\, q^{i-1-r}$, implying $M'[B] - M'[A] = tK$ for some t.

Moreover, consider two nodes C and D at levels, respectively, $s+w$ and $s+i$, with the same relative position as A and B in the tree. Then, the q-ary representations of their positions in the levels are, respectively, $p_{s+i-1}, ..., p_i, a_{i-1}, ..., a_0$ and $p_{s+w-1}, ..., p_w, b_{w-1}, ..., b_{i-1}, ..., b_0$, where $p_{u+i-1} = p_{u+w-1}$, for $u = 1, ..., s$. Applying the procedure COLOR-SUBTREE, it holds that $M[D] = M'[D] \bmod K = \sum_{r=w}^{w+s-1} (p_r + 1)\ q^{s+w-1-r} + \sum_{r=0}^{w-1} (b_r + 1)\ q^{s+w-1-r} \bmod K$, and $M[C] = M'[C] \bmod K = \sum_{r=i}^{i+s-1} (p_r + 1)\, q^{s+i-1-r} + \sum_{r=0}^{i-1} (a_r + 1)\, q^{s+i-1-r} \bmod K$. Clearly, $M'[D] = M'[C] + q^s (M'[B] - M'[A]) = M'[C] + q^s tK$. Thus, $M'[D] \equiv M'[C] \bmod K$; that is $M[D] = M[C]$.

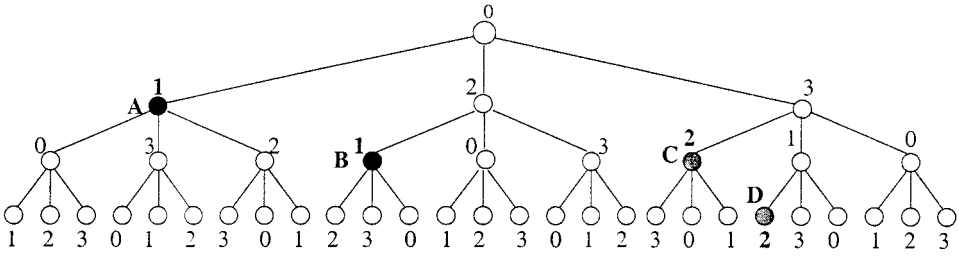

FIG. 9. The COLOR-SUBTREE optimal mapping to access $S^{4,3}$ of T_3^3. This mapping is regular as the pairs of nodes A, B and C, D show.

We can then conclude that the COLOR-SUBTREE mapping is regular since, given two tree nodes A and B mapped to the same memory bank, any tyro nodes C and D, which are in the same relative position to each other as A and B, are also mapped to the same bank. ▮

As an example, In Fig. 9, two pair of nodes (A, B) and (C, D), which are in the same relative position to each other as A and B, are assigned the colors 1 and 2, respectively.

Finally, it remains to prove that the COLOR-SUBTREE mapping is *flexible* in the sense that it optimally solves the conflict-free access problem for subtrees of any arity. Clearly, this mapping guarantees conflict-free access to any complete q-ary subtree of height $h = \lfloor \log_q K \rfloor$ as well as to any complete t-ary subtree $S^{(t^{h+1}-1)/(t-1), t}$ of the same height h, where $1 \leqslant t \leqslant q-1$.

THEOREM 7. *The COLOR-SUBTREE mapping is optimal for accessing complete t-ary subtree $S^{(t^{h+1}-1)/(t-1), t}$ of a complete q-ary tree, where $2 \leqslant t \leqslant q$.*

Proof. Observe that given any two nodes x and y of a complete q-ary tree of height h, since their lowest common ancestor is at a distance no more than h from both the nodes, there always exists a complete t-ary tree of height h, for any $2 \leqslant t \leqslant q$, that is an instance of the subtree $S^{(t^{h+1}-1)/(t-1), t}$, to which both x and y belong. Hence, at least K memory banks are required to design a conflict-free mapping for $S^{(t^{h+1}-1)/(t-1), t}$ where $2 \leqslant t \leqslant q$. This proves that the COLOR-SUBTREE mapping is optimal. ▮

Note that for $t = 1$, the template $S^{\lfloor \log_q K \rfloor, 1}$ degenerates into a path of length $\lfloor \log_q K \rfloor$. The previous result does not hold for $t = 1$ since for any two nodes of a complete q-ary tree $S^{K, q}$, there does not always exist a path to which they belong. Therefore, some nodes of $S^{K, q}$ can be colored the same, without generating conflicts while accessing paths. Indeed, as shown in Section 3, $\lfloor \log_q K \rfloor + 1$ memory banks are sufficient to guarantee conflict-free access to the ascending path template $S^{\lfloor \log_q K \rfloor, 1} = P^{\lfloor \log_q K \rfloor + 1}$. For access arbitrary paths in complete q-ary trees, interested readers can refer to [2].

5. BINOMIAL SUBTREE TEMPLATES

In this section, we design optimal mappings for conflict-free access to binomial subtrees. A preliminary version of these results was presented in [12].

DEFINITION 4. A *binomial tree*, B_n, of order n is recursively defined as an ordered tree such that

(i) B_0 is a single node,

(ii) B_n consists of two binomial trees, B_{n-1}, linked together such that the root of one is the leftmost child of the root of the other.

The tree B_n has height n and its root $r = (0, 0)$ has n children which will be denoted from right to left as $r_0, r_1, ..., r_{n-1}$. Thus the degree of the root is $\delta(r) = n$. By definition, B_n has a total of 2^n nodes and $\binom{n}{i}$ nodes at level i.

The binomial tree of order i and rooted at the $(i+1)$th child r_i of the root r will be denoted as $B_i^{r_i}$. Moreover, given a node x with degree $\delta(x) \geqslant i$, let B_i^x denote a binomial tree of order i contained in the binomial subtree $B_{\delta(x)}^x$ which is rooted at node x.

We are interested in accessing the following two templates in binomial trees:

- *binomial subtrees* $B^{K,t}$ of size $K=2^t$ and order t,
- *oriented binomial subtrees* $O^{\Delta,K,t}$ of size K and order t, as defined by the tuple Δ (see Section 5.1).

Note that for binomial trees, the size $K=2^t$ is uniquely determined by the order t of the tree. Although K is redundant, we keep it in the notation for $B^{K,t}$ to be consistent with the notation $S^{K,q}$ used for complete q-ary subtrees. Similarly, $B^{K,t}(x)$ will denote the template instance rooted at node x.

We propose the mapping COLOR-BINOMIAL that uses $\mathtt{M}=\sum_{i=0}^{t-1}\binom{n}{i}+\binom{n-1}{t-1}$ colors for conflict-free access to any occurrence of the template $B^{K,t}$ of the binomial tree B_n. The mapping, illustrated in Fig. 10, colors the binomial tree level-wise in such a way that it somehow reminds us of the coloring of complete q-ary trees proposed by Das and Sarkar in [7].

In the following, $B^{K,t}(i,j)$ denotes an instance of the binomial subtree template, rooted at node (i,j) of B_n. Similarly, $(i-1, p(j))$ and $(i-2, p(p(j)))$, respectively, denote the parent and the grandparent of the node (i,j).

PROCEDURE *COLOR-BINOMIAL*$(B^{2^t,t}, \mathtt{M}, B_n)$

- **Step 1**
  Let us partition the color-set $\{0, 1, ..., \mathtt{M}-1\}$, where $\mathtt{M}=\sum_{i=0}^{t-1}\binom{n}{i}+\binom{n-1}{t-1}$, into two subsets: $U=\{0, 1, ..., \sum_{i=0}^{t-1}\binom{n}{i}-1\}$ and $Z=\{\sum_{i=0}^{t-1}\binom{n}{i}, ..., \sum_{i=0}^{t-1}\binom{n}{i}+\binom{n-1}{t-1}-1\}$.

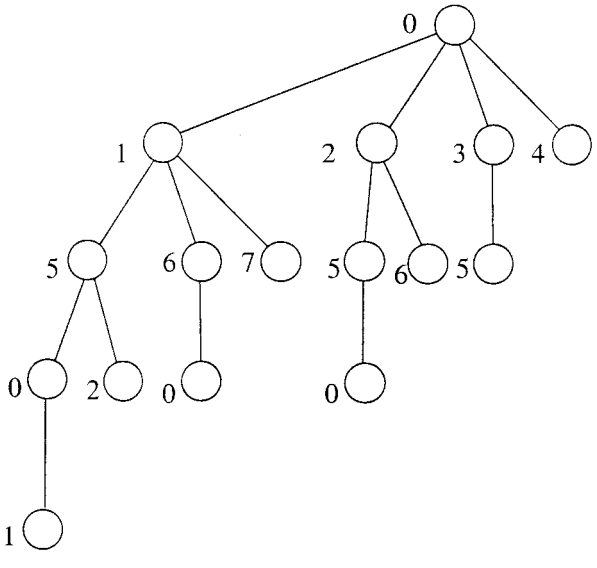

FIG. 10. A binomial tree B_4 colored according to the procedure COLOR-BINOMIAL to access optimally binomial subtree template $B^{4,2}$ using $M=8$ colors.

- Assign the colors of the subset U to the $|U|$ nodes of the uppermost t levels of the binomial tree B_n.
- Assign the colors of the subset Z as follows to the nodes at level t (in left-to-right order) of the binomial tree B_n.

  — **For** $n-1 \geqslant i \geqslant t-1$, **do**
  Assign the $\binom{i}{t-1}$ nodes at level t of B_n which form the level $t-1$ of the subtree $B_i^{r_i}$, to the colors $\{\sum_{i=0}^{t-1}\binom{n}{i}, ..., \sum_{i=0}^{t-1}\binom{n}{i}+\binom{i}{t-1}-1\}$ of the subset Z.

- **Step 2**
  The rest of the tree is colored level-by-level as follows.

  — **For** each level l, $2 \leqslant l \leqslant n-t+1$, **do**

  For each node (l, i), $0 \leqslant i \leqslant \binom{n}{l}-1$, **do**

Assign the nodes at level $t-1$ of the binomial subtree rooted at (l, i) to the same colors used to color the nodes that belong to the uppermost t levels of the subtree rooted at the grandparent $(l-2, p(p(i)))$ of the node (l, i), but that do not belong to the uppermost $t-1$ levels of the tree rooted at the parent $(l-1, p(i))$ of (l, i). See Fig. 11 for an illustration.

Next we show that the number of colors used by the COLOR-BINOMIAL mapping is the minimum number of colors required to achieve conflict-free access to binomial subtrees.

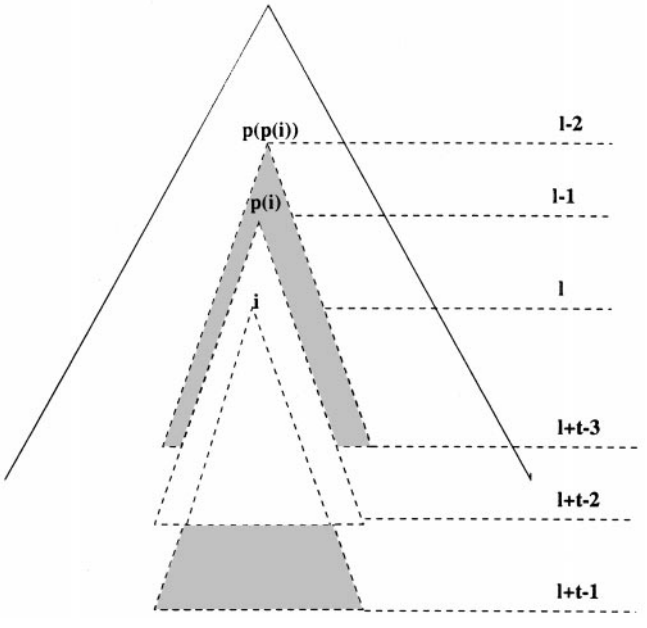

FIG. 11. Illustrating Stem 2 of the mapping *COLOR-BINOMIAL*($B^{2^l, t}$, $\mathbb{M}$, B_n)

LEMMA 4. *At least* $\mathbb{M} = \sum_{i=0}^{t-1} \binom{n}{i} + \binom{n-1}{t-1}$ *colors are required for conflict-free access to binomial subtrees* $B^{2^t, t}$ *of* B_n, *for* $t \leqslant n$.

Proof. Given two nodes x and y in the uppermost t levels of the binomial tree B_n, we show that there always exists a template instance $B^{2^t, t}(r)$ rooted at the root r of B_n that contains both x and y. However, there is no restriction on the choice of nodes x and y. In fact, they can belong to the same path or one of them can be the root r itself.

Let $l(x)$ and $l(y)$ denote the levels of x and y in B_n^r. Moreover, let r_x and r_y (not necessarily distinct) be the $[l(x)-1]$th ancestor and the $[l(y)-1]$th ancestor of x and y, respectively. In other words, r_x and r_y are the children of the root r on the paths toward x and y. Their degrees satisfy the relations $\delta(r_x) \geqslant l(x)-1$ and $\delta(r_y) \geqslant l(y)-1$. Without loss of generality, suppose that $\delta(r_y) \geqslant \delta(r_x)$. Then, the instance $B^{2^t, t}(r)$ of the template $B^{2^t, t}$ containing nodes x and y (not necessarily distinct) is recursively constructed as follows:

- $t-1 \geqslant \delta(r_x) \geqslant \delta(r_y) \geqslant 0$ (terminal case): $B^{2^t, t}(r)$ consists of the root node r and the subtrees rooted at its rightmost t children;
- $\delta(r_y) > t-1 \geqslant \delta(r_x) \geqslant 0$: The instance $B^{2^t, t}(r)$ consists of r, the subtrees rooted at the rightmost $t-1$ children of r, and an instance $B^{2^{t-1}, t-1}(y)$ constructed recursively;
- $\delta(r_y) \geqslant \delta(r_x) > t-1$: $B^{2^t, t}(r)$ consists of r, the subtrees rooted at the $t-2$ rightmost children of r, and the constructed instances $B^{2^{t-2}, t-2}(x)$ and $B^{2^{t-1}, t-1}(y)$.

Thus, all the $\mathbb{M} = \sum_{i=0}^{t-1} \binom{n}{i} + \binom{n-1}{t-1}$ nodes of the uppermost t levels of the binomial tree B_n^r must be assigned distinct colors or memory banks. ∎

THEOREM 8. *The mapping* COLOR-BINOMIAL *is optimal for accessing binomial subtree templates* $B^{2^t, t}$.

Proof. The proposed mapping is conflict-free since the lowest common ancestor of two nodes x and y assigned to the same memory bank by the procedure *COLOR-BINOMIAL*$(B^{2^t, t}, \mathbb{M}, B_n)$ is at a distance $\geqslant t+1$ from at least one of them. Therefore x and y cannot belong to the same template instance, and the optimality follows from Lemma 4. ∎

Note that $2^{s-t}-1$ conflicts occur while accessing the template $B^{2^s, s}$, where $s > t$, of a binomial tree colored by the procedure COLOR-BINOMIAL $(B^{2^t, t}, \mathbb{M}, B_n)$.

Although this mapping is optimal in terms of the number of memory banks used, it requires a huge number of memory banks. Moreover, it is not direct. To overcome these drawbacks we introduce below the concept of *oriented binomial subtree* templates.

5.1. Oriented Binomial Subtrees

Let us introduce the notion of an *oriented t-tuple* with respect to a binomial tree B_n and then recursively define an oriented binomial subtree template.

DEFINITION 5. An *oriented t-tuple*, $\Delta = (i_{t-1}, ..., t_0)$, is a sorted sequence of t integers in the closed interval $[0, n-1]$ such that:

- $i_j > i_{j-1}$ for $1 \leqslant j \leqslant t-1$, and
- $\forall z \in [0, ..., t-1]$, $\exists! i_j$: $(i_j \bmod t) = z$, where $0 \leqslant j \leqslant t-1$.

Moreover, $\Delta_j = (i_{j-1}, ..., i_0)$ denotes the rightmost j indices in $\Delta = \Delta_t$.

Recalling that ξ_i denote the $(i+1)$th child (counting from left to right, starting from 0) of a node ξ, we define:

DEFINITION 6. An instance $O^{\Delta, 2^t, t}(\xi)$ of the *oriented template* $O^{\Delta, 2^t, t}$, rooted at node ξ and based on the *t-tuple* Δ, is a binomial subtree of order t such that:

- The root ξ has t children $\{\xi_{i_t}, \xi_{i_{t-1}}, ..., \xi_{i_0} \mid i_j \in \Delta\}$;
- Recursively, the binomial tree rooted at ξ_{i_j}, for $t \geqslant j \geqslant 0$, is an instance $O^{\Delta, 2^j, j}(\xi_{i_j})$, rooted at ξ_{i_j}, of the oriented binomial tree $O^{\Delta, 2^j, j}$ based on the j-tuple Δ_j.

Figure 12 shows in bold lines an instance of the oriented template $O^{\Delta, 8, 3}$, rooted at the root $(0, 0)$, based on the 3-tuple, $\Delta = (3, 2, 1)$.

Before presenting the mapping COLOR-ORIENTED-BINOMIAL, let us assign *binary addresses* to the nodes of a binomial tree following the approach in [16].

Assigning the address $(0, 0, ..., 0)$ to the root, let p be such that $c_p = 1$ and $c_m = 0\ \forall m \in \{p+1, p+2, ..., n-1\}$, and let $p = -1$ if $c = 0$. Then, the children of the nodes $i = (0, 0, ..., 0, c_i = 1, b_{i-1}, ..., b_0)$, where $b_j = \{0, 1\}$, $0 \leqslant j \leqslant i-1$, are the nodes $(0, 0, ..., c_m = 1, 0, ..., c_i = 1, b_{i-1}, ..., b_0)$, for $m \in \{i+1, i+2, ..., n-1\}$. In other words, the children of a node whose leading 1 is at position i are the nodes whose binary addresses are generated by complementing one by one the leading zeros of the binary encoding of the address of their parents (see Fig. 12).

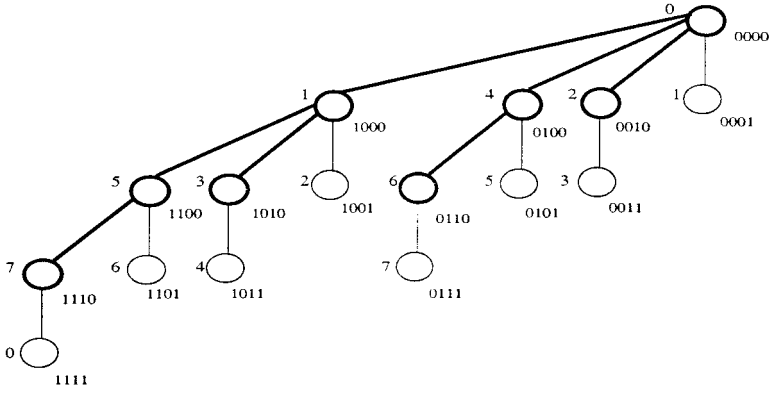

FIG. 12. Conflict-free access to the occurrences of the oriented subtree template $O^{\Delta, 8, 3}$ of B_4, where $\Delta = (3, 2, 1)$. On the left (resp., right) of each node it is shown its color (resp., binary address).

The mapping COLOR-ORIENTED-BINOMIAL derives the color $M[x]$ assigned to the node x from its binary address.

PROCEDURE *COLOR-ORIENTED-BINOMIAL*($O^{\Delta, 2^t, t}$, 2^t)

- $M[x] = (\sum_{i=0}^{n-1} b_i 2^{i \bmod t}) \bmod 2^t$, for all nodes $(b_{n-1}, ..., b_0) \in B_n$.

THEOREM 9. *The* COLOR-ORIENTED-BINOMIAL *mapping is optimal for accessing the oriented binomial subtree template* $O^{\Delta, 2^t, t}$ *for any t-tuple* Δ.

Proof. For each node $\xi \in B_n$ such that $\delta(\xi) \geqslant t$, and for any t-tuple, Δ, the addresses of the subset of nodes belonging to the oriented template $O^{\Delta, 2^t, t}(\xi)$ span the 2^t possible configurations of the dimensions b_i, where $i \in \Delta$. Therefore, all instances of the family of oriented binomial trees $O^{\Delta, 2^t, t}$ can be accessed in a conflict-free manner using a number of memory banks equal to the template size. Hence the optimality. ∎

This mapping is direct by definition. It is also balanced because any tree B_n can be decomposed into 2^{n-t} disjoint subtrees B_t, and each of them is conflict-free. Hence, the load on each memory bank is 2^{n-t}, and the ratio between the minimum and the maximum memory loads is exactly 1.

6. CONCLUSIONS

This paper dealt with load-balanced mappings of tree structures in a multibank memory system which allows conflict-free access to templates of interest. In particular, we proposed efficient schemes for accessing paths and subtrees of q-ary trees and subtrees of binomial trees. For our work on a universal scheme for accessing composite templates, refer to [1].

Our mapping techniques are generic and independent of the specific topological interconnection among processors or the communication mechanism between the processors and memory. However, one can take advantage of a specific architecture by fine tuning our framework.

Since our data storage schemes are for fine-grain memory access, as such they may not work well if we require template access to data structures stored in secondary memories. Designing such schemes is a topic of our future research. We also plan to extend the template data access problem to accommodate the effects of nonuniform memory access patterns and interprocessor communication topologies (specifically, different locations have different communication costs) as well as the effects of granularity in memory access (e.g., multilevel memory hierarchy). In other words, we intend to study the interplay between locality preserving (i.e., reducing communication overhead) vs conflict-freeness (i.e., spreading data as much as possible to achieve high memory-bandwidth). Finally, an interesting future work is to investigate how storing trees in this manner can speed up a broad class of parallel algorithms on trees, similar to the cost-optimal parallel priority queue (heap) implementation based on path templates [10].

ACKNOWLEDGMENTS

We are grateful to the anonymous referees for providing valuable suggestions and helpful comments which improved the quality and organization of the manuscript considerably. We also thank the Editor for handling our paper in a timely manner.

REFERENCES

1. V. Auletta, S. K. Das, A. De Vivo, M. C. Pinotti, and V. Scarano, Toward a universal mapping algorithm for accessing trees in parallel memory systems, *in* "Proceedings of IEEE Int'l Parallel Processing Symposium, Orlando, Apr. 1998," pp. 447–4548.
2. A. A. Bertossi and M. C. Pinotti, "Mappings for Conflict-Free Access of Paths in Bidimensional Arrays, Circular Lists, and Complete Trees," Nota Interna IEI-CNR B4-020, October 1999.
3. G. E. Blelloch, P. B. Gibbons, Y. Mattias, and M. Zagha, Accounting for memory bank contention and delay in high-bandwidth multiprocessors, *IEEE Trans. Parallel Distrib. Systems* (Sept. 1997), 943–958.
4. M. Balakrishnan, R. Jain, and C. S. Raghavendra, On array storage for conflict-free memory access for parallel processors, *in* "Proc. Int.'l Conf. on Parallel Processing, Aug. 1988," pp. 103–107.
5. P. Budnik and D. J. Kuck, The organization and use of parallel memories, *IEEE Trans. Comput.* (1971), 1566–1569.
6. R. Creutzburg, Parallel optimal subtree access with recursively linear memory functions, *in* "Proc. of Int'l Workshop on Parallel Processing by Cellular Automata and Arrays, Berlin, Sept. 1986" (T. Legendi, D. Parkinson, R. Vollmar, and G. Wolf, Eds.), pp. 203–209.
7. S. K. Das and F. Sarkar, Conflict-free data access of arrays and trees in parallel memory systems, *in* "Proc. of the Sixth IEEE Symposium on Parallel and Distributed Processing, Dallas, Oct. 1994," pp. 377–384.
8. S. K. Das, F. Sarkar, and M. C. Pinotti, Conflict-free access of trees in parallel memory systems and its generalization with applications to distributed heap implementation, *in* "Proc. Int'l Conf. on Parallel Processing, Oconomowoc, WI, Aug. 1995," Vol. III, pp. 164–167.
9. S. K. Das, F. Sarkar, and M. C. Pinotti, Distributed priority queues on hypercube architectures, *in* "16th IEEE Int'l Conf. on Distributed Computing Systems, Hong Kong, May 1996," pp. 620–627.
10. S. K. Das, F. Sarkar, and M. C. Pinotti, Parallel priority queues in distributed memory hypercubes, *IEEE Trans. Parallel Distrib. Systems* (June 1996), 555–564.
11. S. K. Das and M. C. Pinotti, $O(\log \log N)$ time algorithms for hamiltonian-suffix and min-max-pair heap operations on the hypercube, *J. Parallel Distrib. Comput.* **52** (Feb. 1998), 200–211.
12. S. K. Das and M. C. Pinotti, Conflict-free access to templates of trees and hypercubes, *in* "Proc. 3rd Annu. Int'l Conf. on Computing and Combinatorics (COCOON), Shanghai, Aug. 1997," pp. 1–10.
13. S. K. Das and M. C. Pinotti, Load balanced mapping of data structures in parallel memory modules for fast and conflict-free templates access, *in* "Proc. 5th Int. Workshop on Algorithms and Data Structures (WADS'97), Halifax, Canada, Aug. 1997" (Dehne, Rau-Chaplin, Sack, and Tamassia, Eds.), Lecture Notes in Computer Science, Vol. 1272, pp. 272–281, Springer-Verlag, Berlin/New York, 1997.
14. J. Denes and A. D. Keedwell, "Latin Squares and Their Applications," Academic Press, New York, 1974.
15. M. Gössel and B. Rebel, Memories for parallel subtree access, *in* "Proc. of Int'l Workshop on Parallel Algorithms and Architectures" (A. Albrecht, H. Jung, and K. Mehlhorn, Eds.), Lecture Notes in Computer Science, Vol. 299, pp. 120–130, Springer-Verlag, Berlin/New York, 1987.
16. S. L. Johnsson and C. Ho, Optimum broadcasting and personalized communication in hypercubes, *IEEE Trans. Comput.* (Sept. 1989), 1249–1268.
17. K. Kim and V. K. Prasanna, Latin squares for parallel array access, *IEEE Trans. Parallel Distrib. Systems* (April 1993), 361–370.

18. M. C. Pinotti, S. K. Das, and F. Sarkar, Conflict-free template access in k-ary and binomial trees, *in* "Proceedings of ACM-Int'l Conference on Supercomputing, Wien, Austria, July 1997," pp. 237–244.

19. H. A. G. Wijshoff, Storing trees into parallel memories, *in* "Parallel Computing 85" (Feilmeier, Ed.), pp. 253–261, Elsevier Science, Amsterdam, 1986.

SAJAL K. DAS received the B.Tech. in 1983 from Calcutta University, the M.S. in 1984 from the Indian Institute of Science at Bangalore, and the Ph.D. in 1988 from the University of Central Florida at Orlando, all in computer science. Currently he is a full professor of computer science and engineering and also the Founding Director of the Center for Research in Wireless Mobility and Networking (CReWMaN) at the University of Texas at Arlington. From 1988–1999, he was a member of the faculty of computer science at the University of North Texas, Denton, where he founded the Center for Research in Wireless Computing (CReW) in 1997. During 1995–1997, he was the Director of the Center for Research in Parallel and Distributed Computing (CRPDC) at UNT. Das is a recipient of the Honor Professor Award from UNT in 1991 and 1997 for best teaching and scholarly research, and UNT's Developing Scholars Award in 1996 for outstanding research. He has been a visiting scientist at the National Research Council in Pisa, Italy, and the Slovak Academy of Sciences in Bratislava, and also a visiting professor at the Indian Statistical Institute, Calcutta. His current research interests include the design and analysis of parallel algorithms and data structures, multiprocessor interconnection networks, resource management in wireless networks, mobile computing, QoS provisioning in wireless multimedia, and performance modeling and simulation. He has published over 150 research papers in these areas and directed several funded projects. He was a corecipient of the Best Paper Awards at the ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'99) and ACM/IEEE International Workshop on Parallel and Distributed Simulation (PADS'97). He serves on the Editorial Boards of the *Journal of Parallel and Distributed Computing* (as subject area editor of mobile computing), *Parallel Processing Letters*, and the *Journal of Parallel Algorithms and Applications*. In 1997–1999, he was the Coeditor of the IEEE TCPP Newsletter. He has guest-edited special issues of the *Journal of Parallel and Distributed Computing* on parallel and distributed data structures, wireless and mobile communications and computing, and self-stabilizing distributed systems. He has served on the program committees of numerous conferences, including IEEE IPPS, ICPP, IEEE SPDP, IEEE INFOCOM, and ACM/IEEE MobiCom; he is the General Vice-Chair of MobiCom-2000, General Co-Chair of the Fourth IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS'98), General Chair of the Third ACM International Workshop on Wireless Mobile Multimedia (WoWMoM-2000), General Vice-Chair of the IEEE International Conference on High Performance Computing (HiPC2000), Program Vice Chair of HiPC'99, and the Founding Program Chair of WoWMoM'98 and WoWMoM'99. He also serves on the ACM SIGMOBILE and IEEE TCPP Executive Committees. He is a member of the IEEE and ACM.

M. CRISTINA PINOTTI received the Ph.D. *cum laude* in computer science from the University of Pisa, Italy, in 1986. Since 1987 she has been a researcher with the Consiglio Nazionale delle Ricerche at the Instituto di Elaborazione della Informazione, Pisa. In 1994 and 1995 she was a research associate in the Department of Computer Science, University of North Texas, Denton, Texas. Her research interests include the design and analysis of parallel algorithms, parallel data structures, multiprocessor interconnection networks, computer arithmetic, residue number systems, and VLSI layouts. Pinotti is a member of the IEEE Computer Society.