*M*appings for Conflict-Free Access of Paths in Bidimensional Arrays Circular Lists, and Complete Trees

Alan A.Bertossi, M.Cristina Pinotti

B4-20
ott-1999

Mappings for Conflict-Free Access of Paths in Bidimensional Arrays, Circular Lists, and Complete Trees *

Alan A. Bertossi [†] and M. Cristina Pinotti
Istituto di Elaborazione dell' Informazione
National Council of Research
Pisa, ITALY
E-mail: {`bertossi@science.unitn.it`, `pinotti@iei.pi.cnr.it`}

Abstract

Since the divergence between the processor speed and the memory access rate is progressively increasing, an efficient partition of the main memory into multibanks is useful to improve the overall system performance. The effectiveness of the multibank partition can be degraded by *memory conflicts*, that occur when there are many references to the same memory bank while accessing the same memory pattern. Therefore, mapping schemes are needed to distribute data in such a way that data can be retrieved via regular patterns without conflicts. In this paper, the problem of conflict-free access of *arbitrary* paths in bidimensional arrays, circular lists and complete trees is considered for the first time and reduced to variants of graph-coloring problems. Balanced and fast mappings are proposed which require an optimal number of colors (i.e., memory banks). The solution for bidimensional arrays is based on a combinatorial object similar to a Latin Square. The functions that map an array node or a circular list node to a memory bank can be calculated in constant time. As for complete trees, the mapping of a tree node to a memory bank takes time that grows logarithmically with the number of nodes of the tree.

Key Words: Bidimensional array, circular list, complete tree, conflict-free access, mapping scheme, multibank memory system, path template.

*This work has been supported by the "Provincia Autonoma di Trento" under a research grant.
†On leave from University of Trento, Department of Mathematics, Trento, Italy

Running Title: Conflict-Free Access of Paths

Address for Correspondence:

M.C. Pinotti
IEI-CNR
Via S. Maria, 46
56127 Pisa
ITALY
E-mail: `pinotti@iei.pi.cnr.it`

List of Footnotes:

This work has been supported by the "Provincia Autonoma di Trento" under a research grant.

1 Introduction

In recent years, the traditional divergence between the processor speed and the memory access rate is progressively increasing. Thus, an efficient organization of the main memory is important to achieve high-speed computations. For this purpose, the main memory can be equipped with *cache* memories – which have about the same cycle time as the processors – or can be partitioned into *multibanks*. Since the cost of the cache memory is high and its size is limited, the multibank partition has mostly been adopted, especially in shared-memory multiprocessors [3]. However, the effectiveness of such a memory partition can be limited by *memory conflicts*, that occur when there are many references to the same memory bank while accessing the same memory pattern. To exploit to the fullest extent the performance of the multibank partition, mapping schemes can be employed that avoid or minimize the memory conflicts [15]. Since it is hard to find universal mappings – mappings that minimize conflicts for arbitrary memory access patterns – several specialized mappings, designed for accessing regular patterns in specific data structures, have been proposed in the literature (see [12, 2] for a complete list of references).

In particular, for bidimensional arrays, Budnik and Kuck [7], Balakrishnan et al. [4], Kim and Prasanna [12], and Das and Sarkar [8] studied mappings that provide conflict-free access to rows, columns, positive and negative diagonals, subarrays, and distributed subarrays. The techniques used range from *Latin squares* to *Perfect Latin squares*, from *linear mappings* to *quasi-groups* [11]. Subsequently, mappings for other data structures like complete trees and binomial trees have been devised. In particular, mappings that provide conflict-free access to complete subtrees, root-to-leaves paths, sublevels, and composite patterns obtained by their combination, have been investigated in [8, 9, 1, 10, 14]. The mapping schemes proposed in those papers are *optimal*, i.e., they use as few memory modules as possible; *balanced*, i.e., the nodes of data structures are distributed as evenly as possible among the banks; *fast*, i.e., the bank address to which a node is assigned is computed quickly with no knowledge of the entire structure mapping; and *flexible*, i.e., they can be used for templates of different size.

In the present paper, optimal, balanced and fast mappings are designed for conflict-free access of paths in bidimensional arrays, circular lists, and complete trees. With respect to the above mentioned papers, paths in bidimensional arrays and circular lists are dealt with for the first time. Moreover, access to any (not only to root-to-leaves) paths in complete trees is provided. The remainder of this paper is organized as follows. In Section 2, the conflict-free access problem is formally stated. In Section 3, the problem of accessing paths in bidimensional arrays is solved. The proposed solution is a variant of a graph-coloring, which requires an optimal number of colors and is achieved using a combinatorial object similar to a Latin Square. As a byproduct, the memory bank to which an array node is assigned is computed in constant time. In Section 4, the problem of accessing paths in circular lists is optimally solved and the function that maps a circular list node to a memory bank can be calculated in constant time. In Section 5, the same problem on complete trees is also optimally solved via a variant of a graph-coloring problem. The time needed to assign a tree node to a memory bank grows logarithmically with the number of nodes of the tree. Conclusions are offered in Section 6.

2 Conflict-Free Access

When storing a data structure D, represented in general by a graph, on a memory system consisting of N memory banks, a desirable issue is to map any subset of N arbitrary nodes of D to all the N different banks. This problem can be viewed as a *coloring* problem where the distribution of nodes of D among the banks is done by coloring the nodes with a color from the set $\{0, 1, 2, \ldots, N-1\}$. Since it is hard to solve the problem in general, access of regular patterns, called *templates*, in special data structures – like bidimensional arrays, circular lists, and complete trees – are considered hereafter.

A *template* T is a connected subgraph of D. The occurrences $\{T_1, T_2, \ldots, T_m\}$ of T in D are the *template instances*. For example, if D is a complete binary tree, then a path of length k can be a template, and all the paths of length k in D are the template instances.

After coloring D, a *conflict* occurs if two nodes of a template instance are assigned to the same memory bank, i.e., they get the same color. An access to a template instance T_i results in c conflicts if $c+1$ nodes of T_i belong to the same memory bank.

Given a memory system with N banks and a template T, the goal is to find a *memory mapping* $U : D \to N$ that colors the nodes of D in such a way that the number of conflicts for accessing any instance of T is minimal. In fact, the *cost* for T_i colored according to U, $Cost_U(D, T_i, N)$, is defined as the number of conflicts for accessing T_i. The template instance of T with the highest cost determines the overall cost of the mapping U. That is,

$$Cost_U(D,T,N) \stackrel{\text{def}}{=} \max_{T_i \in T} Cost_U(D,T_i,N).$$

A mapping U is *conflict-free* for T if $Cost_U(D,T,N) = 0$.

Among desirable properties for a conflict-free mapping, a mapping should be balanced, fast, and optimal. A mapping U is termed *balanced* if it evenly distributes the nodes of the data structure among the N memory banks. For a balanced mapping, the *memory load* is almost the same in all the banks. A mapping U will be called *fast* if the color of each node can be computed quickly (possibly in constant time) without knowledge of the coloring of the entire data structure. Among all possible conflict-free mappings for a given template of a data structure, the more interesting ones are those that use the minimum possible number of memory banks. These mappings are called *optimal.* It is worth to note that not only the template size but also the overlapping of template instances in the data structure determine a lower bound on the number of memory banks necessary to guarantee a conflict-free access scheme. This fact will be more convincing by the argument below for accessing paths in D.

Let $G_D = (V, E)$ be the graph representing the data structure D. The template P_k is a *path* of length k in D. The template instance $P_k[x,y]$ is the path of length k between two vertices x and y in V, that is, the sequence $x = v_1, v_2, \ldots, v_{k+1} = y$ of vertices such that $(v_h, v_{h+1}) \in E$ for $h = 1, 2, \ldots k$.

The conflicts can be eliminated on $P_k[x,y]$ if $v_1, v_2, \ldots, v_{k+1}$ are assigned to all different memory banks. The conflict-free access to P_k can be reduced to a classical coloring problem on the associated graph G_{DP_k} obtained as follows. The vertex set of G_{DP_k} is the same as the vertex set of G_D, while the edge (r,s) belongs to the edge set of G_{DP_k} iff the distance d_{rs} between the vertices r and s in G_D satisfies $d_{rs} \leq k$, where the distance is the length of the shortest path between r and s. Now, colors must be assigned to the vertices of G_{DP_k} so that every pair of vertices connected by an edge is assigned a couple of different colors and the minimum number of colors is used. Hence, the role of *maximum clique* in G_{DP_k} is apparent for deriving lower bounds on the conflict-free access on paths. A *clique* K for G_{DP_k} is a subset of the vertices of G_{DP_k} such that for each pair of vertices in K there is an edge. By well-known graph theoretical results, a clique of size n in the associated graph G_{DP_k} implies that at least n different colors are needed to color G_{DP_k}. In other words, the size of the largest clique in G_{DP_k} is a lower bound for the number of memory banks required to access paths of length k in D without conflicts.

On the other hand, the conflict-free access to P_k on G_D is equivalent to color the nodes of G_D in such a way that any two nodes which are at distance k or less apart have assigned different colors. Unfortunately, this latter coloring problem is NP-complete [13] for general graphs. This justifies the investigation either for *good* heuristics for general graphs or optimal algorithms for special classes of graphs. In the next three sections, optimal mappings for bidimensional arrays, circular lists and complete binary trees will be derived for conflict-free accessing P_k.

3 Accessing Paths in Bidimensional Arrays

Let a bidimensional array A be the data structure D to be mapped into the multibank memory system. An array $r \times c$ has r rows and c columns, indexed respectively from 0 to $r-1$ (from top to bottom) and from 0 to $c-1$ (from left to right), with r and c both greater than 1.

The graph $G_A = (V, E)$ representing A is a mesh, whose vertices correspond to the elements of A and whose arcs correspond to any pair of adjacent elements of A on the same row or on the same column. For the sake of simplicity, A will be used instead of G_A since there is no ambiguity. Thus, a generic node x of A will be denoted by $x = (i, j)$, where i is its row index and j is its column index.

Lemma 1 *At least $M = \left\lceil \frac{(k+1)^2}{2} \right\rceil$ memory banks are required for conflict-free accessing P_k in A.*

Proof Consider a generic node $x = (i, j)$ of A, and its opposite node at distance k on the same column, i.e., $y = (i-k, j)$. All the nodes of A at distance k or less from both x and y are mutually at distance k or less, as shown in Figure 1. Therefore, in the associated graph G_{AP_k}, they form a clique, and they must be assigned to different colors. In details, such a clique, denoted as $K_A(x, k)$, is defined as follows:

$$K_A(x,k) = \left\{(i-k+t, j-t), \ldots, (i-k+t, j+t) \ : \ 0 \le t \le \left\lfloor \tfrac{k}{2} \right\rfloor \right\} \ \cup$$
$$\left\{ \left(i - \left\lceil \tfrac{k}{2} \right\rceil + t, j - \left\lceil \tfrac{k}{2} \right\rceil + t\right), \ldots, \left(i - \left\lceil \tfrac{k}{2} \right\rceil + t, j + \left\lceil \tfrac{k}{2} \right\rceil - t\right) \ : \ 1 \le t \le \left\lceil \tfrac{k}{2} \right\rceil \right\}$$

Summing up over t, the size of the clique results to be

$$|K_A(x,k)| = \sum_{t=0}^{\lfloor \frac{k}{2} \rfloor} (2t+1) + \sum_{t=1}^{\lceil \frac{k}{2} \rceil} \left(2\left(\left\lceil \frac{k}{2} \right\rceil - t \right) + 1 \right) = \left\lceil \frac{(k+1)^2}{2} \right\rceil .$$

Hence, at least $M = |K_A(x,k)| = \left\lceil \frac{(k+1)^2}{2} \right\rceil$ colors are required. □

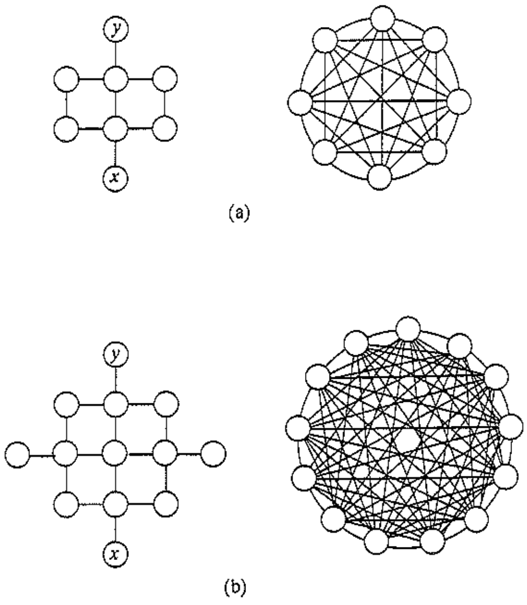

Figure 1: A subset $K_A(x,k)$ of nodes of A that forms a clique in G_{AP_k}: (a) $k=3$, (b) $k=4$.

Below, a conflict-free mapping is given to color all the nodes of an array A using as few colors as in Lemma 1. Therefore, the mapping is optimal. From now on, the color assigned to node x is denoted by $\gamma(x)$.

```
**Algorithm Array-Coloring** $(A,k)$;
    • Set $M = \left\lceil \frac{(k+1)^2}{2} \right\rceil$ and $\Delta = \begin{cases} k+1 & \text{if } k \text{ is even} \\ k & \text{if } k \text{ is odd} \end{cases}$
    • Assign to each node $x=(i,j) \in A$ the color $\gamma(x) = (i\Delta + j) \bmod M$.
```

Intuitively, the above algorithm first covers A with a tessellation of basic sub-arrays of size $M \times M$. Each basic sub-array S is colored in a Latin Square fashion as follows:

- the colors in the first row of S appear from left-to-right in the sequence $0, 1, 2, \ldots, M-1$;
- the color sequence for a generic row is obtained from the sequence at the previous row by a Δ left-cyclic shift.

For $k=3$, the coloring of A, decomposed into 6 basic sub-arrays of size $M \times M$, is illustrated in Figure 2.

Theorem 1 *The Array-Coloring mapping is optimal, fast, and balanced.*

Proof To prove optimality, it must be shown that the mapping is conflict-free and that the minimum number of colors is used.

0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2
6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5
1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0
4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3
7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6
2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1
5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4
0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2
6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5
1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0
4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3
7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6
2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1
5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4

Figure 2: An array A of size 16×24 with a tessellation of 6 sub-arrays of size 8×8 colored by the Array-Coloring algorithm to conflict-free access P_3.

Consider a generic node $x = (g, f)$ of A and the associated clique $K_A(x, k)$, defined in Lemma 1. In order to prove that the mapping is conflict-free, one only needs to show that all the nodes of $K_A(x, k)$, which are mutually at distance no more than k, are assigned by the Array-Coloring algorithm to different colors. Formally, consider an arbitrary pair of nodes $w = (i, j)$ and $z = (h, \ell)$ belonging to $K_A(x, k)$, such that $i - h \geq 0$ (if $i - h < 0$, the roles of w and z could be swapped). Then the mapping is conflict-free if the Array-Coloring algorithm guarantees that the colors $\gamma(w)$ and $\gamma(z)$ are different. Moreover, let $\sigma(w, z) \equiv (\gamma(w) - \gamma(z)) \bmod M \equiv (i\Delta + j) - (h\Delta + \ell) \bmod M$ be the difference between the two colors assigned to w and z. Then, the mapping is conflict-free if the following two conditions simultaneously hold:

$$\begin{cases} \sigma(w,z) \not\equiv 0 \bmod M, \\ \\ |i-h| + |j-\ell| \leq k. \end{cases} \tag{1}$$

In order to show that the conditions in (1) hold for any pair of nodes of $K_A(x, k)$, the two cases k even and k odd must be distinguished.

When k is even, one has that $M = \left\lceil \frac{(k+1)^2}{2} \right\rceil = \frac{k^2+2k+2}{2} = \frac{k}{2}(k+1) + \frac{k}{2} + 1$ and $\Delta = k + 1$. Observe that $\sigma(w, z) < (i - h)\Delta + |j - \ell| < k\Delta + k = k(k + 1) + k < 2M$ and $\sigma(w, z) > -|j - \ell| > -M$. Then, the congruence $\sigma(w, z) \not\equiv 0 \bmod M$ is equivalent to $\sigma(w, z) \neq 0$ and $\sigma(w, z) \neq M$.

Clearly, $\sigma(w, z) = 0$ iff $(i - h)(k + 1) = |\ell - j|$, which is verified only if either $z = w$ or $|\ell - j|$ is a multiple of $k + 1$. But, since $|\ell - j| \leq k$ implies $\sigma(w, z) \neq 0$, no two distinct nodes of $K_A(x, k)$ can have the same color.

Thus, it remains to prove that $\sigma(w, z) \neq M$. Assume by contradiction that $\sigma(w, z) = (i - h)(k + 1) + |j - \ell| = M$. Therefore, three cases may occur:

(i) $i-h \in \left[0, \left\lfloor \frac{M}{k+1} \right\rfloor - 1\right]$,

(ii) $i-h = \left\lfloor \frac{M}{k+1} \right\rfloor$,

(iii) $i-h = \left\lceil \frac{M}{k+1} \right\rceil$.

In case (i), $\sigma(w,z) = M$ implies $|j-\ell| = \sigma(w,z) - (i-h)(k+1) \geq M - \left(\left\lfloor \frac{M}{k+1} \right\rfloor - 1\right)(k+1) > k$ which contradicts the fact that $|j-\ell| \leq k$.

In case (ii), $\sigma(w,z)$ can be equal to M if and only if $|j-\ell| = M - \frac{k}{2}(k+1)$ since $\lfloor \frac{M}{k+1} \rfloor = \frac{k}{2}$, that is, $\sigma(w,z) = M$ if and only if $|j-\ell| = \left(\frac{k}{2}+1\right)$. Thus, in case (ii), for any pair of nodes z and w of $K_A(x,k)$ which do not satisfy the first condition in (1), it results that $j-\ell$ is equal to a positive integer and precisely,

$$j - \ell = \frac{k}{2} + 1.$$

But this violates the second condition in (1) because $(i-h)+(j-\ell) = \frac{k}{2} + \frac{k}{2} + 1 = k+1$.

Finally, in case (iii), $\sigma(w,z) = M$ if and only if $|j-\ell| = M - \left(\frac{k}{2}+1\right)(k+1)$. That is, for any pair of nodes z and w of $K_A(x,k)$ not satisfying the first condition in (1), it yields $j-\ell < 0$, and, precisely,

$$j - \ell = -\frac{k}{2}.$$

But again this violates the second condition in (1) because the distance between w and z is $(i-h) + |j-\ell| = \frac{k}{2} + 1 + \frac{k}{2} = k+1$.

In conclusion, for k even, any two nodes whose colors differ exactly by M are $k+1$ apart, and their relative positions are depicted in Figure 3(a).

When k is odd, it follows that $M = \left\lceil \frac{(k+1)^2}{2} \right\rceil = \frac{k^2+2k+1}{2} = k\frac{k+1}{2} + \frac{k+1}{2}$. Moreover, $\Delta = k$. Observe that $\sigma(w,z) < k\Delta + k = k(k+1) < 2M$ and $\sigma(w,z) > -M$. Then, $\sigma(w,z) \not\equiv 0 \bmod M$ is again equivalent to $\sigma(w,z) \neq 0$ and $\sigma(w,z) \neq M$.

Clearly, $\sigma(w,z) = 0$ iff $(i-h)k = |\ell - j|$, which is verified only if either $w = z$ (i.e., $i-h = \ell - j = 0$) or $i-h \geq 1$ and $\ell - j$ is a multiple of k. Hence, two distinct nodes of $K_A(x,k)$ which have the same color are at distance $(i-h) + |\ell - j| > k$.

It remains to prove that $\sigma(w,z) \neq M$. As before, three cases may occur:

(i) $i-h \in \left[0, \left\lfloor \frac{M}{k} \right\rfloor - 1\right]$,

(ii) $i-h = \left\lfloor \frac{M}{k} \right\rfloor$,

(iii) $i-h = \left\lceil \frac{M}{k} \right\rceil$.

Note that $\left\lfloor \frac{M}{k} \right\rfloor = \frac{k+1}{2}$ and $\left\lceil \frac{M}{k} \right\rceil = \frac{k+1}{2} + 1$.

Repeating the same reasoning done for k even, one can show again that any two nodes whose colors differ by M are $k+1$ apart. Their relative positions are illustrated in Figure 3(b).

So, the Array-Coloring Algorithm is conflict-free. Moreover, since it uses the minimum number of colors, the proposed mapping is optimal.

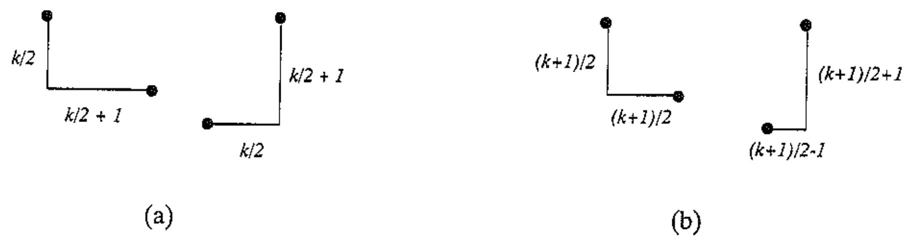

Figure 3: Relative positions in A of two nodes which are assigned to the same color: (a) k even, (b) k odd.

It is easy to see that the time required to color all the $n = rc$ nodes of an array is $O(n)$. Moreover, to color only a single node $x = (i, j)$ of the tree requires only $O(1)$ time, since $\gamma(x) = (i\Delta + j) \bmod M$, and hence the mapping is fast.

In order to prove that the mapping is balanced, observe that each color appears once in each sub-row of size M. Hence, the number m of nodes with the same color verifies $r\lfloor\frac{c}{M}\rfloor \leq m \leq r\lceil\frac{c}{M}\rceil$. □

Observe that the Array-Coloring Algorithm guarantees conflict-free access to some paths longer than k. Specifically, it is possible to access without conflicts any horizontal path of length M and any vertical path of length $L = \frac{M}{\text{g.c.d.}(M,\Delta)}$ because L is the minimum integer such that $L\Delta \equiv 0 \bmod M$. Finally, since the distance between two consecutive nodes on the same diagonal of A is 2, any $\lfloor\frac{k}{2}\rfloor$ consecutive elements on a diagonal can be accessed with no conflicts.

4 Accessing Paths in Circular Lists

Let a circular list C be the data structure D to be mapped into the multibank memory system. A circular list of n nodes, indexed consecutively from 0 to $n-1$, is a sequence of n nodes such that node i is connected to both nodes $(i-1) \bmod n$ and $(i+1) \bmod n$.

The graph $G_C = (V, E)$ representing C is a ring, whose vertices correspond to the elements of C and whose arcs correspond to any pair of adjacent elements of C. For the sake of simplicity, C will be used instead of G_C since there is no ambiguity.

Lemma 2 *Let* $M = \begin{cases} n & \text{if } n < k+1, \\ (k+1) + \left\lceil \frac{n \bmod (k+1)}{\lfloor \frac{n}{(k+1)} \rfloor} \right\rceil & \text{if } n \geq k+1. \end{cases}$

At least M memory banks are required for conflict-free accessing P_k in C.

Proof For conflict-free accessing P_k in C two nodes with the same color must be at distance at least $k+1$. When $n < k+1$, all the nodes are mutually at distance less than k and must all be colored with different colors. When $n \geq k+1$, each color may appear at most $t = \left\lfloor \frac{n}{(k+1)} \right\rfloor$ times. Therefore,

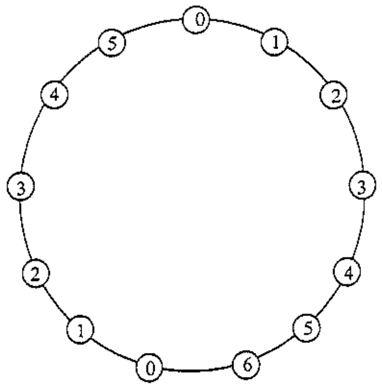

Figure 4: Conflict-free access to P_4 in a circular list C of 13 nodes colored by the Circular-List-Coloring algorithm with $M = 7$.

at least $\lceil \frac{n}{t} \rceil$ colors are needed. Observed that $n = \left\lfloor \frac{n}{(k+1)} \right\rfloor (k+1) + (n \bmod (k+1))$, it follows that at least $M = \lceil \frac{n}{t} \rceil = (k+1) + \left\lceil \frac{n \bmod (k+1)}{\lfloor \frac{n}{(k+1)} \rfloor} \right\rceil$ memory banks are required. □

Below, an optimal conflict-free mapping is provided to color all the nodes of a circular list C using as few colors as in Lemma 2. As before, the color assigned to node x is denoted by $\gamma(x)$.

Algorithm Circular-List-Coloring (C, k);

- Set $M = \begin{cases} n & \text{if } n < k+1 \\ (k+1) + \left\lceil \frac{n \bmod (k+1)}{\lfloor \frac{n}{(k+1)} \rfloor} \right\rceil & \text{if } n \geq k+1 \end{cases}$
- Set $\theta = sM$ where $s = \begin{cases} n \bmod (M-1), & \text{if } n \bmod M \neq 0 \\ \frac{n}{M}, & \text{if } n \bmod M = 0 \end{cases}$
- Assign to node $x \in C$, the color $\gamma(x) = \begin{cases} x \bmod M & \text{if } x < \theta \\ (x - \theta) \bmod (M-1) & \text{if } x \geq \theta \end{cases}$

Note that a linear (that is, non circular) list L can be optimally colored to conflict-free access P_k with $M' = k+1$ colors, which matches the trivial lower bound given by the number of nodes in P_k. In fact, L can be optimally colored by a *naive algorithm* which assigns to node x the color $\gamma(x) = x \bmod M'$. Such a naive algorithm does not work for circular lists. For example, consider the circular list C of 13 nodes, shown in Figure 4, to be colored to access P_4. Applying the naive algorithm with $M' = 5$, only the first 10 nodes can be feasibly colored with 5 colors, but 3 additional colors are then required for feasibly coloring the last 3 nodes, for a total of 8 colors. In contrast, the optimal Circular-List-

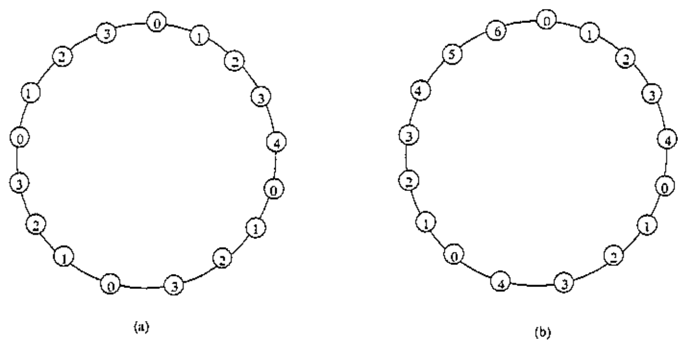

Figure 5: A circular list C of 17 nodes colored to conflict-free access P_3 according to: (a) the Circular-List-Coloring algorithm (where $M = 5$), (b) the naive algorithm with $M' = 5$.

Coloring algorithm requires 7 colors only. Moreover, it is worth to point out that the naive algorithm does not always work for circular lists even when applied with $M' = M = (k+1) + \left\lceil \frac{n \bmod (k+1)}{\left\lfloor \frac{n}{(k+1)} \right\rfloor} \right\rceil$. For instance, for $n = 17$ and $k = 3$, Lemma 2 gives $M = 5$. Applying the naive algorithm with $M' = 5$ to this instance, 15 nodes can be colored using 5 colors, but 2 additional colors are needed for feasibly coloring the last 2 nodes for a total of 7 colors (as shown in Figure 5(b)). Instead, the optimal coloring provided by the Circular-List-Coloring algorithm uses only 5 colors, as shown in Figure 5(a). Indeed, the naive algorithm always produces a feasible (although not necessarily optimal) coloring if applied using $M' = (k+1) + n \bmod (k+1)$.

Theorem 2 *The Circular-List-Coloring mapping is optimal, fast, and balanced.*

Proof To prove optimality, two cases may be distinguished. If $n \equiv 0 \bmod M$, Lemma 2 gives $M = k+1$ and the Circular-List-Coloring algorithm reuses the same color at distance M. Hence, no conflict arises. If $n \not\equiv 0 \bmod M$, Lemma 2 gives $M \geq k+2$. Two nodes get the same color only if they are at distances M or $M-1$, which are both greater than or equal to $k+1$. Hence, as before, no conflict arises. Since the algorithm uses as few colors as possible, the mapping is optimal. It is also fast since each node is colored in constant time. Finally, each color is assigned to exactly $\frac{n}{M}$ nodes when n is a multiple of M, and no more than $\left\lceil \frac{\min(n,\theta)}{M} \right\rceil + \left\lceil \frac{\max(n-\theta,0)}{M-1} \right\rceil$ nodes are colored with the same color in all the other cases. □

It is interesting to note at this point that, given a circular list of n nodes, the minimum number $M = M(n,k)$ of colors required to conflict-free access P_k satisfies the following properties (see Figure 6):

- Up to $n = 2(k+1) - 1$, $M(n,k) = n$ results, i.e. all the nodes must have different colors. Indeed, all of them are mutually at distance no more than k and, therefore, they form a clique on the graph G_{CP_k}.
- When $n > 2(k+1) - 1$, $M(n,k)$ depends on both n and k, and, for a fixed k, is not a monotone

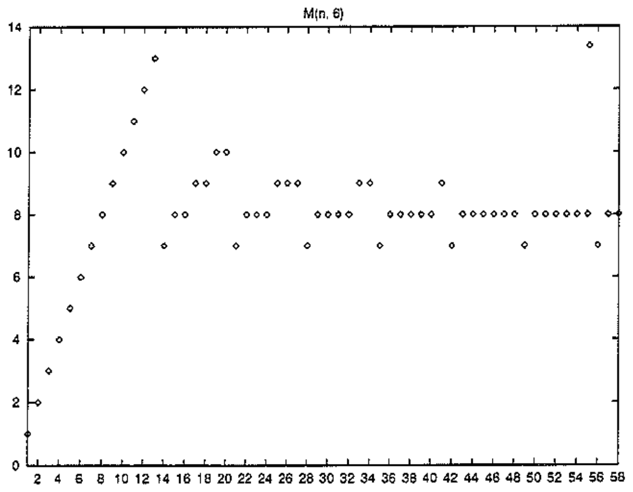

Figure 6: The number of colors $M(n,6)$ required to conflict-free access P_6 when n ranges between 1 and 58.

function of n. In contrast, for arrays and trees (as will be proved in the next section), M depends only on k and is monotone.

5 Accessing Paths in Complete Trees

Let a rooted complete binary tree B be the data structure to be mapped into the multibank memory system. The *level* of node $x \in B$ is defined as the number of edges on the path from x to the root, which is at level 0. The maximum level of the nodes of B is the *height* of B. Let $Lev_B(i)$ be the set of all nodes of B at level $i \geq 0$.

A complete binary tree of height H is a rooted tree B in which all the leaves are at the same level and each internal node has exactly 2 children. Thus, $Lev_B(i)$ contains 2^i nodes. The h-th *ancestor* of the node (i,j) is the node $(i-h, \lfloor \frac{j}{2^h} \rfloor)$, while its children are the nodes $(i+1, 2j)$ and $(i+1, 2j+1)$, in the left-to-right order.

From now on, the generic node x, which is the j-th node of $Lev_B(i)$, with $j \geq 0$ counting from left to right, will be denoted by $x = (i,j)$. Therefore, the generic path instance $P_k[x,y]$ will be denoted by $P_k[(i,j),(r,s)]$, where $x = (i,j)$ and $y = (r,s)$.

Lemma 3 *At least $M = 2^{\lfloor \frac{k}{2} \rfloor + 1} + 2^{\lceil \frac{k}{2} \rceil} - 2$ memory banks are required to conflict-free access P_k in B.*

Proof Consider a generic node $x = (i,j)$. All the $2^{\lfloor \frac{k}{2} \rfloor + 1} - 1$ nodes in the subtree S of height $\lfloor \frac{k}{2} \rfloor$ rooted at the $\lfloor \frac{k}{2} \rfloor$-th ancestor of x are mutually at distance not greater than k.

In addition, consider the $\lceil \frac{k}{2} \rceil$ nodes, $\mu_1, \mu_2, \ldots \mu_{\lceil \frac{k}{2} \rceil}$, ancestors of x, on the path I of length $\lceil \frac{k}{2} \rceil$ from the $\lfloor \frac{k}{2} \rfloor$-th ancestor of x up to the k-th ancestor of x. All these nodes are at distance not greater than k from node x, and together with the nodes of S they are at mutual distance not greater than k.

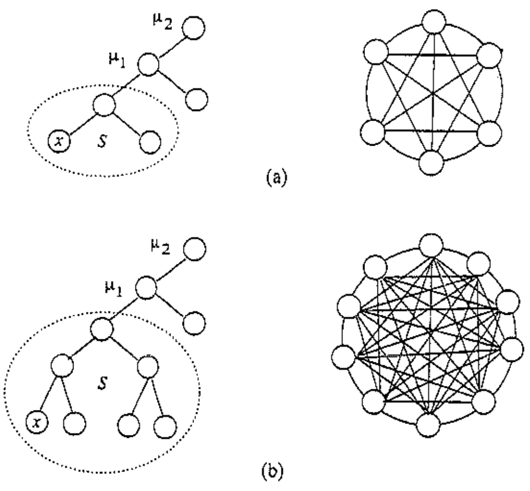

Figure 7: A subset $K_B(k)$ of nodes of B that forms a clique in G_{BP_k}: (a) $k = 3$, (b) $k = 4$.

Moreover, for $1 \le j \le \lceil \frac{k}{2} \rceil - 1$, consider the $2^{\alpha_j+1} - 1$ nodes in the complete subtree of height $\alpha_j = k - \lfloor \frac{k}{2} \rfloor - j - 1$, rooted at the μ_j's child which does not belong to I. Such nodes are at distance not greater than k from x. Furthermore, these nodes, along with the nodes of S and I, are all together at mutual distance not greater than k.

Hence, in the associated graph G_{DP_k} there is at least a clique of size

$$2^{\lfloor \frac{k}{2} \rfloor+1} - 1 + \left\lceil \frac{k}{2} \right\rceil + \sum_{j=1}^{\lceil \frac{k}{2} \rceil - 1} \left(2^{\alpha_j+1} - 1\right) = 2^{\lfloor \frac{k}{2} \rfloor+1} - 1 + \left\lceil \frac{k}{2} \right\rceil + \sum_{h=0}^{\lceil \frac{k}{2} \rceil - 2} \left(2^{h+1} - 1\right).$$

From that, the claim easily follows. Figure 7 shows a subset $K_B(k)$ of nodes of B which are at pairwise distance not greater than k, for $k = 3$ and 4, and hence forms a clique in the associated graph G_{BP_k}.

□

An optimal conflict-free mapping to color a complete binary tree B acts as follows.

A basic subtree $K_B(k)$ defined as in the proof of Lemma 3 is identified and colored. Such a tree is then overlaid to B in such a way that the uppermost $\left\lfloor \frac{k}{2} \right\rfloor$ levels of B coincide with the lowermost $\left\lfloor \frac{k}{2} \right\rfloor$ levels of $K_B(k)$. Then, the complete coloring of B is produced level by level by assigning to each node the same color as an already colored node.

Formally, for a given k, define the binary tree $K_B(k)$ as follows:

- $K_B(k)$ has a leftmost path of $k+1$ nodes.
- the root of $K_B(k)$ has only the left child;
- a complete subtree of height $i-1$ is rooted at the right child of the node at level i on the leftmost path of $K_B(k)$.

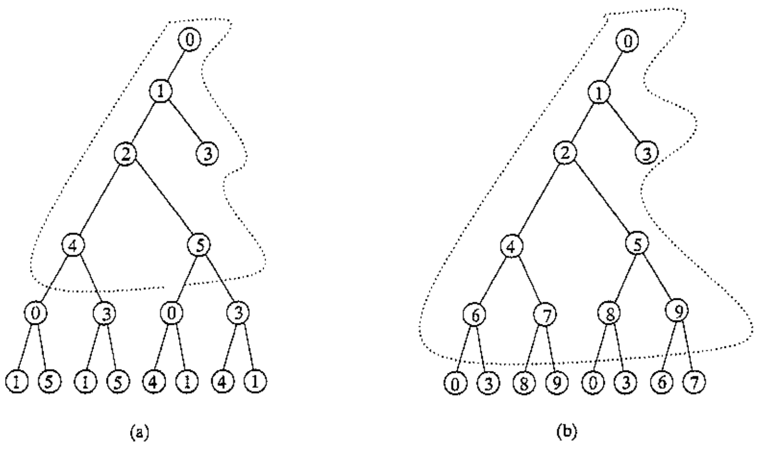

Figure 8: Coloring of B for conflict-free accessing: (a) P_3, (b) P_4. (Both $K_B(3)$ and $K_B(4)$ are depicted by dash splines.)

The $2^{\lfloor \frac{k}{2} \rfloor+1} + 2^{\lceil \frac{k}{2} \rceil} - 2$ nodes of $K_B(k)$ must be colored with $2^{\lfloor \frac{k}{2} \rfloor+1} + 2^{\lceil \frac{k}{2} \rceil} - 2$ different colors. Thus, the uppermost $\left\lfloor \frac{k}{2} \right\rfloor$ levels of B are already colored.

For the sake of simplicity, to color the remaining part of B, the levels are counted starting from the root of $K_B(k)$. That is, the level of the root of B will be renumbered as level $\left\lfloor \frac{k}{2} \right\rfloor + 1$. Now, fixed $x = (0, k)$, the algorithm to color B acts as follows.

```
Algorithm Binary-Tree-Coloring ($B, k$);
  • Set $M = 2^{\lfloor \frac{k}{2} \rfloor+1} + 2^{\lceil \frac{k}{2} \rceil} - 2$;
  • Color $K_B(k)$ with $M$ colors;
  • Visit the tree $B$ in breadth first search, and for each node $x = (i, j)$ of $B$, with
    $j \geq k + 1$, do:
      – Set $\pi = j \bmod 2^{\lfloor \frac{k}{2} \rfloor}$, $\alpha = \lceil \log(\pi+1) \rceil$, $\delta = k - \alpha + 1$ and $\tau = \left( \left\lfloor \frac{j}{2^{\delta-1}} \right\rfloor - 1 \right) \bmod 2$;
      – Assign to $x$ the same color as that of the node $y = (r, s)$, where
            $r = i - \delta + \alpha$
        and
            $s = \begin{cases} \left\lfloor \frac{j}{2^{\delta}} \right\rfloor & \text{if } \alpha = 0 \\ \left\lfloor \frac{j}{2^{\delta}} \right\rfloor 2^{\alpha} + \tau 2^{\alpha-1} + (\pi \bmod 2^{\alpha-1}) & \text{if } \alpha \neq 0 \end{cases}$
```

Examples of colorings to conflict-free access P_3 and P_4 are illustrated in Figure 8.

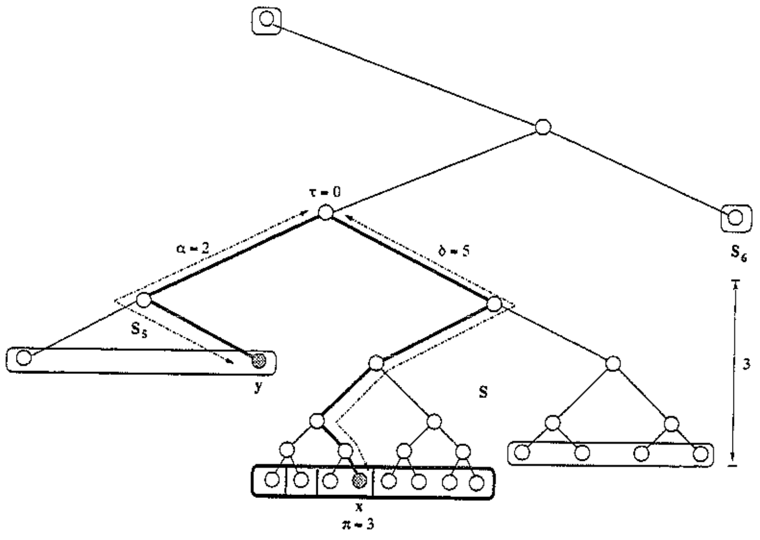

Figure 9: For $k = 6$, node $x = (i, j)$ inherits the same color as node $y = (r, s)$.

Theorem 3 *The Binary-Tree-Coloring mapping is optimal, fast and balanced.*

Proof. To prove that the mapping is optimal, it must be shown that it is conflict-free and it uses as few colors as those given by Lemma 3. First, observe that the $2^{\lfloor \frac{k}{2} \rfloor}$ leaves of a subtree of height $\lfloor \frac{k}{2} \rfloor$ are at mutual distance not greater than k, and therefore they must be colored with all different colors. Thus, let each level of B be partitioned (starting from the leftmost node) into consecutive blocks of size $2^{\lfloor \frac{k}{2} \rfloor}$. The *block* $b(i, w)$, with $w \geq 0$, at level i of B consists of the $2^{\lfloor \frac{k}{2} \rfloor}$ consecutive nodes $(i, w2^{\lfloor \frac{k}{2} \rfloor}), (i, w2^{\lfloor \frac{k}{2} \rfloor} + 1), \ldots, (i, (w+1)2^{\lfloor \frac{k}{2} \rfloor} - 1)$, which must all be assigned to a different color. Consider the node $x = (i, j)$ to be colored. The node $x = (i, j)$ belongs to the block $b_x = b\left(i, \left\lfloor \frac{j}{2^{\lfloor \frac{k}{2} \rfloor}} \right\rfloor\right)$, and it appears in the $(\pi + 1)$-th position inside the block. Consider the leftmost node z of b_x, where $z = \left(i, \left\lfloor \frac{j}{2^{\lfloor \frac{k}{2} \rfloor}} \right\rfloor 2^{\lfloor \frac{k}{2} \rfloor}\right)$. Then, a generalization $K_B(z, k)$ of $K_B(k)$ can be defined depending on z.

$K_B(z, k)$ includes the following nodes of B:

- the nodes on the path Γ of length k from the father of z up to the $(k + 1)$-th ancestor of z;
- for $\lfloor \frac{k}{2} \rfloor + 2 \leq q \leq k$, the nodes of the complete binary tree S_q of height $k - q$ rooted at the child, which does not belong to Γ, of the q-th ancestor of z;
- the nodes of the complete binary tree S of height $\lfloor \frac{k}{2} \rfloor$ rooted at the $\left(\lfloor \frac{k}{2} \rfloor + 1\right)$-th ancestor of z.

It is crucial to note that all the following nodes are at distance $k + 1$ from all the nodes in b_x:

(i) the root of $K_B(z, k)$,

(ii) the leaves of S_q, with $\lfloor \frac{k}{2} \rfloor + 2 \leq q \leq k$,

(iii) the leaves of S, which are not parents of any node in b_x.

The nodes of $b_x = b\left(i, \left\lfloor \frac{j}{2^{\lfloor \frac{k}{2} \rfloor}} \right\rfloor\right)$ are colored from left to right copying the same colors used in the nodes of $K_B(z,k)$ specified in (i), (ii), and (iii) above, and considered by increasing level and from left to right, as illustrated in Figures 10 and 11 for k even and odd, respectively. In particular,

- $z = \left(i, \left\lfloor \frac{j}{2^{\lfloor \frac{k}{2} \rfloor}} \right\rfloor\right)$ is assigned to the same color as the root of $K_B(z,k)$, which is the $(k+1)$-th ancestor of x;
- for $k \geq q \geq \left\lfloor \frac{k}{2} \right\rfloor + 2$, the 2^{k-q} nodes of b_x, $(i, \lfloor \frac{j}{2^\delta} \rfloor 2^{\lfloor \frac{k}{2} \rfloor} + 2^{k-q} - 1 + 1), \dots (i, \lfloor \frac{j}{2^\delta} \rfloor 2^{\lfloor \frac{k}{2} \rfloor} + 2^{k-q} - 1 + 2^{k-q})$, are assigned to the same colors as the leaves of the tree S_q.

Observe that the number of nodes colored with the two steps above is $1 + \sum_{q=\lfloor \frac{k}{2} \rfloor + 2}^{k} 2^{k-q} = 2^{\lceil \frac{k}{2} \rceil - 1}$. When k is odd, this is enough to color the entire block since $2^{\lceil \frac{k}{2} \rceil - 1} = 2^{\lfloor \frac{k}{2} \rfloor + 1 - 1} = 2^{\lfloor \frac{k}{2} \rfloor}$. In fact, the set of nodes of $K_B(z,k)$ specified in (iii) above is empty for k odd. In contrast, when k is even, only the first half of the block has been colored since $2^{\lceil \frac{k}{2} \rceil - 1} = 2^{\lfloor \frac{k}{2} \rfloor - 1}$. Thus, to color the second half of the block, one further step is required, which uses the colors of the nodes of $K_B(z,k)$ specified in (iii) above:

- The rightmost $2^{\lfloor \frac{k}{2} \rfloor - 1}$ nodes of b_x are assigned to the same colors as the rightmost (resp., leftmost) $2^{\lfloor \frac{k}{2} \rfloor - 1}$ leaves of the complete binary tree rooted at $(\left\lfloor \frac{k}{2} \right\rfloor + 1)$-th ancestor of z, depending on the fact that the $\left\lfloor \frac{k}{2} \right\rfloor$-th ancestor of z is a left (resp., right) child of its father.

In order to prove that the mapping is conflict-free, an inductive reasoning on the level i of the tree is followed. The basis for the induction is $i = k$, when the tree coincides with $K_B(k)$ and it is colored, by definition, with all different colors. For $i > k$, consider a generic node $x = (i,j)$, its block b_x and its leftmost node z. By inductive hypothesis, all the nodes in the tree up to level $i-1$ are colored in a conflict-free manner, but with color repetitions. In particular, the subtree $K_B(z,k)$ is conflict-free and since its nodes are mutually at distance at most k they must have been assigned to all different colors. The algorithm colors b_x copying the colors of some nodes in $K_B(z,k)$, specified in (i), (ii), and (iii), which are exactly at distance $k+1$ from the nodes of b_x. Therefore, there are no color repetitions in b_x and no conflict can arise. Note that nodes in different blocks at level i may inherit the same color, but since any two nodes in different blocks are at distance at least $k+1$ no conflict can arise. Therefore, all the nodes in the tree up to level i are colored in a conflict-free manner.

Finally, since the tree is colored with the colors of $K_B(k)$, whose number equals the lower bound of Lemma 3, the tree-coloring mapping is optimal.

It is easy to see that the time required to color all the n nodes of a tree is $O(n)$. However, to color only a single node x of the tree requires only $O(\log n)$ time since, in the worst case, all the nodes along a path from x up to the root must have been colored.

One can readily see that, if the height H of the tree B is a multiple of k, then the nodes of B can be partitioned into $m = \lceil \frac{2^{H+1}-1}{M} \rceil$ subsets, each of which induces a copy of $K_B(k)$. Therefore, each color is used m times, and the mapping is balanced. $\square$

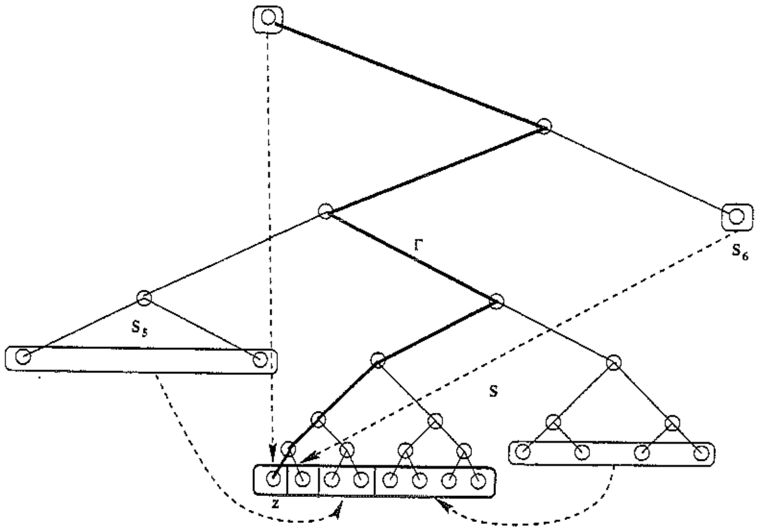

Figure 10: The generalization $K_B(z,6)$ of $K_B(6)$ for the node z. The root of $K_B(z,6)$, the leaves of the subtrees S_6, S_5, and the rightmost leaves of S are used to color the nodes in the block b_z.

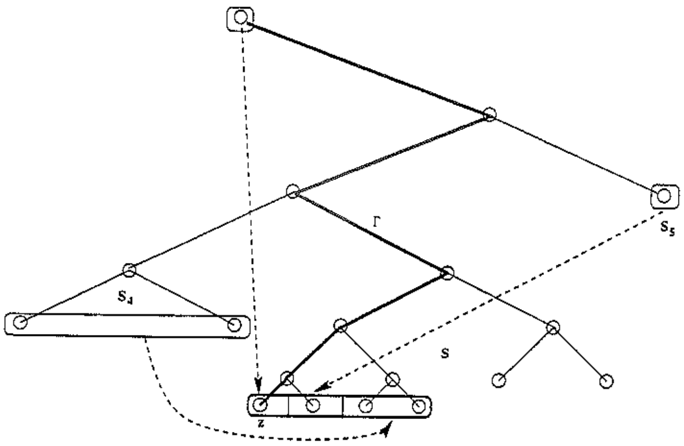

Figure 11: The generalization $K_B(z,5)$ of $K_B(5)$ for the node z. The root of $K_B(z,5)$ and the leaves of the subtrees S_4 and S_5 are used to color the nodes in the block b_z.

The results shown for binary trees can be extended to a q-ary tree Q, with $q \geq 2$.

Corollary 1 *At least*

$$M = \frac{q^{\lfloor \frac{k}{2} \rfloor + 1} - 1}{q-1} + \left\lceil \frac{k}{2} \right\rceil + (q-1) \sum_{h=0}^{\lceil \frac{k}{2} \rceil - 2} \frac{q^{h+1} - 1}{q-1} = 1 + \frac{q^{\lfloor \frac{k}{2} \rfloor + 1} - 1 + q^{\lceil \frac{k}{2} \rceil} - q}{q-1}$$

memory modules are required to conflict-free access P_k in a q-ary tree Q. □

Similarly to the binary case, for a given k, define a q-ary tree $K_Q^q(k)$ as follows:

- $K_Q^q(k)$ has a leftmost path of $k+1$ nodes;
- the root of $K_Q^q(k)$ has only the leftmost child;
- a complete subtree of height $i-1$ is rooted at the $q-1$ rightmost children of the node at level i on the leftmost path of $K_Q^q(k)$.

Such a $K_Q^q(k)$ is then overlaid to Q in such a way that the uppermost $\left\lfloor \frac{k}{2} \right\rfloor$ levels of Q coincide with the lowermost $\left\lfloor \frac{k}{2} \right\rfloor$ levels of $K_Q^q(k)$. Then, the complete coloring of Q is produced level by level by assigning to each node the same color as an already colored node.

For the sake of simplicity, to color the remaining part of Q, the levels are again counted starting from the root of $K_Q^q(k)$. That is, the level of the root of Q will be renumbered as level $\left\lfloor \frac{k}{2} \right\rfloor + 1$. Now, the algorithm to color Q is the following:

```
Algorithm q-ary-Tree-Coloring (Q, k);

  • Set $M = 1 + \frac{q^{\lfloor \frac{k}{2} \rfloor + 1} - 1 + q^{\lceil \frac{k}{2} \rceil} - q}{q-1}$;
  • Color $K_Q^q(k)$ with $M$ colors;
  • Visit the tree $Q$ in breadth first search, and for each node $x = (i, j)$ of $Q$, with $j \geq k+1$, do:
      – Set $\pi = j \bmod q^{\lfloor \frac{k}{2} \rfloor}$, $\alpha = \lceil \log_q(\pi + 1) \rceil$, $\delta = k - \alpha + 1$ and $\tau = \left( \left\lfloor \frac{j}{q^{\delta-1}} \right\rfloor + 1 \right) \bmod q$;
      – Assign to $x$ the same color as that of the node $y = (r, s)$, where
            $r = i - \delta + \alpha$
        and
            $s = \begin{cases} \left\lfloor \frac{j}{q^\delta} \right\rfloor & \text{if } \alpha = 0 \\ \left\lfloor \frac{j}{q^\delta} \right\rfloor q^\alpha + \tau q^{\alpha-1} + (\pi \bmod q^{\alpha-1}) & \text{if } \alpha \neq 0 \end{cases}$
```

By a reasoning similar to that employed for complete binary trees, the optimality of the q-ary-Tree-Coloring Algorithm easily follows.

6 Conclusions

In this paper, the problem of conflict-free accessing *arbitrary* paths P_k in particular data structures, such as bidimensional arrays, circular lists and complete trees, has been considered for the first time and reduced to variants of graph-coloring problems. Optimal, fast and balanced mappings have been proposed. Indeed, the memory bank to which a node is assigned is computed in constant time for arrays and circular lists, while it is computed in logarithmic time for complete trees. However, it remains as an open question whether a tree node can be assigned to a memory bank in constant time.

On the other hand, the conflict-free access to P_k on an arbitrary data structure D is NP-complete [13], and this justifies the investigation of *good* heuristics. This problem is equivalent to the classical node coloring problem in the associated graph G_{DP_k}. Therefore, it can be solved by the most effective coloring heuristic known so far, that is, the *saturation-degree* heuristic [6], which works as follows. Let $N(x)$ be the neighborhood of node x in the associated graph G_{DP_k}. At each iteration, the saturation-degree heuristic selects the node x to be colored as one with the largest number of different colors already assigned in $N(x)$. Ties between nodes are broken by preferring the node x with the largest number of colored nodes in $N(x)$. Once selected, node x is assigned the lowest color not yet assigned in $N(x)$.

As experimentally proved in [5], the saturation-degree heuristic is especially effective when the minimum number of colors is given by the size of the largest clique K of G_{DP_k}. Therefore, it should work efficiently also for the conflict-free access problem, and, in particular, for d-dimensional arrays as well as for generic, i.e. not necessarily complete, trees. Indeed, it is expected in such cases that the minimum number of required memory banks be equal to the lower bound given by the size of the largest clique K of G_{DP_k}, as happened for bidimensional arrays and complete trees. Unfortunately, the resulting coloring is not guaranteed to be optimal, fast or balanced. Moreover, it is still an open question to determine whether the problem of conflict-free accessing paths on d-dimensional arrays and generic trees is NP-complete.

Finally, in a more practical perspective, the number of memory banks available could be fixed to a constant μ, depending on the memory configuration. Then, if the number of memory modules $M(k)$ required for a given P_k is larger than μ, no conflict-free access is possible. However, assume that $P_{k'}$ is the longest path that can be accessed without conflicts using μ memory banks, i.e. $M(k') \leq \mu$. Then, accessing P_k, no more than $\lceil \frac{k}{k'} \rceil$ conflicts may arise. Hence, the proposed mappings are scalable.

Acknowledgement

The authors are grateful to Richard Tan for his helpful comments, and to Thomas McCormick for having provided the reference [13].

References

[1] V. Auletta, S. K. Das, M. C. Pinotti, and V. Scarano, "Toward a Universal Mapping Algorithm for Accessing Trees in Parallel Memory Systems", *Proceedings of IEEE Int'l Parallel Processing Symposium*, Orlando, pp. 447-454, Apr. 1998.

[2] V. Auletta, A. De Vivo, V. Scarano, "Multiple Template Access of Trees in Parallel Memory Systems". *Journal of Parallel and Distributed Computing*, Vol. 49, 1998, pp. 22-39.

[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. on Parallel and Distrib. Systems*, Vol. 8, 1997, pp. 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*, Vol. 1, 1988, pp. 103-107.

[5] R. Battiti, A.A. Bertossi, M.A. Bonuccelli, "Assigning Codes in Wireless Networks: Bounds and Scaling Properties", *Wireless Networks*, Vol. 5, 1999, pp. 195-209.

[6] D. Brélaz, "New Methods to Color the Vertices of a Graph", *Communications of ACM*, Vol. 22, 1979, pp. 251-256.

[7] P. Budnik, D.J. Kuck. "The Organization and Use of Parallel Memories". *IEEE Trans Comput.*, Vol. 20, 1971, pp. 1566-1569.

[8] S. K. Das and F. Sarkar, "Conflict-Free Data Access of Arrays and Trees in Parallel Memory Systems", *Proc. of the Sixth IEEE Symposium on Parallel and Distributed Processing*, Dallas, TX, Oct. 1994, pp. 377-384.

[9] S. K. Das, F. Sarkar and M. C. Pinotti, "Parallel Priority Queues in Distributed Memory Hypercubes", *IEEE Transactions on Parallel and Distributed Systems*, Vol. 7, 1996, pp. 555-564.

[10] S.K. Das and M.C. Pinotti, "Load Balanced Mapping of Data Structures in Parallel Memory Modules for Fast and Conflict-Free Templates Access" *Proc. 5th Int. Workshop on Algorithms and Data Structures (WADS'97)* Halifax NS, Aug. 1997, LNCS 1272, (Eds. Dehne, Rau-Chaplin, Sack, Tamassia), pp. 272-281.

[11] J. Denes, and A. D. Keedwell, *Latin Squares and Their Applications*, Academic Press, New York, 1974.

[12] K. Kim, V.K. Prasanna, "Latin Squares for Parallel Array Access", *IEEE Transactions on Parallel and Distributed Systems*, Vol. 4, 1993, pp. 361-370.

[13] S.T. McCormick, "Optimal Approximation of Sparse Hessians and its Equivalence to a Graph Coloring Problem", *Mathematical Programming*, Vol. 26, 1983, pp. 153–171.

[14] M. C. Pinotti, S. K. Das, and F. Sarkar, "Conflict-Free Template Access in k-ary and Binomial Trees", *Proceedings of ACM-Int'l Conference on Supercomputing*, Wein, Austria, July 7-11, 1997.

[15] H.D. Shapiro, "Theoretical Limitations on the Efficient Use of Parallel Memories", *IEEE Trans. on Computers*, Vol. 27, 1978, pp. 421-428.