COLORING QUADTREES

Messaoud Benantar[1], Uğur Doğrusöz[2]
Joseph E. Flaherty[2], and Mukkai S. Krishnamoorthy[2]

[1] Large Scale Computing Division, IBM Corporation,
MS 105, Neighborhood Road, Kingston, NY 12401
[2] Department of Computer Science,
Rensselaer Polytechnic Institute, Troy, NY 12180

Abstract

Consider solving linear elliptic partial differential systems on shared-memory parallel computers; with mesh generation and adaptive mesh refinement utilizing an underlying quadtree structure for data management, the terminal nodes of the quadtree can be colored so as to separate contiguous spatial regions that may be processed in parallel without conflict. The quadtree coloring algorithm given in [2, 4] has a linear time complexity and uses a maximum of six colors. However, it is restricted to those quadtrees which correspond to quadrilateral meshes with at most one-level difference across quadrant edges. In this paper, we present a linear eight-color algorithm that works for *any* quadtree by first six-coloring it and then removing coloring conflicts introduced by level differences with a fourth color pair in a consistent manner.

1 Introduction

Triangular, quadrilateral or mixed triangular and quadrilateral meshes may be constructed on an arbitrarily complex two-dimensional problem domain by initially embedding it in a square universe and recursively bisecting those edges of squares that intersect the domain's boundary [1]. This process leads to a *quadtree* dissection of the domain with nodes of the tree representing square regions called *quadrants*. Quadrants at terminal tree nodes may be further divided or distorted to produce the triangular, quadrilateral or mixed mesh. An example of a finite quadtree mesh generation for a domain consisting of a rectangle and a quarter of a circle apperas in Figure 1. Figure 1.a through Figure 1.c show the formation of level 1 through level 3 quadrants, respectively. The corresponding quadtree is given in Figure 1.e. Note how each node of the quadtree corresponds to the quadrants of the mesh and that the higher the level of a quadrant is, the deeper the associated node in the quadtree is. This procedure extends to three-dimensional domains using an octree decomposition of the domain [6]. Benantar et al. [3] developed a coloring procedure that separated contiguous (two quadrants are contiguous or adjacent even if they intersect at only one point; so, all four quadrants of a square are mutually adjacent) terminal quadrants in order to process quadrants having the same color in parallel without conflict on shared-memory computers. Algorithms using a maximum of six and eight colors were described with the six-color procedure providing superior performance due to the finer data granularity [2, 4]. When used with finite element computations, data associated with all elements in a quadrant was generated and assembled into the global algebraic system, which was solved in parallel by colors using a preconditioned conjugate gradient method [3]. However, these procedures are restricted to quadrilateral meshes with at most one-level difference across quadrant edges. In this paper, we present a linear-time eight-coloring algorithm that works for *any* quadtree by

first six-coloring it and then removing coloring conflicts introduced by level differences with a fourth color pair in a consistent manner. Maintaining a one level difference between adjacent quadrants is normally used to avoid severe mesh gradation. Its use at the quadrant level, rather than at the mesh level, may be too conservative. The one-level difference may also be abandoned when using methods of different order in different spatial regions. This strategy, called p-refinement, is capable of achieving very high accuracy.

Following definition, lemmas and theorem are taken from [2].

Definition 1 *[2] A* quasi-binary tree *is a directed binary graph*[1] *obtained from a finite quadtree by the following assertive algorithm:*

- *The root of the finite quadtree corresponds to the root of the quasi-binary tree.*
- *Every terminal quadrant is associated with a node of the quasi-binary tree, but not conversely.*
- *Nodes across a common horizontal edge in the quadtree representation of the domain are connected in the quasi-binary tree.*
- *When a quadrant is divided, its parent in the quasi-binary tree becomes the root of a subtree.*

Lemma 1 *[2] At most three branches can be adjacent to each other in a quasi-binary tree.*

Lemma 2 *[2] Three colors are necessary and sufficient to color the branches of a quasi-binary tree so that no two adjacent branches have the same color.*

Theorem 1 *[2] Six colors are sufficient to color the nodes of a quasi-binary tree so that no two adjacent nodes of the same branch and no two nodes from adjacent branches have the same color.*

Proof: For every color of the three colors used to color the branches of a quasi-binary tree so that no two adjacent branches have the same color Lemma 2, we use a new color to color the simple path corresponding to the branch so that no two adjacent nodes from the same branch have the same color. ■

An example of the six-color procedure is shown in Figure 2.

In this chapter, we describe a linear-time eight-color algorithm for *general* quadtrees (i.e. quadrilateral meshes with no restriction on level difference).

2 Eight-Coloring Algorithm

When the restriction of at most one-level difference across quadrant edges is removed, there is no one-to-one correspondence between the nodes of the quasi-binary tree obtained and the quadrants of the original mesh. In this case, we let the node highest (lowest) in the upper (lower) quasi-binary tree to uniquely represent the quadrant that it is in and color all the other nodes in that quadrant the same as this representative node. Please refer to Figure 3.a for an example. This methodology might lead to cycles of branches colored the same. If the length of such a cycle is even, there is no problem since alternating two colors along this cycle will not create any conflicts. However, such cycles of odd length will (see the r-cycle in Figure 3 for an example).

Our strategy for avoiding such cycles involves introduction of a fourth color pair, say y. In order to do this, we assign all r, g, and b cycles a level number number depending on whose parity the corresponding cycle is to be broken from left or right. The deeper the cycle is in the nesting of cycles, the higher this number will be. Figure 4 shows an example of the assignment of level numbers and how this number is used for deciding whether to break from left or right. Let us identify a node on

[1] Formed by connecting directed binary trees root to root and leaf to leaf.

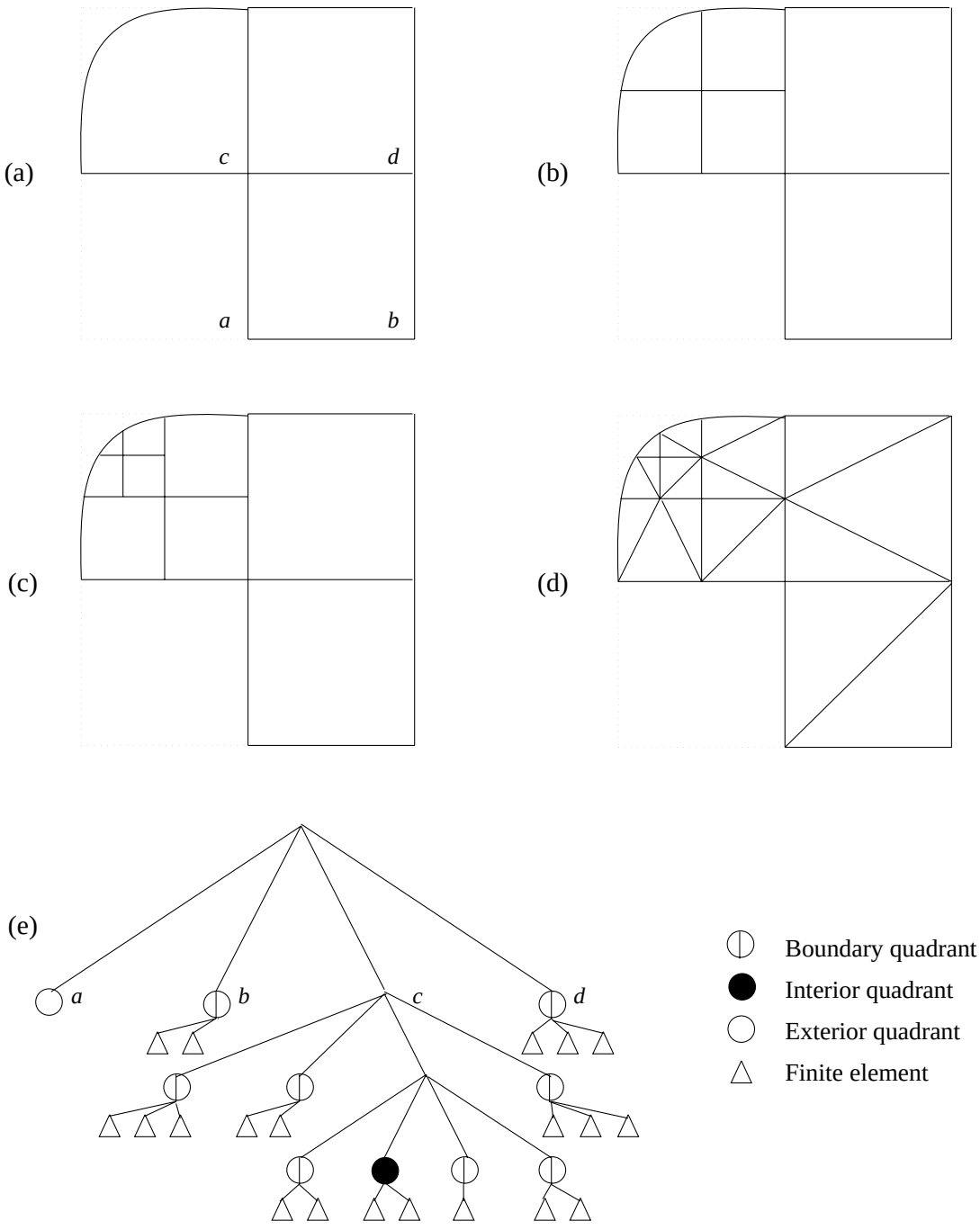


Figure 1: An example of a finite quadtree mesh generation for a domain consisting of a rectangle and a quarter of a circle. In this case, the quadrants obtained are further decomposed into triangular regions.

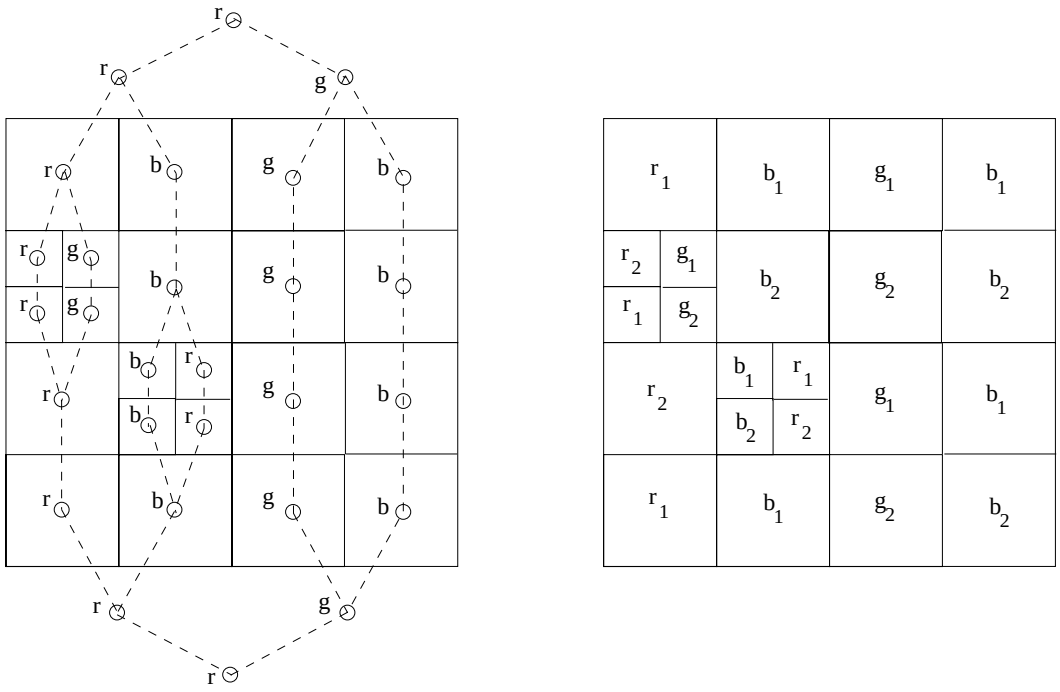


Figure 2: **(a)** An example of six-coloring quadtrees. The dashed lines represent the branches of the quasi-binary tree formed. *r, g*, and *b* are used to represent three separate color pairs. **(b)** A six-coloring of the quadrants obtained by simply replacing each color pair in *(a)* with two different colors alternating.

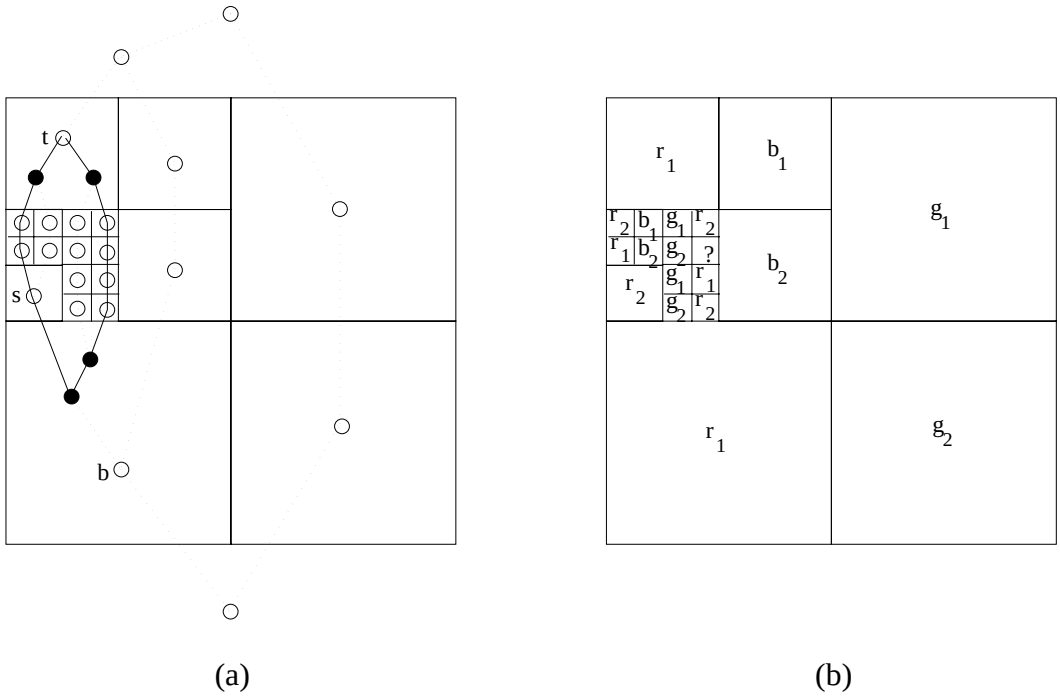


Figure 3: An example for which the six-color algorithm fails because of an r-cycle of odd length formed. **(a)** The nodes of the quasi-binary tree shown with filled circles are not taken to represent the quadrant they are in since they are not the highest (lowest) node in the upper (lower) tree. The edges on the r-cycle is shown with solid lines. **(b)** How six-coloring of an r-cycle (alternating r_1's and r_2's) fails (see the quadrant with the question mark.)

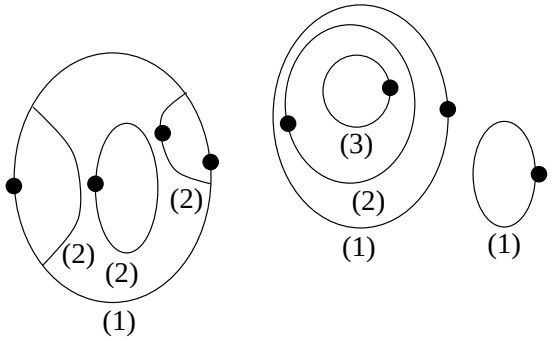


Figure 4: Assignment of level numbers (shown in parentheses) to r, g, and b cycles and the illustration of how these cycles are broken from left or right depending on the parity of their levels (recolored nodes marked with filled circles).

such a cycle as one of three kinds: a *top* node, a *bottom* node, or a (*left* or *right*) *side* node. Please refer to Figure 3.a for examples of these three kinds of nodes labeled as t, b, and s, respectively.

Lemma 3 *Let T be a quasi-binary tree whose six-coloring produces an r, g, or b cycle, say C, consisting of a top node t, a bottom node b, and side nodes s_1, ..., s_k. Suppose the quadrants associated with these nodes in the quadrilateral mesh corresponding to T are labeled q_t, q_b, and q_{s_i}, $i = 1, ..., k$, respectively. No quadrant q_{s_i} extends beyond the left or the right of q_t and q_b. That is, all quadrants q_{s_i}, $i = 1, ..., k$, are vertically sandwiched in the region bounded by the right side of q_t or q_b (whichever extends less towards the right in the mesh) and the left side of q_t or q_b (whichever extends less towards the left.) Please refer to Figure 5 for an illustration of this.*

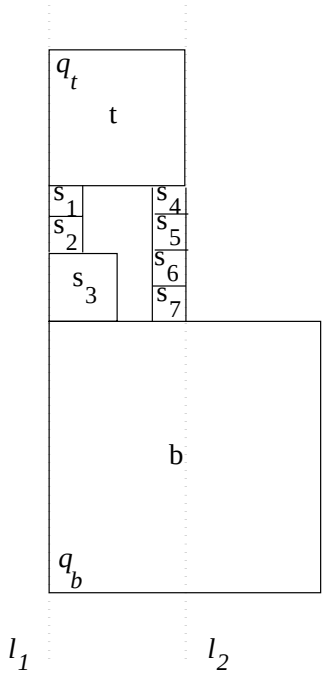


Figure 5: The part of the mesh corresponding to the r-cycle in Figure 3. The side nodes s_1,...,s_7 are bounded between the vertical lines l_1 and l_2 that are obtained by horizontal projection of quadrants q_t and q_b onto each other.

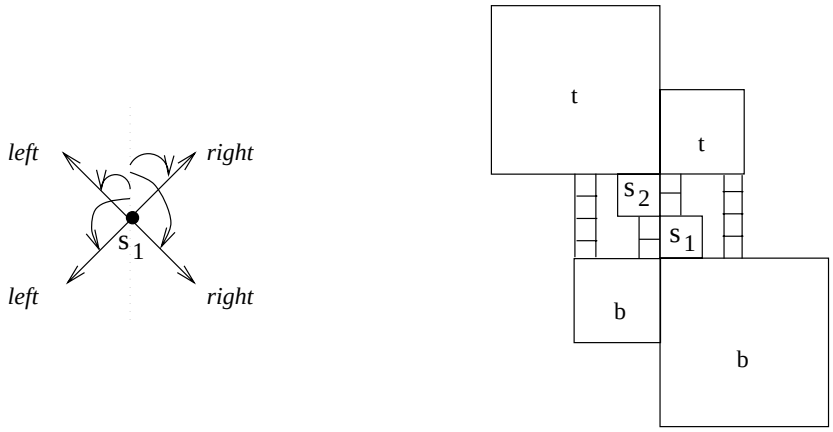
Proof: Follows from the definition of quasi-binary trees. ■

Definition 2 *Two r, g, or b cycles are called* overlapping *if one is laid completely inside another or they are of the same color and share a path. Otherwise, they are said to be* non-overlapping.

Lemma 4 *Let s_1 and s_2 be two side nodes of non-overlapping cycles C_1 and C_2 in a six-coloring, respectively. If the quadrants associated with s_1 and s_2 are adjacent, then exactly one of s_1 and s_2 is a left-side node and the other is a right-side node (please see an example of this in Figure 6.b.)*

Proof: Directly follows from Lemma 3. ■

Definition 3 *Consider two adjacent side nodes s_1 and s_2 recolored with y as discussed earlier. When going from s_1 to s_2, we are said to have taken either a* left *turn or a* right *turn as illustrated in Figure 6.*



(a) (b)

Figure 6: **(a)** Going from side node s_1 to another side node, we take either a left or a right turn depending on the region we move into. **(b)** An example of a *left* turn taken when going from left-side node s_1 to right-side node s_2.

Lemma 5 *Let s_1, s_2, and s_3 be side nodes of non-overlapping cycles C_1, C_2, and C_3 in a six-coloring, respectively. Also suppose the quadrants associated with s_1 & s_2 and s_2 & s_3 are pairwise adjacent. Going from s_1 to s_2, and then to s_3, we take alternating left and right turns, but never subsequent left or right turns.*

Proof: Follows from Lemma 4 and Definition 3. ■

Theorem 2 *Breaking (re-coloring a side node with y) r, g, and b cycles, created by six-coloring, from left or right consistently, depending on the parity of their levels as explained above might create y paths but never leads to y cycles.*

Proof by contradiction: Suppose such a y-cycle, say C, can be formed during the process of breaking these r, g, and b cycles as discussed. Let us assume C is of length k. We will name the r, g, and b cycles that these k nodes are on, C_1, C_2, ..., C_k, respectively. Now consider the way these k cycles intersect:

- *there is at least one cycle, say C_i, that intersects C at more than one node:* such an r, g, or b cycle C_i can never be formed since by definition, we break all such cycles only from one side.
- *otherwise:* in this case, no pair of cycles C_i, $i = 1, ..., k$ are overlapping. In order to complete a cycle of side nodes colored with y, we need to take subsequent left or right turns (otherwise, we can never horizontally get below our starting point) when visiting the side nodes on such a cycle; but, this is not possible by Lemma 5.

Since neither case is possible, the theorem must be true. ■

Figure 7 contains pseudo-code of an algorithm for eight-coloring quasi-binary trees based on Theorem 2. Obviously, it can be easily modified to eight-color quadtrees.

The procedure *sixcolortree* takes time linear in the size of the quasi-binary tree provided since the slight modification to the algorithm taken from [2] does not change its asymptotic time complexity. The procedure *assignlevelno* is also linear in the size of the given quasi-binary tree because the level

algorithm *eightcolortree*(*root*: **pointer_to_tree;** r, g, b, y: **color_pair**)
// Six-color given quasi-binary tree using the algorithm in [2]. //
// In the meantime, find all r, g, and b cycles of odd length, C_1, ..., C_k, along with their top and bottom nodes. //
sixcolortree(*root*, r, g, b)
// Assign level numbers to all cycles C_i, $i = 1, ..., k$. //
assignlevelno(*root*)
// Break each cycle C_i, $i = 1, ..., k$, from left or right depending on parity of their level numbers using fourth color pair y. //
breakcycles(*root*, y)
end algorithm

Figure 7: An eight-coloring algorithm.

number of a cycle C_i is equal to the number of top nodes of other cycles *dominating* the top node of C_i and this can be found in linear time using depth-first search [5]. Another simple search of the colored tree, in which any left or right side node can be chosen as the one to be recolored with the fourth color pair y, can be used to implement *breakcycles*. Therefore, *eightcolortree* takes time linear in the size of the given quasi-binary tree (or the size of the given quadtree) to compute.

3 Concluding Remarks

In this paper, we presented a linear-time eight-color algorithm for coloring *general* quadtrees which are used in solution procedures for linear elliptic partial differential equations on shared-memory parallel computers. Eight colors, or six for that matter, would normally be too many for two-dimensional problems. Thus, the number of quadrants per process would be too small for high efficiency. These algorithms may, however, be extensible to three dimensions where such granularity would not be a problem. Our intuition would suggest that four colors would suffice to color a quadtree; however, such algorithms would be much more complex and asymptotically slower. There is the possibility of creating approximate four color procedures, e.g., procedures that do not maintain an exact balance within colors.

References

[1] P. L. Baehmann, S. L. Wittchen, M. S. Shephard, K. R. Grice, and M. A. Yerry. Robust geometrically based, automatic two-dimensional mesh generation. *Int. J. Num. Meths. Engng.*, 24:1043–1078, 1987.

[2] M. Benantar. *Parallel and Adaptive Algorithms for Elliptic Partial Differential Systems.* PhD thesis, Department of Computer Science, Rensselaer Polytechnic Institute, Troy, NY 12180, 1992.

[3] M. Benantar, R. Biswas, J. E. Flaherty, and M.S. Shephard. Parallel computation with adaptive methods for elliptic and hyperbolic systems. *Comput. Meths. Appl. Mech. Engng.*, 82:73–93, 1990.

[4] M. Benantar, J. E. Flaherty, and M. S. Krishnamoorthy. *Coloring Procedures for Finite Element Computation on Shared-Memory Parallel Computers.* AMD-Vol. 157, Adaptive, Multilevel, and Hierarchical Computational Strategies, A. K. Noor Ed., ASME, New York, 1992.

[5] E. M. Reingold, J. Nievergelt, and N. Deo. *Combinatorial Algorithms : Theory and Practice.* Prentice-Hall, 1977.

[6] M. S. Shephard and M. K. Georges. Automatic three-dimensional mesh generationby the finite octree technique. *Int. J. Num. Meth. Engng.*, 32:709–739, 1991.