Available online at www.sciencedirect.com
ScienceDirect
Theoretical Computer Science 363 (2006) 11 – 17
Theoretical Computer Science
www.elsevier.com/locate/tcs

Coloring octrees

Udo Adamy[a], Michael Hoffmann[a], József Solymosi[b], Miloš Stojaković[a,*,1]

[a] *Institute for Theoretical Computer Science, ETH Zurich, CH-8092 Zurich, Switzerland*
[b] *Department of Mathematics, University of British Columbia, Vancouver, Canada*

Abstract

An octree is a recursive partition of the unit cube, such that in each step a cube is subdivided into eight smaller cubes. Those cubes that are not further subdivided are the leaves of the octree. We consider the problem of coloring the leaves of an octree using as few colors as possible such that no two of them get the same color if they share a facet. It turns out that the number of colors needed depends on a parameter that we call unbalancedness. Roughly speaking, this parameter measures how much adjacent cubes differ in size. For most values of this parameter we give tight bounds on the minimum number of colors, and extend the results to higher dimensions.

Keywords: Quadtree; Octree; Cube; Coloring

1. Introduction

Octrees are a fundamental data structure to store any three-dimensional data in such a way that search and insert operations can be performed efficiently. Therefore, octrees are frequently used, for instance in computational geometry and computer graphics (see e.g. [5]).

An octree is a binary partition of the unit cube, with some additional constraints. More precisely, an octree is obtained by starting from a single cube, and recursively subdividing cubes into eight smaller cubes of equal size. The structure of an octree can be seen as a tree, where each node represents a cube in the subdivision process. If a cube is further subdivided, the corresponding node in the tree has eight children. The cubes associated to children nodes are also referred to as *subcubes* of the cube that corresponds to the parent node. An example of an octree is shown in Fig. 1(a).

We say that two cubes in an octree are *facet-adjacent*, iff they share a two-dimensional facet (that is, a facet of one of the cubes contains a facet of the other cube). Similarly, two cubes are called *edge-adjacent*, iff they share a one-dimensional face, and *vertex-adjacent*, iff they share a zero-dimensional face. Whenever the term "*adjacent*" appears without further qualification in the following, we refer to facet-adjacency.

The purpose of this paper is to analyze the problem of coloring the leaves of octrees (i.e., those cubes that are not further subdivided) using as few colors as possible, such that no two adjacent cubes receive the same color. Fig. 1(b)

* Corresponding author.
E-mail addresses: adamy@inf.ethz.ch (U. Adamy), hoffmann@inf.ethz.ch (M. Hoffmann), solymosi@math.ubc.ca (J. Solymosi), smilos@inf.ethz.ch, milosst@eunet.yu (M. Stojaković).

[1] Supported by the joint Berlin-Zürich graduate program "Combinatorics, Geometry, and Computation", financed by the German Science Foundation (DFG) and ETH Zurich.

0304-3975/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2006.06.021

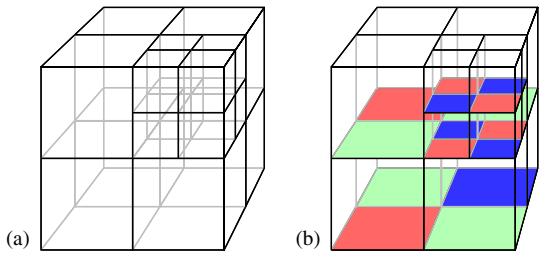

Fig. 1. An octree: (a) structure; (b) a proper 3-coloring.

shows a proper coloring of an octree using three colors. In all figures appearing in this paper we indicate the color of a cube by coloring its bottom facet.

In an octree, adjacent cubes may differ in size. To capture this property, we introduce a parameter that measures the unbalance. An octree is said to be *k-unbalanced*, if the edge lengths of neighboring cubes differ by at most a factor of 2^k. A 0-unbalanced octree is called *balanced*. If there is no condition on the balance of adjacent cubes, we call an octree ∞-unbalanced, or just unbalanced. As it turns out, the number of colors needed for coloring an octree depends on this parameter.

1.1. Related work

The problem of coloring octrees has been formulated by Benantar et al. [1]. It is a natural generalization of the corresponding problem for quadtrees, which are the two-dimensional counterparts of octrees obtained by recursively subdividing a square into four congruent smaller squares.

For quadtrees the coloring problem appears in the context of solving linear elliptic partial differential equations on shared-memory parallel computers [2]. The basic idea is the following. If the squares of a quadtree are colored with respect to vertex-adjacency, then the squares that belong to one particular color class are spatially well separated and can therefore be processed in parallel without conflict. They present a linear-time algorithm for coloring 1-unbalanced quadtrees with at most six colors. For unbalanced quadtrees an algorithm using at most eight colors is given by Benantar et al. [1].

Recently, Eppstein et al. [3] proved several results on coloring quadtrees with respect to edge-adjacency. They prove that 1-unbalanced quadtrees can be colored with three colors, and that unbalanced quadtrees can be colored using at most four colors. Both bounds are tight in the sense that some quadtrees require this many colors. They also improved the bound for coloring the squares of unbalanced quadtrees with respect to vertex-adjacency from eight to six colors, and constructed a quadtree that requires at least five colors.

1.2. Our results

In this paper we mainly concentrate on octree coloring problems with respect to facet adjacency, and quadtree coloring problems with respect to edge-adjacency.

However, recursive subdivisions of cubes are obviously not restricted to dimensions two and three. Since most of our results naturally extend to cubes of arbitrary dimension, we present them in a more general setting. The following definition provides the generalization of octrees to arbitrary dimensional structures, which we call 2^d-trees.

Definition 1. For $d \in \mathbb{N}$, a 2^d*-tree* is a data structure formed by starting from a single d-dimensional cube and recursively subdividing d-dimensional cubes into 2^d smaller d-dimensional cubes of equal size.

Let k be a non-negative integer. A 2^d-tree is called *k-unbalanced* if any two cubes that share a facet (that is, a facet of one of the cubes contains a facet of the other cube) are within a factor of 2^k in edge length.

By coloring a 2^d-tree we mean coloring its *leaves*, that is, those cubes that are not further subdivided. The coloring of a 2^d-tree is called proper, iff any two cubes that share a facet are colored with different colors. We say that a 2^d-tree is *k-colorable*, iff there exists a proper coloring for its leaves using at most k colors.

Table 1
Minimal number of colors needed to color a k-unbalanced 2^d-tree

$d\backslash k$	0	1	2	3	4	5	6	∞
2	2	**3**	4	4	4	4	4	**4**
3	2	4		5	5	5	5	5
4	2	4			6	6	6	6
5	2	4	$\leqslant 6$			7	7	7
6	2	4	$\leqslant 6$				8	8
7	2	4	$\leqslant 6$	$\leqslant 8$				9

The first observation that can be made is that a balanced 2^d-tree is 2-colorable, for every $d \in \mathbb{N}$. Indeed, since all cubes in a balanced 2^d-tree have the same size, it can be represented as a finite set of cells of the axis parallel integer grid subdivision in $\mathbb{R}^d$. The cells of this subdivision can be colored with two colors, by coloring all cells that have even sum of their coordinates with color one, and all the others with color two. We will refer to this coloring as the *coloring in checkered fashion*.

Our results are presented in Table 1. For each d and k, the corresponding entry c in the table gives the minimal number of colors needed to color a k-unbalanced 2^d-tree. This means that, on one hand, we provide an algorithm for coloring k-unbalanced 2^d-trees with c colors and, on the other hand, we give an example of a k-unbalanced 2^d-tree which cannot be colored with $c-1$ colors.

The paper of Eppstein et al. [3] already settles this question for $d=2$, $k=1$, and $d=2$, $k=\infty$. These two results are marked bold in the table.

Obviously, the values in the rows of the table are non-decreasing. The same holds for columns, since by using the structure of a k-unbalanced 2^d-tree which is not c-colorable one can easily construct a k-unbalanced 2^{d+1}-tree which is not c-colorable. These properties set upper and lower bounds in the cases in which we could not determine the exact number of colors needed (empty cells).

2. Lower bounds

Eppstein et al. [3] gave a neat algorithm to color any 1-unbalanced quadtree using at most three colors. Moreover, the color of each square in the quadtree can be set without knowledge of the subdivision.

For 1-unbalanced octrees, it turns out that three colors are not enough. For the proof of this claim we need the following two lemmata.

Lemma 2. *Let A, B and C be congruent cubes such that both B and C share a facet with A, and B and C share an edge. Moreover, let the cube A be further subdivided into eight cubes* (*see Fig.* 2(a)).

If these cubes are properly colored with three colors, then B and C have to get the same color, and the six subcubes of A that are adjacent to B or C have to be colored in checkered fashion with the remaining two colors.

Proof. Suppose that the arrangement of cubes in Fig. 2 is properly 3-colored. We denote the two subcubes of A that are adjacent to both B and C by A_1 and A_2. Since B, A_1 and A_2 are pairwise adjacent they all have to be colored with different colors. We can assume without loss of generality that B is colored with color 1, A_1 with color 2, and A_2 with color 3. Since C is adjacent to both A_1 and A_2, it has to be colored with color 1. Hence, each of the remaining four subcubes of A that are adjacent to B or C has to be colored either with color 2, if it is adjacent to A_2, or with color 3, if it is adjacent to A_1. That gives exactly the coloring of the six subcubes of A by colors 2 and 3 in checkered fashion, as shown in Fig. 2(b). □

Lemma 3. *Let A, B, C and D be congruent cubes, such that all of B, C, and D share a facet with A, and that B and C do not share an edge. Moreover, let cube A be further subdivided into eight cubes* (*see Fig.* 3).

If these cubes are properly colored with three colors, then B, C, and D get the same color, and the eight subcubes of A are colored in checkered fashion with the remaining two colors.

Proof. Apply Lemma 2 twice, once to the cubes A, B, and D, and then to the cubes A, C, and D. □

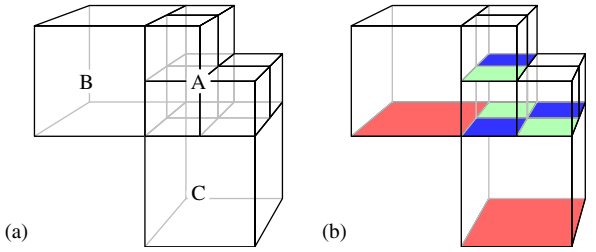

Fig. 2. Illustration for Lemma 2: (a) configuration; (b) forced coloring.

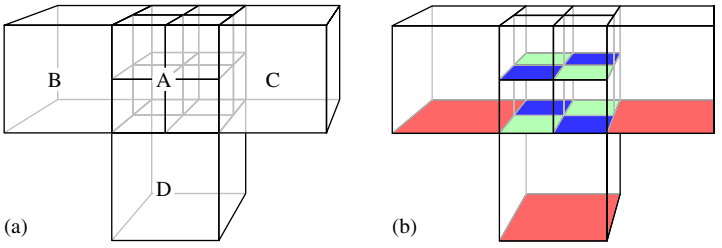

Fig. 3. Illustration for Lemma 3: (a) configuration; (b) forced coloring.

Note that a collection of cubes can be colored in checkered fashion with two fixed colors in *two* different ways. Using the configurations from the last two lemmata as building blocks, we give a lower bound on the number of colors needed to properly color 1-unbalanced octrees.

Theorem 4. *There exists a* 1*-unbalanced octree that is not* 3*-colorable.*

Proof. We show that the 1-unbalanced arrangement of cubes presented in Fig. 4 cannot be colored with three colors. Some cubes of the octree are omitted, since they will not be needed for the analysis. However, they can be easily added without violating the 1-unbalancedness. Note that the cubes that appear in the example are of three different sizes—we will refer to them as big, medium, and small cubes. Denote the two big cubes that are subdivided by A and B, and denote the two medium subcubes of A that are subdivided into small cubes by A_1 and A_2, as shown in Fig. 4(a).

Now, suppose there is a proper 3-coloring. First, we look at the two adjacent big cubes at the back of the configuration, behind A and B. We denote the one behind A by C, and the one behind B by D. They have to be colored with different colors. We can assume without loss of generality that D is colored with color 1 and C is colored with color 2.

Then, apart from A there are three big cubes adjacent to the subdivided cube B. This arrangement enables us to use Lemma 3. Since we know that the cube D has color 1, the other two big cubes should also be colored with color 1. Further, the subcubes of B have to be colored in checkered fashion with colors 2 and 3.

Finally, the subcubes of A (except A_1 and A_2) together with the cube C and the big cube below A form the configuration from Lemma 2. Since C is colored with color 2, the other one should have color 2 as well. The six medium subcubes of A have to be colored in checkered fashion with colors 1 and 3.

It remains to prove that this coloring is not proper. Consider the subdivided cube A_1 and its three neighboring medium cubes that are shaded in Fig. 4(b). By Lemma 3 the medium cubes above and below A_1 both have the same color. Similarly, the medium cubes above and below A_2 both have the same color. But, we already know that the two cubes above A_1 and A_2 are colored in checkered fashion using colors 2 and 3, whereas the two cubes below A_1 and A_2 are colored in checkered fashion using colors 1 and 3. Contradiction. □

Theorem 5. *There exists a* d*-unbalanced* 2^d*-tree that cannot be* $(d+1)$*-colored.*

Proof. We are going to provide examples of a 2-unbalanced quadtree that requires four colors and a 3-unbalanced octree that requires five colors. The latter example can be generalized to prove the statement in higher dimensions.

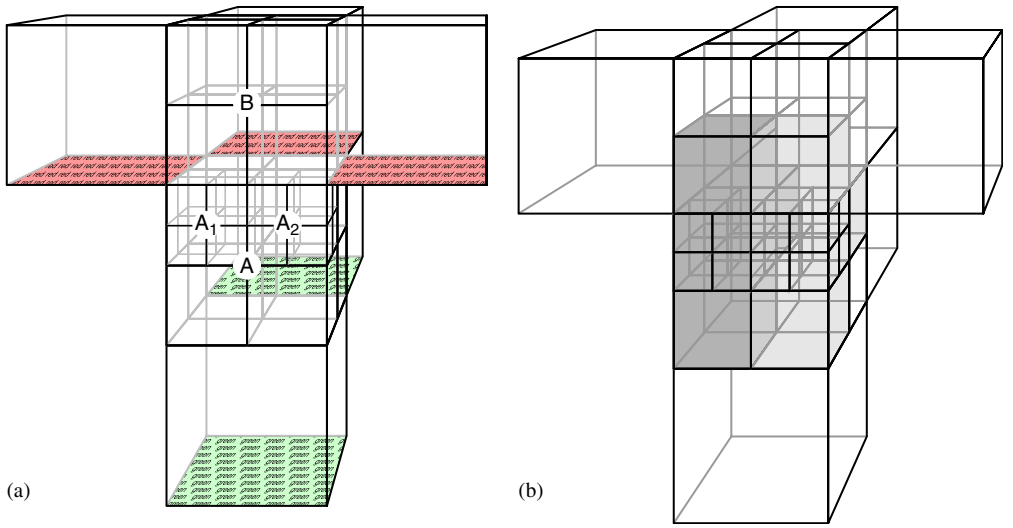

Fig. 4. A 1-unbalanced octree that is not 3-colorable: (a) configuration, with the forced coloring of all big cubes; (b) two configurations to apply Lemma 2.

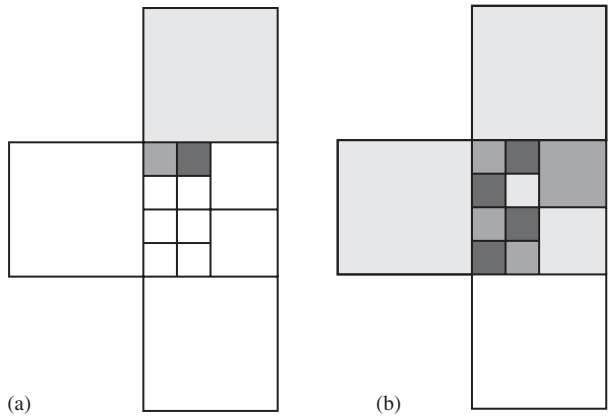

Fig. 5. A 2-unbalanced quadtree that is not 3-colorable: (a) initial 3-clique; (b) remaining colors are forced.

First, we consider the 2-unbalanced quadtree as shown in the Fig. 5. Note that some squares of the depicted quadtree are omitted, as they will not be needed for the analysis. They can be easily added without violating the 2-unbalancedness.

Assume that this quadtree can be 3-colored. The three squares marked in Fig. 5(a) are pairwise adjacent, and, hence, have to be colored differently. If we proceed to color those squares for which the color is uniquely determined, we end up with the coloring depicted in Fig. 5(b). Obviously, this cannot be extended to the last square. Contradiction.

Next, we exhibit a 3-unbalanced octree T that cannot be colored with four colors. The construction is shown in Fig. 6(a). Note that it contains a blow-up of the quadtree from Fig. 5 to three dimensions as a ground structure.

Assume that this octree can be 4-colored. The four cubes marked in Fig. 6(a) are pairwise adjacent, and, hence, have to be colored differently. If we proceed to color those cubes for which the color is uniquely determined, we end up with the coloring depicted in Fig. 6(b). Obviously, this cannot be extended to the last cube. Contradiction.

In the following, we use the octree T to construct corresponding examples in higher dimensions. First, we show how to construct a 4-unbalanced 2^4-tree T'. Start with a four-dimensional cube C. A copy of T is then placed onto an arbitrary facet of C. We extend ("grow") each of the three-dimensional cubes in the copy of T to the four-dimensional cube, such that it does not overlap with the cube C.

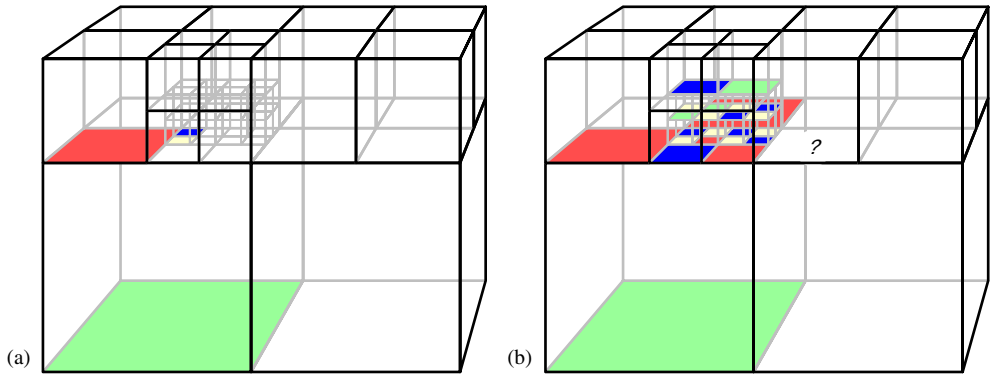

Fig. 6. A 3-unbalanced octree that is not 4-colorable: (a) initial 4-clique; (b) remaining colors are forced.

The obtained configuration of four-dimensional cubes can be easily completed (by adding more cubes) to a 2^4-tree T'. Since the octree T has only four levels of subdivision, T' is a 4-unbalanced 2^4-tree.

The cubes in T' obtained from cubes in T are adjacent to one another the same way as in T, and in addition they are all adjacent to the cube C. Since T cannot be colored with four colors, T' cannot be colored with five colors.

This construction can be repeated in the same fashion, each time increasing the dimension by 1. This way, for every $d \geqslant 3$ we obtain a d-unbalanced 2^d-tree that cannot be $(d+1)$-colored. □

3. Algorithms for coloring 2^d-trees

We proved that at least four colors are needed to color a 1-unbalanced octree. Somewhat surprisingly, four colors are enough even for a 1-unbalanced 2^d-tree of arbitrary dimension.

Theorem 6. *Every k-unbalanced 2^d-tree is $(2k+2)$-colorable, for $k \geqslant 0$, $d \geqslant 2$.*

Proof. We color the cubes of the k-unbalanced 2^d-tree according to their size—all the cubes with edge length 2^{-l} are colored in the checkered fashion using the colors $(2l \bmod 2k+2)$ and $(2l+1 \bmod 2k+2)$.

It remains to be shown that this coloring is proper. Two cubes of different size that are colored with the same color have edge ratio at least 2^{k+1}, and therefore they cannot be adjacent in a k-unbalanced 2^d-tree. If two adjacent cubes have the same size then they are colored with different colors by definition of the coloring in checkered fashion. Hence, the coloring is proper. □

The upper bound we just obtained is tight in some cases, but not in general. The theorem below provides a better upper bound, if $d < 2k$.

Theorem 7. *Every 2^d-tree is $(d+2)$-colorable, for $d \geqslant 2$.*

Proof. We follow the idea presented in [3]. First, we assign an arbitrary color to the unit cube. Then, we subdivide it recursively assigning colors to the newly obtained cubes—in step t we color all cubes of edge size 2^{-t}. Note that this way cubes are colored even if they are subdivided. Eventually, we obtain a coloring of all leaves of the 2^d-tree.

For each subdivision of a cube of color c into 2^d subcubes, we color them as follows. A pairwise non-adjacent half of them (one color class of the checkered 2-coloring) is colored with color c. After that each of the cubes in the other half has d neighbors of color c and d more neighbors (since at that moment no cube of smaller size has been colored). Altogether, there are at most $d+1$ different colors in its neighborhood. Hence, one of the $d+2$ colors can be used to color it. □

Note that the existence of a coloring for the case $d = 2$ (quadtrees) follows from the four-color theorem for planar graphs [4].

4. Open problems

It remains to determine the exact number of colors needed to color 2-unbalanced octrees (is it four or five?). We conjecture that four colors are enough.

Another interesting direction is to investigate the octree coloring problems with respect to edge/vertex-adjacency, where to the best of our knowledge no results have been obtained so far.

References

[1] M. Benantar, U. Dogrusöz, J.E. Flaherty, M.S. Krishnamoorthy, Coloring quadtrees, Manuscript, 1996.
[2] M. Benantar, J.E. Flaherty, M.S. Krishnamoorthy, Coloring procedures for finite element computation on shared-memory parallel computers, in: Adaptive, Multilevel, and Hierarchical Computation Strategies, AMD, Vol. 157, New York, 1992, pp. 435–449.
[3] D. Eppstein, M.W. Bern, B. Hutchings, Algorithms for coloring quadtrees, Algorithmica 32 (1) (2002) 87–94.
[4] N. Robertson, D. Sanders, P. Seymour, R. Thomas, The four-colour theorem, J. Combin. Theory Ser. B 70 (1) (1997) 2–44.
[5] H. Samet, The quadtree and related hierarchical data structures, ACM Comput. Surveys (CSUR) 16 (2) (1984) 187–260.