Treemaps for Search-Tree Visualization

Rémi Coulom

July, 2002

Abstract

Large Alpha-Beta search trees generated by game-playing programs are hard to represent graphically. This paper describes how treemaps can be applied to the visualization of these trees. The principle of treemaps is presented, and difficulties of its application to the particular structure of search trees are reviewed. An original "ordered squarified" layout is proposed. It has been implemented in a freely available program, that can be easily re-used by computer-chess programmers.

Introduction

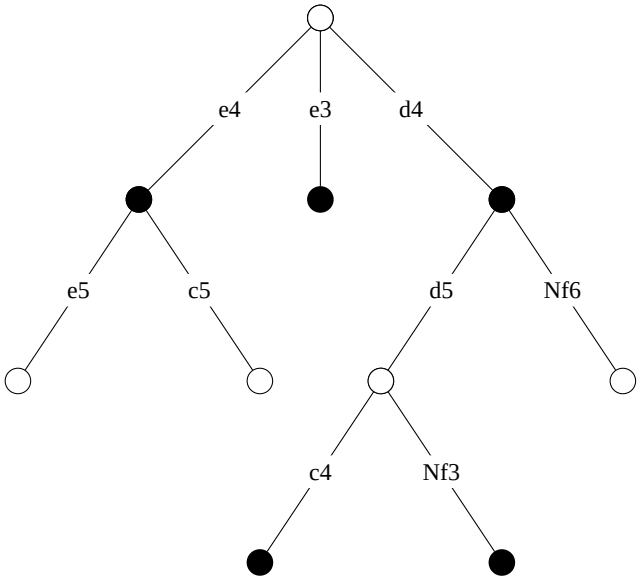
Many game-playing programs are based on the alpha-beta algorithm [5] and can explore huge search trees in a few seconds. Being able to visualize these large trees could be of a great help to tune search algorithms. Unfortunately, they may have millions of nodes (up to 200 million nodes per second for Deep Blue [3]), and traditional node-and-link diagrams do not allow to represent them conveniently.

Treemaps, developed by Shneiderman and Johnson [6, 4], are particularly well adapted to this problem. They are extremely efficient to represent extensive attributes (size, cost, value) of elements organized in a hierarchy. Their first application was the visualization of disk usage in a large directory structure. This paper shows how the visualization of search trees can also take advantage of treemaps properties.

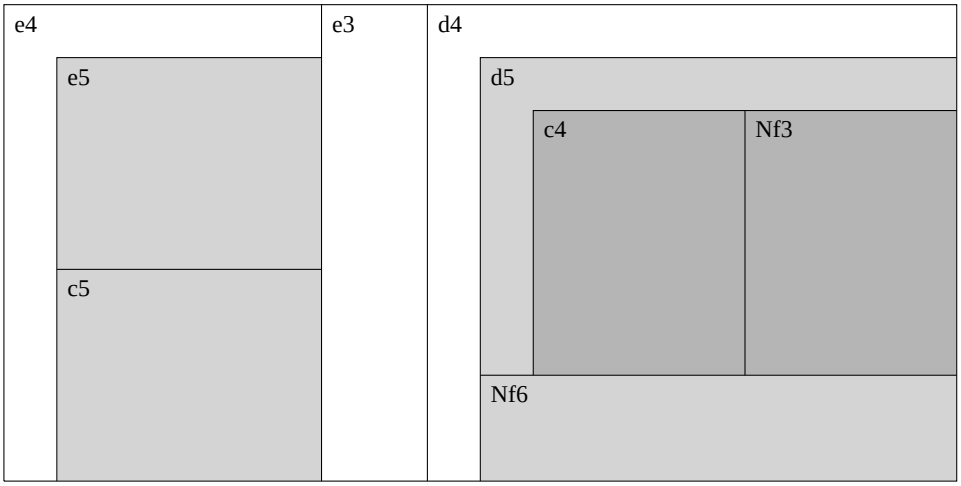
The first section of this paper presents the general basic principle of treemap visualization. The second part deals with aspects of this problem that are specific to alpha-beta trees. In particular, obtaining a good aspect ratio for rectangles while keeping move-ordering information is a key issue. An "ordered squarified" layout is proposed to solve this problem.

1 Principle

Figure 1 illustrates the principle of treemaps. The whole tree is represented as a rectangle. Each sub-tree is represented as a sub-rectangle of its parent rectangle. At the first level of the hierarchy, the whole rectangle is split vertically. Then, sub-rectangles are split horizontally. Sub-sub-rectangles are split vertically, and so on. Each splitting is done so that the area covered by a rectangle is proportional to the number of nodes it contains.



(a) A traditional node-and-link representation of a tree



(b) The corresponding treemap

Figure 1: Principle of treemap visualization

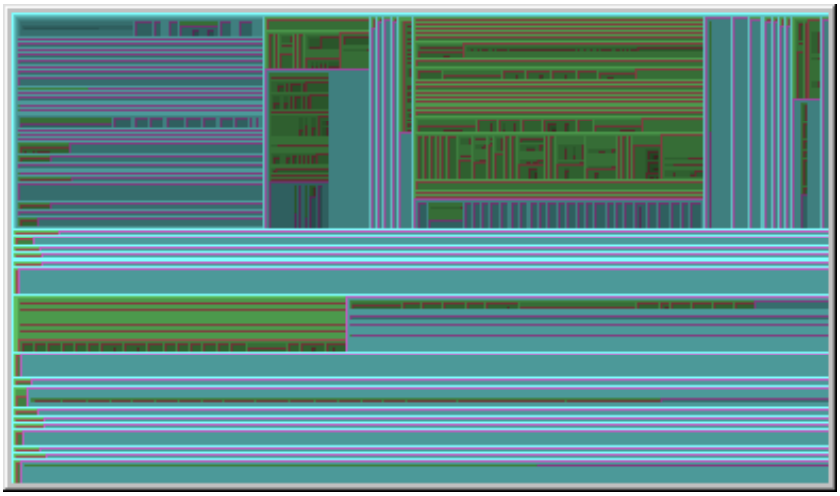


Figure 2: Poor aspect ratios with the standard slice-and-dice layout

1.1 Color Scheme

Besides the size of each node, which is represented by the area of its rectangle, it is possible to represent other information by selecting the color used to fill the rectangle. This information can be of any kind: extension, cutoff, check, depth, player to move, hash-table hit, *etc.*

In the screen captures presented in this paper, brightness indicates the depth of the node. More precisely, the color level is equal to

$$\text{level} = \frac{\text{flatness}}{\text{depth} + \text{flatness} - 1},$$

where "flatness" is a constant parameter. This formula gives an effect of depth to the treemap. Beta cutoffs have a higher red component. Nodes wasted because of a bad move ordering have a higher green component.

2 Efficient Layout Algorithm

The simple slicing algorithm that was presented in the previous section has serious shortcomings. When visualizing search trees, more sophisticated techniques are required. This section explains why, and what better layout algorithms can be used.

2.1 Aspect Ratio Problems

The basic layout presented in the previous section is called "slice and dice". Regardless of the shape of the parent rectangle, it is sliced vertically or horizontally, depending on the depth of the node. Unfortunately, this causes a lot of problems for alpha-beta trees. Very often, a single move causes a beta cutoff, and is followed by a full-width extension of the tree. Each of these moves is then followed by a single move that fails high and so on. As a consequence, rectangles are always sliced in the same direction and become extremely thin. Figure 2 illustrate this with a chess tree generated from the starting position.

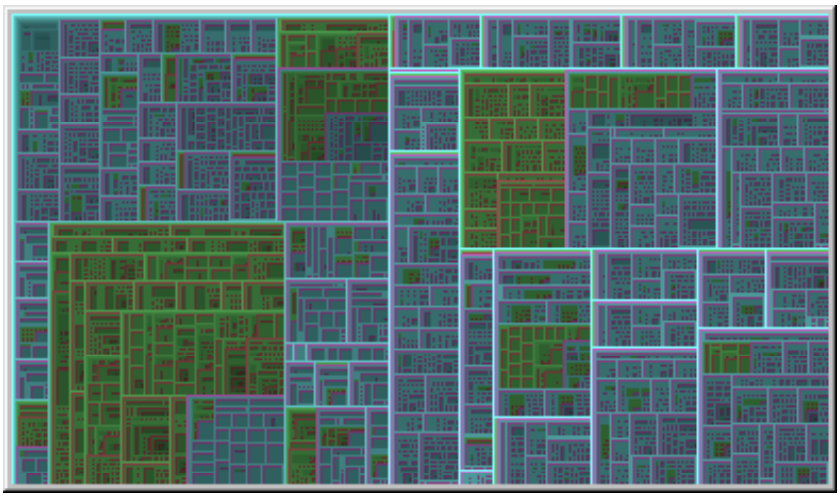


Figure 3: An ordered squarified treemap

This problem can be solved by using more clever algorithms to split the area of one rectangle into sub-rectangles. In particular, Bruls *et al.* proposed a "squarified" layout algorithm [2]. The principle of this method consists in enforcing an aspect ratio that is as close as possible to 1.

2.2 Move Ordering

A major difficulty with classical squarified treemaps is that they loose node-ordering information that is extremely important to analyze the shape of search trees. In order to solve this problem, a number of alternative layout algorithms have been proposed [7, 1]. The strip treemap algorithm [1] is an interesting solution, that is similar to squarified treemaps, except that rectangles are not sorted by size and all the strips are horizontal.

A simpler way to keep ordering information consists in using the squarified treemap layout, without sorting nodes by size. This is likely to produce better aspect ratios that the strip-treemap algorithm, and does not require any look-ahead. Figure 3 shows a treemap produced by this method.

Move-ordering information can be retrieved from the layouts produced by this method, though it is not as straightforward as in the case of the strip-treemap algorithm. An example of how the order of moves can be obtained is shown on Figure 4. The first move is always at the top left, and the last move is always at the bottom right. So, on Figure 4, e4 is the first move. In the remaining rectangle, the same principle is applied recursively, so h3 is the next move. h3 is followed by g4, g3, and f4, because these move form the only strip that is aligned with h3. The remaining rectangle starts with h4 at its top left, and e3 is the only node alighned with h4 in an horizontal strip. The same principle is applied again and the following moves are d4, d3, c4, b4, b3, a4, a3, Nf3, Nh3, Na3, and Nc3.

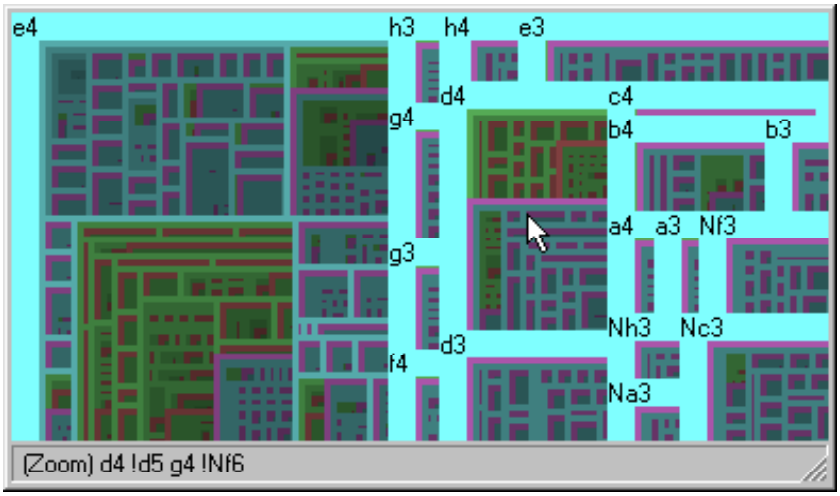


Figure 4: The order of labeled moves is: e4, h3, g4, g3, f4, h4, e3, d4, d3, c4, b4, b3, a4, a3, Nf3, Nh3, Na3, Nc3

2.3 Layout Stability

In order to explore the search tree, the user of the treemap visualization tool can zoom in and out, or resize the window. The consequence is a change in the aspect ratio of the rectangle, which may produce a completely different layout. Figure 5 illustrates this phenomenon after zooming into the pointed node of Figure 4. This change can be confusing, and it is desirable to find ways to prevent it. So, an option to lock the layout has been provided, which results in Figure 6.

Conclusion

Some ideas to build a tool for search-tree visualization based on the treemap technique have been presented. In particular, the ordered squarified treemap layout algorithm has been proposed. It is well adapted to deal with the aspect-ratio and move-ordering issues that arise with alpha-beta trees.

The program developed during these experiments is freely available from the author's home page, with source and documentation. It can read any chess tree in PGN format. This way, other programmers should be able to easily export their search trees to this tool.

Many improvements to the current program could be made. In particuliar, it would be nice to have a graphical board in parallel with the treemap that would display the position of the currently selected node. Also, using it to perform a graphical "`diff`" between two chess trees might be a good way to visualize the effect of changing an heuristic in the search algorithm.

References

[1] Benjamin B. Berderson, Ben Shneiderman, and Martin Wattenberg. Ordered and quantum treemaps: Making effective use of 2D space to display hierarchies. In

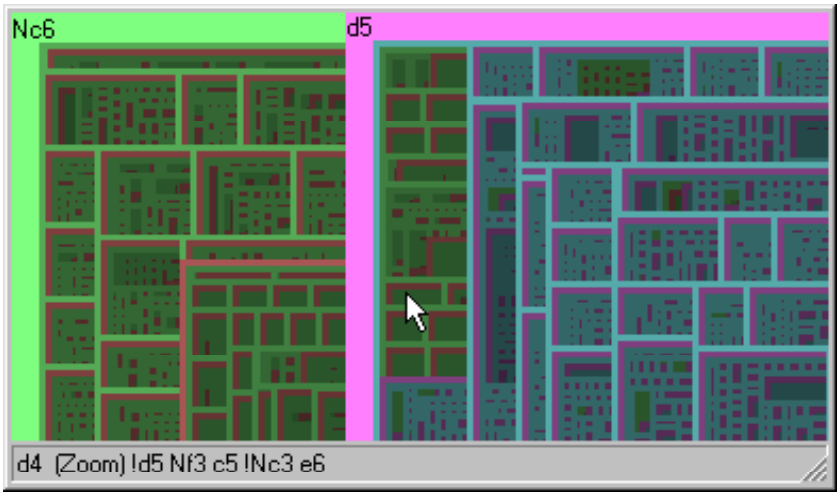


Figure 5: The Layout can be lost when zooming in

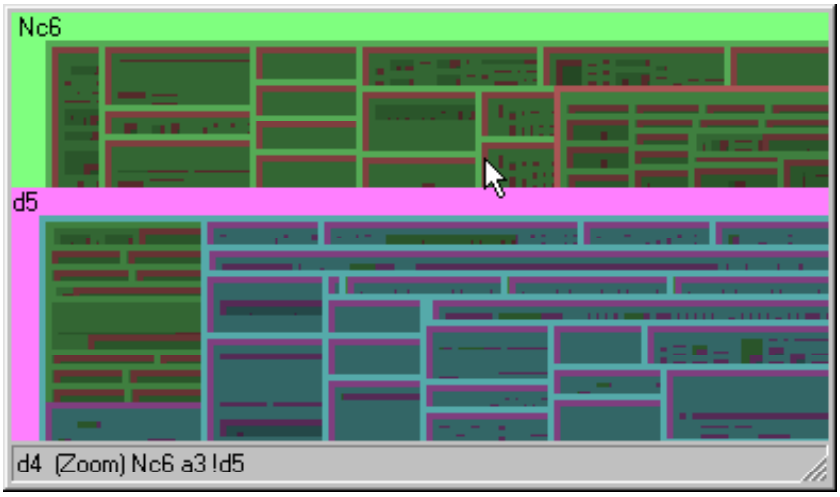


Figure 6: Locking the layout produces poorer aspect ratios, but keeps a stable look of the tree representation when zooming.

ACM Transactions on Computer Graphics. 2002.

[2] Mark Bruls, Kees Huizing, and Jarke J. van Wijk. Squarified treemaps. In W. de Leeuw and R. van Liere, editors, *Proceedings of Joint Eurographics and IEEE TCVG Symposium on Visualization*, pages 33–42. Springer, Vienna, 2000.

[3] Murray Campbell, A. Joseph Hoane Jr., and Feng-hsiung Hsu. Deep blue. *Artificial Intelligence*, 134:57–83, 2002.

[4] Brian Johnson and Ben Shneiderman. Treemaps: a space-filling approach to the visualization of hierarchical information structure. In *Proceedings of the second International IEEE Visualization Conference*, pages 284–291, October 1991.

[5] Donald E. Knuth and Ronald W. Moore. An analysis of alpha-beta pruning. *Artificial Intelligence*, 6:293–326, 1975.

[6] Ben Shneiderman. Tree visualization with tree-maps: A 2-d space-filling approach. *ACM Transactions on Graphics*, 11(1):92–99, January 1992.

[7] Ben Shneiderman and Martin Wattenberg. Ordered treemap layouts. In *Proceedings of IEEE Symposium on Information Visualization*. IEEE Press, Los Alamitos, CA, 2001.