Improving the Visualization of Hierarchies with Treemaps: Design Issues and Experimentation

David Turo
tur@cs.umd.edu
Department of Computer Science
University of Maryland, College Park, MD 20742

Brian Johnson
brianj@cs.umd.edu

Abstract

Controlled experiments with novice treemap users and real data highlight the strengths of treemaps and provide direction for improvement. Issues discussed include experimental results, layout algorithms, nesting offsets, labeling, animation and small multiple displays. Treemaps prove to be a potent tool for hierarchy display. The principles discussed are applicable to many information visualization situations.

1 Introduction

Treemaps are a novel method for presenting large hierarchical information spaces on planar display areas of limited size [9, 13]. A treemap is generated by recursively slicing the screen into rectangular bounding boxes to convey global structure (hierarchy). Individual node information is presented through display attributes such as bounding box size and fill color. Two families of treemap algorithms have been developed for tiling 2-D planes: slice-and-dice, which alternates between vertical and horizontal screen slices (Figures 2,3,4), and top-down, which slices in only one dimension, either horizontal or vertical (Figure 1). Treemaps combine features of multivariate coding and display layout to present hierarchies in a richly visual environment which fosters relative comparison of structures in the hierarchy.

Two experiments, one with UNIX users, and one involving employees of Genie, a consumer information service run by GE Information Services, drive the discussion presented herein. Section 2 of this paper discusses the two primary treemap layout algorithms. Section 3 describes improvements to the slice-and-dice algorithm to further convey information about the hierarchy. Section 4 offers a sampling of current research directions. Section 5 describes the experiments conducted with treemaps. Section 6 is the conclusion.

2 Partitioning Algorithms

The partitioning of rectangular screen region into a treemap can take one of two approaches: the slice-and-dice approach as previously described in [9] and the top-down approach [12], which is discussed below.

2.1 Top-Down

Development of the top-down algorithm was motivated by the desire to preserve the structure (and user familiarity) of traditional tree diagrams, which flow from one side of the screen to the other (usually from top to bottom). The algorithm slices the rectangular region along one dimension and flows from the root (on one side of the screen) to the leaves (on the opposite side). The algorithm relies upon each node in the tree having a pre-determined weight, dependent upon a domain-specific attribute. An overview of the basic algorithm is given below using a top to bottom flow.

TOP-DOWN ALGORITHM

1. The bounding box of the root node is the entire treemap display area. Make the root node the "current node".
2. Divide the bounding box of the current node proportionally along its vertical axis using its weight compared with the sum of the weights of its children. This produces an upper region for the current node and a lower region for its children.
3. Partition the lower region along the horizontal axis among the children based on their weights, creating a bounding box for each child.
4. Iteratively make each child the "current node" and go to step 2.

Essentially, the horizontal axis is recursively sliced and divided among the children. All of the leaf nodes of the tree eventually "touch" the bottom of the display. Figure 1 illustrates a traditional tree structure overlaid on its top-down representation. Offsets are used to emphasize hierarchy structure.

The area of each bounding box is determined by the weight attribute. For example, if the hierarchy in Figure 1 was an organization chart with size representing salary and color representing years of service (the lighter grays representing the most years), large light gray boxes would indicate long-term employees who are well paid.

This concept of emphasizing importance through size is similar to the fisheye concept [7], though there are multiple points of interest in treemaps.

As the top-down approach to tiling planar areas limits recursive subdivision to one dimension, acceptable results are produced only for hierarchies of limited size. Testing indicates hierarchies of around 100 to 200 nodes overwhelm the top-down algorithm on typical displays with 640 x 480
resolution. As an example, the GEnie hierarchy used in the
treemap experiment contained 120 nodes (products) in two-
levels (product manager and product type). This hierarchy
could not be displayed using the general top-down algorithm
due to limited horizontal resolution; a modified top-down
approach solved this by partitioning the vertical axis at the
final level. This modified algorithm (displayed in Figure 5)
works well, but it is not generalizable to hierarchies that are
not of a uniform, fixed depth.

The main benefit of the top-down design, therefore, is its
ability to conform to traditional tree diagram conventions.
With small hierarchies, traditional tree diagrams may be used
in conjunction with top-down treemaps as in Figure 1, fostering
comparative analysis while preserving traditional diagram-
matic notation.

2.2 Slice-and-Dice

The slice-and-dice algorithm avoids the problems of the
top-down algorithm by recursively partitioning the planar
display area along both dimensions. Much larger hierarchies,
greater than 1,000 nodes, can be displayed (Figure 8). The
slice-and-dice algorithm is presented and discussed in detail
in [9].

A slice-and-dice treemap presents the hierarchy as a series
of recursively-drawn bounding boxes, sliced alternatively
vertically and horizontally. Figure 3 displays the same hier-
archy as in Figure 1, only drawn using the slice-and-dice
algorithm.

Because of the quantity of information that can be pre-
sented, large treemap diagrams should be thought of as
powerful visualization tools requiring a degree of familiarity.
Since the slice-and-dice treemap proves to be the most viable
for display of large hierarchies, all further discussion of
treemaps will assume partitioning along both dimensions.

2.3 Treemap Display Limitations

All static hierarchy presentations have limits as to the
quantity of information they are capable of presenting on a
finite display space. When these limits are reached, naviga-
tional techniques such as scrolling or panning must be used,
creating the potential for loss of context [11]. Common char-
acter-based applications use a set number of lines to display
the hierarchy. Graphical tree diagrams have more leeway:
depending upon the drawing algorithm and the size of the
display space, a hundred or so nodes can be adequately
represented on screen without the need for panning or zoom-
ing.

More advanced graphical diagrams such as cone trees [11]
increase the display limit through the use of a virtual third
dimension at the expense of increased navigation (in this

(See color plates, p. CP-16.)
The number of nodes that can be displayed by a treemap can be an order of magnitude greater than traditional graphi-
cal tree diagrams. This is the result of the tiling approach
which packs the display space. Treemaps, though, have limits
as well; as with previous presentation methods, zooming,
panning, and animation can extend these limits.

Table 1 indicates display limits for binary trees using the
above approaches with non-overlapping nodes. The formulas
for tree diagrams assume no horizontal separation space for
nodes on the leaf level and also assume enough vertical space
to display all tree levels. The formulas for treemaps assume
no offsets and leaf nodes of equal weight, which will generate
square bounding boxes for this example. It should be noted
that without offsets only the leaf nodes appear in the displays,
although internal structure can be inferred. Figure 4 illustrates
the same tree as Figure 3 with offsets removed. The maximum
size of representable hierarchies decreases as offset size
increases.

2.4 Degenerate Cases

Treemaps can display the largest hierarchies when the
aspect ratios of the bounding boxes are approximately one.
When this condition does not hold, information may begin to
"drop" out of the display.

Rather than having nodes with small weights or extreme
aspect ratios disappear from the display, it is possible to set
minimum node dimensions. With this approach nodes whose
display size would normally fall below the resolution of the
display medium are assigned some small constant width or
height by borrowing display space from sibling nodes.

Although this approach prevents nodes from dropping out
in many cases, it has its own limitations. When the number of
such nodes to be partitioned along a given axis exceeds the
resolution of the display along that axis, information will still
disappear. Regions where this occurs can be indicated by a
special color and zooming facilities provided. Black areas in
Figure 8 indicate clusters of nodes that are not displayed.

Since the display size (bounding box) of a node is deter-
mined by its weight, nodes typically drop out of the display
in order of their weights. This "graceful degradation" pre-
serves relatively important nodes while indicating where
collections of relatively less important nodes are located.

3 Conveying Information

User disorientation is a problem when presenting large
bodies of information, especially when users are confronted
with unconventional display methods like treemaps. The
treeViz (a treemap based Macintosh directory visualization
tool) have highlighted a number of usability issues and
subsequent refinements. A few of the more important issues
and refinements are:

- nesting offsets
- user control of attribute mapping, node filtering, and
  sibling node sort order
- animation
- small multiple displays
- zooming
- textual "signposts"

3.1 Nesting Offsets

Treemaps convey structure via containment (nesting and
grouping) in the same fashion as Venn diagrams [9, 14].
Nesting offsets give users control over the allocation of
display space between internal and leaf nodes. Larger offsets
put greater emphasis on internal nodes and hence the structure
of the hierarchy; smaller offsets emphasize leaf nodes. With-
out offsets only leaf nodes are directly visible; the internal
structure of the hierarchy must be inferred from the grouping
of leaf nodes.

Users viewing new hierarchies often need offsets in order
to become familiar with the global structure of the hierarchy.
After a short period of use users generally prefer smaller
offsets (0, 2, or 4 pixels), as noted in the two experiments and
through the use of TreeViz. Small offsets provide a degree of
global context while still maximizing the display space avail-
able for the display of leaf nodes.
3.2 Attribute Mapping

User confidence in the treemap application may be improved by providing the user more control over attributes which determine node size (weight) and color [8, 15]. Figure 2, for example, displays the same treemap as in Figure 3, except that all leaf nodes have the same weight (the areas of the leaf nodes' bounding boxes are all the same).

Modifying these attributes through functions allows users to emphasize features in the data in varying importance. Inversion of attributes is a simple function that flips the order of importance: instead of salespeople with the largest profits having the largest bounding boxes, salespeople with the smallest profits would have the largest bounding boxes. Logarithmic or power functions have proven useful for providing alternative views of the hierarchy.

Domain attributes may also be mapped to color. Quantifiable attributes (placed on a numeric scale) worked well with different luminosity levels of the same hue (maintaining constant saturation) in the GEnie experiment to represent profitability levels.

If an non-quantifiable attribute is to be displayed, the approach of assigning different hues to each attribute is effective. The Macintosh implementation of treemaps utilizes evenly separated hues while maintaining constant luminosity to convey different file types.

User control over the color is of primary concern as color preference varies by task and individual. Aesthetically pleasing color schemes can be preconfigured for the user and accompanied by a color key or chart for user reference [3]. One area of color control addresses the problem of color deficiencies and monochrome monitors: providing a transformation to a gray scale or patterns alleviates the problem. When distinct hues are eliminated, information can be conveyed via gray scale [6].

3.3 Node Filtering

Filtering nodes allows users to concentrate on features of interest. In hierarchies users may wish to see only those nodes satisfying certain properties. Examples include: internal nodes, leaf nodes, specified branches of the hierarchy, nodes of certain depths or nodes with a particular attribute (for example, all text files).

Two functions are useful for initial orientation: expanding a level at a time allows the user to gradually step down the hierarchy and view each level before proceeding further; expanding particular nodes allows users to view the detail of node(s) in the context of the complete hierarchy.

3.4 Sibling Node Sort Order

The order in which sibling nodes are displayed within a parent can be used to further orient the user or provide additional information about these nodes (node type, rank, alphabetic order, etc.).

A concern specific to grouping sibling nodes in treemaps is that of display size. A very thin node between much larger nodes tends to become lost. It is often useful to group sibling internal and leaf nodes separately, as leaf nodes are generally far smaller than internal nodes. It is also advantageous to group leaf nodes together as a nested block (when offsets have been specified) instead of nesting individual leaf nodes. This nested block saves display space and also provides further distinction between leaf and internal nodes.

3.5 Animation: Relative vs. Absolute

Treemap animation was used in the GEnie experiment with positive results: growing and shrinking bounding boxes reflected changes in the underlying data and conveyed these changes in a powerful way. Six months worth of GEnie product data were available, displayed one month at a time. Users were able to step the diagram through the months with a slider positioned beneath the treemap.

Animating a full treemap (one that fills up the entire display area) over a time sequence presents relative changes to the user. For example, suppose a hierarchy consisted of only two bounding boxes, each taking up 1/2 of the screen. If during the next animation sequence, the first box grows to occupy 3/4 of the screen, the diagram should be interpreted to mean "Relatively, the first node's weight is three times the value of the second node's weight."

From an absolute perspective, though, the weight of the two nodes may have changed in a number of ways: increase of the first node's weight with a smaller increase, no change, or a decrease in the second's; no change in the first node's weight with a decrease in the second's; or a decrease in the first node's weight and a larger decrease in the second's.

Treemaps can, however, be utilized for animation of absolute data if local maximums are known for the attribute which determines the weight. For example, in an application which displays daily stock market information with stock value as the size attribute, the day with the total highest stock values would be reflected in a treemap which occupies the full display area. The other days, with lower total values, would use treemaps which occupy a percentage of the total display area based upon their value divided by the maximum value. Using this method, the entire treemap would grow and shrink, and the area of each bounding box would reflect absolute data over the time period being examined.

3.6 Small Multiples

Treemaps promote relative comparisons and are particularly suited to the presentation of small multiple views or animation when relative comparisons are desired [15].

The GEnie experiment used this technique to display six months' worth of data side by side. Users were asked to interpret changes by following particular squares across the diagram and noting changes in size, position, and color. Figure 5 is a small multiples view of the six months in the GEnie experiment using a subset of the product data. The
small multiple technique also applies to the concept of multivariate data display, discussed in Section 4.5.

As the products are always displayed vertically in the same order, a single product may be followed across the months to gauge performance. In this case the first product’s revenue (size attribute) is expanding relative to the other products in the product category; the darkening gray indicates the profit of the product is changing. The widths of the sections indicate overall growth in this 6 month period.

3.7 Zooming

Zooming allows the promotion of any node to full display size (the zoomed node becoming the new root of the displayed hierarchy), providing space for the presentation of small cluttered regions. But navigational tools are a double-edged sword, for while they allow users to hone in on regions of interest, they also cut off users from previous contextual features.

Zooming was successfully implemented in the GEnie experiment. Users had the option of zooming in on any node in the display, thus allowing quick navigation through the hierarchy. Options to zoom back one level and zoom back to the original root were also provided.

Care should be taken to avoid disorienting the user. The zoom effect, therefore, should incorporate some traditional visual cues (such as zoom lines or increasing or diminishing rectangle outlines) to identify what exactly is being zoomed in to or away from.

3.8 Textual Signposts

Text labels can be used to further orient the user to the hierarchy if space is available. Nodes that are large enough to provide textual signposts are useful as landmarks in a sea of boxes. Figure 5 gives an indication of how text labels may be utilized in a treemap application. Each node in the diagram has a text label in the upper left corner.

4 Current Research

Current research directions have been greatly influenced by the usability studies and experiments that have been conducted thus far. We have found users to be generally receptive to the idea of treemaps but wishing to use them to display more familiar hierarchies. Tools for visualization are no more interesting than the data they present. Can treemap users mine the wealth in rich hierarchical data sets? We think there is great potential.

4.1 Queries

A capability that would allow users to specify queries and have the results shown by highlighting or blinking matching bounding boxes is the most prevalent request. Issues here relate to appropriate highlighting mechanisms and feedback to users. An offshoot of this involves implementation of dynamic queries [16, 2], which allow users to generate a large number of queries in a short period of time via direct manipulation with sliders or other widgets.

4.2 Aspect Ratio Perception Problem

Treemaps use a single numeric weight to determine the display area of a node in the hierarchy. Perceptual difficulties arise when area comparisons are made between nodes of differing heights and widths, as users cannot accurately gauge fine area differences between rectangles differing in both dimensions. Figure 2 illustrates this problem — all of the leaf nodes have the same weight (area), but their heights and widths differ.

Two-D representations are poor for comparing linear values that are similar, but they can show greater ranges, a benefit in the case of file sizes which can range over six orders of magnitude. Display area can be used to rapidly locate nodes of interest, which can then be compared in detail via mouse tracking and dialog boxes.

4.3 Dynamic Algorithms

In a dynamic environment it is useful to isolate global recomputation from local perturbations such as node insertion, deletion, or size changes. Treemaps allocate space in a relative manner, and as such are inherently susceptible to global recomputation. Current algorithmic research is concentrating on minimizing recomputation in dynamic environments.

4.4 2 1/2-D Treemaps

A “third” dimension can be added by using the 2-D rectangular area as the base of a 3-D solid. Increasing the visual vocabulary can provide for richer information resolution [4]. Simply extruding the rectangles produces a Manhattan-like scene, where rectangular solids obscure one another. Using a single point in the third dimension creates pyramids, which do not obscure each other as much. The height dimension can code one more variable, and the location of the apex,
 apex skew, and the four sides can code additional properties in a manner similar to datajacks [3, 5]. Leaf nodes become pyramids and internal nodes become flat top plateaus. Figure 6 illustrates 2 1/2-D treemaps. True 3-D treemaps would be volumes partitioned on all 3 dimensions.

Free movement of the perspective point can provide natural zooming and perspective. More interesting nodes, defined by nearness to the perspective point, receive more display space, a natural fisheye view. As is always the case with treemaps, nodes with greater weights (more interesting data points) also receive greater display space, as their base dimensions are greater.

4.5 Multivariate Comparison

Treemaps have potential as a multivariate exploratory data analysis tool. Hierarchies can be created based on the degree of interest in a set of categorical variables [10]. The display space is partitioned amongst the categorical levels of each variable relative to their proportionate values. Treemaps can be generated either singly or as a series of small multiples. Figure 7 shows a small multiples presentation of subject performance in the treemap vs. UNIX directory browsing experiment.

The 12 largest vertical bounding boxes represent the total time required by each subject to answer all questions. The horizontal partition for each subject represents the treemap questions on the top and the UNIX questions on the bottom; within each interface condition are the 7 individual questions. The 12 short horizontal bars show the relative performance of treemaps vs. UNIX for each subject. It is readily apparent that, in general, subjects performed faster using the treemap interface. Interactive browsing also shows that the global questions (largest boxes) took substantially more time to answer using UNIX. The same information is also presented in Table 2.

5 Experimentation

The utility of treemaps was tested during the past year in two experiments. Both experiments were designed to reflect real-world situations and needs.

5.1 Directory Browsing Experiment

A within-subject counterbalanced experiment with twelve subjects was used to compare treemaps with UNIX (tcsh shell) for directory browsing tasks. A directory hierarchy with approximately 500 files and 50 directories was used. Subjects answered 7 questions with each interface. Figure 7 presents a relative view of users performance and Table 2 presents a tabular view user performance measured in seconds per question. Results were analyzed using a 1-way repeated measures ANOVA.

The first 5 questions were local in scope, dealing with particular files or directories. All local questions were correctly answered within the allotted time (5 minutes per question). On the local questions, statistically significant performance time differences ($p = 0.05$) were found for the first two questions, which favored UNIX. Since subjects all had at least one year of UNIX experience and no previous experience with treemaps other than the 15 minute training period, it is possible that this effect may have been due to learning effects. Subjects performed comparably using either interface on the remaining questions that were of local scope.

Since treemaps present the entire hierarchy at once it was hypothesized that treemaps would be faster for questions that are global in scope. Global questions dealt with portions of the hierarchy larger that single directories. Statistically significant performance time differences ($p = 0.05$) were found for both global questions, favoring treemaps. Error rate analysis was also used as many of the subjects were unable to answer these questions correctly within the allotted time. Five subjects were unable to complete either one or both of the global questions correctly. A total of six errors were made as one subject could answer neither of the questions. All of the errors were made by subjects using UNIX. UNIX users made statistically significantly more errors ($p = 0.05$). All users successfully completed the global questions using the treemap interface, demonstrating the effectiveness of treemaps for global comparisons.

A few subjects remained after the experiment to use the treemap technique to visualize their own personal UNIX

(See color plates, p. CP-16.)
directories. Visualizing their own information from a new perspective proved to be both interesting and exciting.

Treemaps can significantly aid such tasks as locating large old files or clusters of files with similar attributes. Treemaps proved to be an effective visualization tool for file hierarchies.

5.2 GEnie Experiment

An experiment was conducted using a top-down treemap (Figure 5) to display financial data for the product hierarchy of the General Electric Information Services’ consumer information service, the General Electric Network for Information Exchange (GEnie). Six months of GEnie data was used — for each product, revenue and profit figures were noted.

The experiment utilized a between-group design with 18 GEnie employees and tested two versions of the treemap against hard-copy financial reports currently in use at GEnie. The treemap versions included an “animated” version, in which the data was stepped through one month at a time via a slider, and a “small multiples” version, in which all six months were placed side-by-side. Size, color and animation were investigated via twelve questions on revenue, profits and data trends. Users were timed for each question with a maximum limit of five minutes per question. The results were analyzed using a 1-way ANOVA.

The performance of both treemap groups was statistically significantly faster ($p = 0.05$) than that of the hard-copy group for tasks requiring users to identify global trends in revenue and profitability across the six months. Size allowed the users to concentrate on products of interest (products that generated the most revenue) [7] and quickly answer questions related to high revenue; a hindrance to this was the aspect ratio problem mentioned earlier where products of similar revenue had to be compared manually.

Color was a potent tool for analyzing profit changes, and the low-saturated blues and reds (representing positive and negative profit levels, respectively) showed up on the screen in sharp contrast. Users found these colors satisfactory, but several seemed to be overwhelmed by the blend at times. Animation and zooming were used to a great extent, and users subjectively gave these features high marks; the problems mentioned above with relative animation surfaced here, however, and contributed to some high response times for treemaps.

Subjectively, there was a strong preference for the animated treemaps in terms of screen layout, information conveyed and capabilities. We did discover a tendency to not “trust” the diagram in the initial questions, given users’ lack of experience with treemaps (even with training) and the aspect ratio problem mentioned above. This caused the treemap users’ times to be skewed upward in those questions. The tendency disappeared in the latter half of the experiment.

6 Conclusion

Treemaps represent a unique approach to effectively communicating information about large hierarchies, which contain information that users may previously have gleaned over a long period of time or overlooked entirely. Providing users with the capability to display the entire hierarchy allows information to be drawn from it that may not have been obvious via traditional means. Applications include file hierarchies, organizational charts, medical clinical trials, sales figures, stock portfolio analysis, budget allocations — there are many possibilities. As information spaces grow in size and complexity, the need for data visualization tools will only increase.

Acknowledgments

Supported in part by the University of Maryland Systems Research Center under NSF Grant CDR-88-03012.

We would like to acknowledge the support of the members of the Human-Computer Interaction Lab, whose suggestions and criticisms have been greatly appreciated.

The TreeViz application, developed at the HCIL for the Macintosh, is distributed by the University of Maryland’s Office of Technology Liaison, College Park, MD, 20742.

<table>
<thead>
<tr>
<th>Subject</th>
<th>Group</th>
<th>Ave</th>
<th>StDev</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>TA</td>
<td>43</td>
<td>31</td>
<td>46</td>
<td>25</td>
<td>21</td>
<td>58</td>
<td>57</td>
<td>269</td>
<td>61</td>
<td>18</td>
</tr>
<tr>
<td>2</td>
<td>UA</td>
<td>44</td>
<td>13</td>
<td>41</td>
<td>18</td>
<td>23</td>
<td>21</td>
<td>29</td>
<td>41</td>
<td>19</td>
<td>18</td>
</tr>
<tr>
<td>3</td>
<td>UB</td>
<td>42</td>
<td>54</td>
<td>92</td>
<td>13</td>
<td>11</td>
<td>11</td>
<td>61</td>
<td>148</td>
<td>420</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>TB</td>
<td>30</td>
<td>16</td>
<td>52</td>
<td>64</td>
<td>10</td>
<td>26</td>
<td>40</td>
<td>235</td>
<td>35</td>
<td>24</td>
</tr>
<tr>
<td>5</td>
<td>TB</td>
<td>46</td>
<td>16</td>
<td>50</td>
<td>23</td>
<td>25</td>
<td>22</td>
<td>206</td>
<td>26</td>
<td>20</td>
<td>57</td>
</tr>
</tbody>
</table>

Table 2: Treemap vs. UNIX Experiment Results (in seconds per question)
References


Figure 8: Slice-and-Dice Treemap
Macintosh file hierarchy with 1500 files and 190 directories. Notice the duplicate directory at the top level which contains a copy of the system file.

(See color plates, p. CP-16.)