Edge Equalized Treemaps

Aimi Kobayashi
Department of Computer Science
University of Tsukuba
Ibaraki, Japan
kobayashi@iplab.cs.tsukuba.ac.jp

Kazuo Misue
Faculty of Engineering,
Information and Systems
University of Tsukuba
Ibaraki, Japan
misue@cs.tsukuba.ac.jp

Jiro Tanaka
Faculty of Engineering,
Information and Systems
University of Tsukuba
Ibaraki, Japan
jiro@cs.tsukuba.ac.jp

***Abstract*—Treemap is a visualization method for hierarchical structures in which nodes are drawn as rectangles and arranged in a nested style. Several variations of Treemap have been developed to represent different types of data. In this paper, we propose an Edge Equalized Treemap, a representation that embeds visual data such as a bar chart in leaf rectangles. This representation is characterized by leaf rectangles of equal widths. Because their widths are equal, the scale intervals of charts in a leaf rectangle can be unified, meaning that we can compare charts simply by looking at them. We compare the Edge Equalized Treemap with existing layout methods, and demonstrate the usefulness of our approach.**

***Keywords*-Visualization, Hierarchies, Treemap, Equalized Edges, Charts with Orthogonal Coordinate Axes**

I. Introduction

Treemap [1] is a visualization method in which nodes are drawn as rectangles and arranged in a nested style. Originally, Treemaps represented trees where each leaf node had only one weight. However, there are trees in which leaf nodes have multiple data and time-series data. In order to draw such trees, many representations derived from Treemap have been proposed.

We focus on a representation that embeds charts representing time-series data, such as bar charts or area charts, in leaf rectangles [2]. When using existing methods, it is difficult to unify the scale intervals of charts in the entire Treemap. As a result, it is confusing and thus difficult to compare different charts.

Therefore, we take into consideration the ease of comparison between charts, and develop a representation in which the widths of leaf rectangles are the same. In this representation, because the charts have the same width, the scale intervals of the charts' horizontal axes can be unified. Furthermore, to adapt the heights to the weights, it is easy to unify the scale intervals of the vertical axes.

II. Related Work

Many visualization methods for hierarchical structures exist; among these, Treemap is one of the most widely used methods. Treemap represents hierarchies as divisions of space. Each node is represented as a rectangle, and child nodes are placed in their parent node. Treemaps can represent large-scale hierarchies in a given area of visualization.

Various representations have been derived from Treemap. Squarified [3] and Circular [4] Treemaps are characterized by the shape of their nodes. A Squarified Treemap is a representation where the nodes' shape is as close as possible to a square, and a Circular Treemap is a representation where nodes are drawn as circles instead of rectangles.

Ordered [5] and Spatially Ordered [6] Treemaps are characterized by the positions of their nodes. An Ordered Treemap targets ordered hierarchies, and its representation maintains this order. A Spatially Ordered Treemap is a representation targeting hierarchies with geographical location information, maintaining the location relationships among nodes. Other representations exist, such as a Cluster Treemap [7], which is a representation where the distance between data is reflected by relative positions in Treemap.

In the above-mentioned representations, the widths, heights, or radii of leaf nodes are not unified. Therefore, when we embed charts in leaf rectangles, it is difficult to unify the scale intervals of charts across the entire Treemap.

The original Treemap divides the drawing area recursively. Therefore, there are no blanks, but the widths and heights of rectangles vary. For some data, it would be helpful to unify the shapes of rectangles, and some such representations have been proposed. Quantum Treemaps [8] represent nodes as rectangles that are proportional to the object size; the aspect ratio is suitable for the representation of important photographs. HeiankyoView [9] represents hierarchical structures through containment relationships.

Schreck et al. [10] developed a method of representing branch nodes as a grid, based on Quantum Treemaps. Because the widths and heights of leaf rectangles are all the same in this representation, it is easy to unify the scale intervals of the horizontal axes when we embed charts.

The three above-mentioned representations have rectangles whose shapes are the same, and the area of these rectangles cannot be adjusted according to the data drawn on the chart. When a chart handles data with a very large value, the scale interval of the numerical value axis becomes smaller. If we unify the scale intervals in existing variations

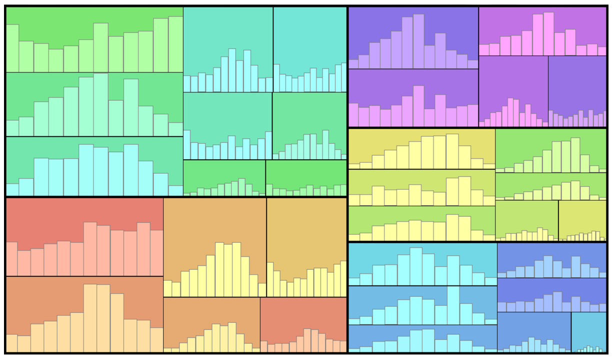

Figure 1. Squarified Treemap for rainfall data

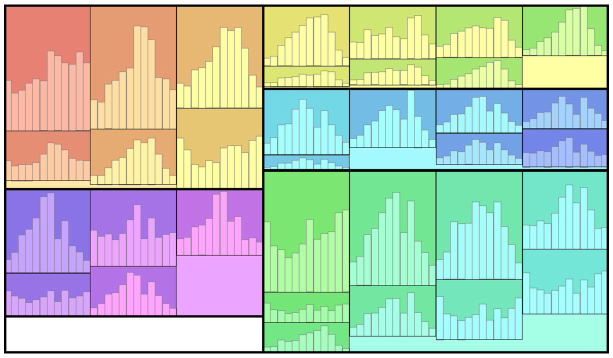

Figure 2. Edge Equalized Treemap for rainfall data

of Treemap, a chart with a smaller amount of data cannot use its assigned area efficiently.

III. Ease of comparison between charts

We consider embedding charts with orthogonal coordinate axes into a Treemap. Here, we suppose that charts are drawn on a two-dimensional plane, and have two axes perpendicular to each other; two examples are bar charts and area charts. If we wish to represent time-series data, many charts will use the horizontal axis for time. When two or more charts have the same time axis, the scale intervals of their axes should be unified for ease of comparison. Therefore, we aim to unify the scale intervals of the horizontal axes when we embed the charts into Treemap.

A. Embedding charts into Treemap

Figure 1 shows a Squarified Treemap with embedded bar charts. This Treemap represents the annual rainfall of Honshu[1] (the main island), Japan. Each rectangular area represents the rainfall in one prefecture, and the thick borders represent individual regions. We can understand the annual rainfall of a prefecture by observing the area of its corresponding rectangle. In addition, by studying the bar chart embedded in each rectangle, we can appreciate the monthly rainfall in a prefecture. The horizontal axis of the bar chart expresses time, and the vertical axis describes the amount of rainfall. Figure 2 represents the same data set using our proposed Edge Equalized Treemap. We will discuss the detailed construction of this Treemap in later sections.

1) Problems with width variation: As we can see in Figure 1, the leaf rectangles have various widths. Moreover, different charts have different bar widths. Therefore, if the scale interval of the vertical axis is unified in Treemap, we cannot elicit quantitative values from the areas of the bars. On the other hand, if we want to describe the same value by the same area, it is not easy to unify the scale intervals of the vertical axes. Making a value in the chart correspond to both height and area is not possible, and may cause misunderstandings.

2) Margin at the top of the rectangle: To compare multiple charts, the scale intervals of their axes should be unified.

The leaf rectangles in Figure 1 have various heights. In particular, the maximum value of a vertical axis is extremely large, necessitating a bar chart drawn in a very tall rectangle. Therefore, unused spaces remain at the top of the leaf rectangles, and thus the drawing area is not used efficiently.

B. Requirements for embedding charts

We consider two requirements for embedding charts into Treemap. The first is the unification of the scale intervals of charts, and the second is a change in the size of the drawing area for each chart.

1) Unification of axes and their scale intervals: In general, when we compare charts that describe the same kind of data, we should assign the same variable to the same axis. For example, when we compare monthly precipitation charts of region A and region B, we should assign the months to the horizontal axis and the amount of rainfall to the vertical axis in both charts. In addition, the scale intervals of the corresponding axes should be the same. If the scale interval is different for each chart, it is not easy to compare the values correctly.

2) Area efficiency for embedding charts: We consider that cutting the upper part of a chart whose maximum value is small does not affect the ease of comparison. Therefore, when we embed charts into Treemaps where the filling rate is important, we should change the drawing area according to the maximum value in the chart. For example, it is appropriate to embed a chart whose maximum value is large into an area whose height is large, and vice versa for a chart whose maximum value is small.

[1]There are 47 prefectures in Japan. These prefectures compose eight regions, five of which are in Honshu.

IV. Edge Equalized Treemap

A. Unification of widths

We have decided to unify the widths of leaf rectangles and use their height to correspond to the weight of the leaf node (Figure 3). There are two reasons for our choice. First, we can unify the scale intervals of horizontal axes across charts with embedded leaf rectangles. For example, if the leaf nodes have the same number of data, we can unify the bar widths in two or more charts. Second, we can prevent extremely large margins if we unify the scale intervals of the vertical axes. For example, we can set a suitable bar height by adapting it to the weight of a leaf node.

While we aim to unify the widths of the leaf rectangles, we can also unify their heights by swapping the width and height.

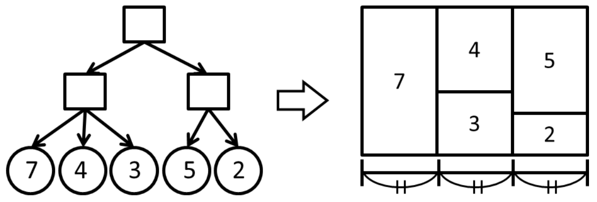

Figure 3. The concept of an Edge Equalized Treemap

B. Allowing blanks

We decided not to limit the existence of blank space in Treemaps. We take into account the aspect ratio as considered in the Squarified Treemap, along with the filling rate.

If we want to unify the widths of leaf rectangles by partitioning the space, we can achieve this using horizontal lines as partitions. By adopting this policy, we maintain the filling rate at 1 and unify the widths of the leaf rectangles. In this case, however, the aspect ratio of each leaf rectangle tends to be low. When we consider that charts are embedded into leaf rectangles, rectangles with extremely low aspect ratios are not desirable.

Therefore, we allow blank space, and we introduce the criterion of the filling rate. We believe it is necessary to achieve a balance between the filling rate and the aspect ratio.

C. Setting of constraints

Treemaps have the following constraints:

- A node is described as a rectangle.
- A parent–child relationship is placed in a nested rectangle.
- Rectangles are placed so that they do not overlap each other.

We developed a new Treemap that also includes the following constraints:

- The widths of leaf rectangles are unified.
- The height of a leaf rectangle is proportional to the weight of a leaf node.
- The filling rate and aspect ratio are as close to 1 as possible.

Here, the filling rate is the sum of the areas of all leaf rectangles divided by the area of the whole Treemap. The aspect ratio is the length of the short side of the rectangle divided by the length of the long side of the rectangle. These are formulated as follows.

Let T be a rooted tree in which each root is expressed by r and a set of leaf nodes is L. The area of node v is expressed by $a(v)$. The width of a rectangle is $w(v)$, and the height of a rectangle is $h(v)$.

The filling rate f is given by equation (1). The output is a positive number less than or equal to 1.

$$f = \frac{1}{a(r)} \sum_{l \in L} a(l) \tag{1}$$

The aspect ratio $g(l)$ of node l is expressed by equation (2), and the average aspect ratio of all leaf rectangles is $\overline{g}$. $g(l)$ and $\overline{g}$ are both positive numbers less than or equal to 1.

$$g(l) = min(\frac{h(v)}{w(v)}, \frac{w(v)}{h(v)}) \tag{2}$$

$$\overline{g} = \frac{1}{|L|} \sum_{l \in L} g(l) \tag{3}$$

In order to balance the average filling rate and the aspect ratio, we aim to increase s, which is defined in equation (4). Here, w_1 and w_2 represent the weights applied to f and $\overline{g}$, respectively. It is possible to adjust the balance of the filling rate and aspect ratio by changing w_1 and w_2.

$$s = w_1 \cdot f + w_2 \cdot \overline{g} \tag{4}$$

V. Determination of the layout

If we unify the widths of the rectangles in each level of the hierarchy, the top-down division of space can be used for as well as many other variations of the Treemap. However, one of our constraints states that we must unify the widths of all leaf nodes, regardless of the depth. In addition, we only wish to unify the widths of leaf nodes; we do not want to unify the widths of branch nodes. Therefore, we consider the top-down partitioning of space to be unsuited to the achievement of our goal.

A. *Rough flow of the process*

We examine bottom-up methods of positioning the rectangles, as described in the following.

First, we prepare rectangles corresponding to leaf nodes with equal widths. The height of the rectangle is proportional to the weight of the leaf node. When the rectangles of all child nodes have been determined, we place them into as narrow a rectangular area as possible. We repeat this process bottom-up, from the leaf nodes to the root node.

Using this method, we cannot know in advance the width and height of the rectangles. Therefore, after the rectangle corresponding to the root node is decided, we adjust the horizontal and vertical scale to fit the drawing area.

B. *Flow of the process including trial and error and evaluation*

In the process described in Section V-A, there are two degrees of freedom.

(1) How do we determine the height of the rectangle corresponding to the leaf node of a certain width?
(2) How do we fit child nodes into the rectangular area of each branch node?

In fact, (1) does not influence the resulting layout by adjusting the scale to fit the drawing area. We want to find a good layout based on the aesthetic criteria that we set previously (aspect ratio and filling rate). However, based directly on the aesthetic criteria, it is difficult to determine the rectangle into which we should place the child rectangles. There are two reasons behind this statement. If the rectangle of the root node is not determined, we cannot calculate the filling rate. In addition, if the scale of the Treemap is not adjusted to the drawing area, we cannot evaluate the aspect ratio. Because of this, we decide to parameterize the filling of the rectangle and use a trial and error method. We will arrange the rectangles and evaluate the results of each layout while changing the parameters. Using this process, we will be able to obtain a better layout.

C. *Concrete flow of the process*

We show the flow of the process in Figure 4. First, we take as input a weighted hierarchical structure for each leaf node. The width and height of the leaf rectangles can then be determined according to their weights. Second, we repeat the process of filling the rectangles, from the leaf nodes to the root node. After the layout of all the rectangles is decided, we adjust the size of the Treemap to the drawing area and evaluate the result of the layout with an evaluation function. We then change a parameter and go back to the process of filling the rectangles. By repeating the above process, we are able to find the layout that scores best in the evaluation.

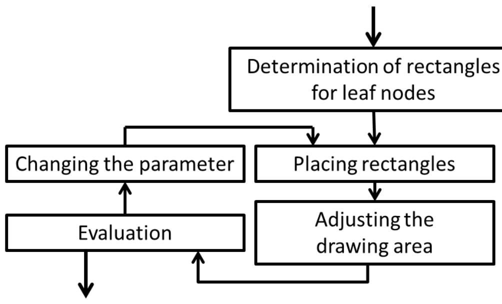
Determination of rectangles for leaf nodes
Changing the parameter
Placing rectangles
Evaluation
Adjusting the drawing area

Figure 4. Flow of the process

1) Determination of rectangles for leaf nodes: We decide the width and height of a leaf rectangle according to the weight of a leaf node. The width of a leaf rectangle is 1 and the height is proportional to the weight. When these rectangles are filled, their heights are variable. Therefore, the proportional constant for determining the height has no meaning; it is important that the proportional constants are the same across all leaf nodes.

These widths and heights are provisional until the layout of all nodes is decided. They are changed when the Treemap is adjusted to the size of the drawing area.

2) Placing rectangles: We treat the problem of fitting child nodes into the rectangular area of branch nodes as a Strip Packing Problem. The Strip Packing Problem describes the fitting of rectangles into a fixed-width rectangular space in order to minimize the vertical length. We swap the horizontal and vertical in this problem, and decide to place the child node rectangles from left to right in the fixed-height rectangle.

In our problem, the height of the rectangle into which the child nodes will be placed is unknown. Therefore, we fit the rectangles of the child nodes while slowly adjusting the height parameter.

We use two algorithms as solutions to the Strip Packing Problem.

One of them is the Next-Fit Algorithm [11], which is a solution to the Bin Packing Problem (Figure 5). If the order of placement of the rectangles has been decided, the Next-Fit Algorithm offers a high-speed layout solution. In this algorithm, rectangles are filled from the upper left to the lower right of the Treemap. The algorithm places rectangles from top to bottom; if it cannot place a rectangle to the bottom, it draws a line on the right and continues placing rectangles on the line from the top. By repeatedly changing the placing order, we can adopt the best result.

We also use the Stack Algorithm (Figure 6). This assumes that the rectangle widths are equal. This algorithm sorts rectangles in descending order; the rectangles are then placed in order, from the smallest rectangle at the top to the largest rectangle. Because this algorithm does not depend on the placing order of the rectangles, it can be faster than the Next-Fit Algorithm.

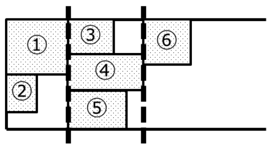

Figure 5. Placement using the Next-Fit Algorithm

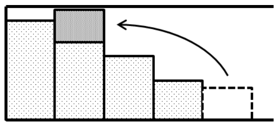

Figure 6. Placement using the Stack Algorithm

3) Determination of rectangles for branch nodes: The dimensions of each branch node are determined by the result of the placement of child nodes. That is, the minimized rectangle that surrounds the child node rectangles gives the branch node rectangle.

This algorithm uses a bottom-up approach to place rectangles. Therefore, when the rectangles of brother nodes are constructed and placed, they determine the rectangle of their parent node. We place rectangles by repeating this process.

4) Adjusting the drawing area: After all nodes are placed, the size of the Treemap is adjusted. We treat our problem as a nested Strip Packing Problem. Because our approach is based on working from the inner nested part, it is difficult to determine the size of the outermost rectangle. For this reason, we adjust the size of the Treemap to the drawing area after the layout has been decided.

5) Evaluation of the layout: Using the evaluation function s defined in equation (4), we evaluate the resulting layout, with a higher value of s denoting a better layout. We compare the result to the previous highest value, and save the better layout as the best result.

6) Changing the height parameter: In order to get a better result, we reconstruct the layout after changing the height parameter. When we place rectangles, we need to set the maximum height value that can be stacked by each branch node. This parameter influences the result of the layout. Therefore, after evaluating each result, we change the height parameter and position the rectangles again. The minimum value of this parameter is chosen to be the height of the tallest rectangle of the child nodes in each branch node, and the maximum value is chosen to be the sum of the heights of the child node rectangles in each branch nodes. The variation in the height parameter for each iteration is the height of the shortest child node rectangle in each branch node.

VI. Use of the Edge Equalized Treemap

We now demonstrate the use of our method by drawing a hierarchical structure. Using an Edge Equalized Treemap, we embed bar charts into each leaf rectangle, and unify the scale intervals of the value axis (vertical axis) in Treemap.

Our Edge Equalized Treemap shows rainfall data for different prefectures. This data is organized in a hierarchy, with Honshu, Japan as the root node; the child nodes show regions, and their child nodes show prefectures (these are leaf nodes). Leaf nodes show the values of monthly rainfall, and use annual rainfall as their weights.

Figure 2 shows the execution result. The area of each rectangle represents the annual rainfall of each prefecture. A bar chart represents monthly rainfall, with the horizontal axis denoting the month and the vertical axis showing the amount of rainfall.

VII. Discussion

We compared an existing representation to our Edge Equalized Treemap and performed an evaluation. We represented the annual rainfall data in Honshu, Japan, using a Squarified Treemap, as can be seen in Section III-A. We represented the same data using an Edge Equalized Treemap in Section VI. We compared and evaluated the two representations.

A. Various widths

When using a Squarified Treemap, the leaf rectangles have various widths, such that the widths of the bar charts cannot be unified. Because of this, the values in the bar chart cannot correspond to both the height and the area of the bar. In other words, we cannot accurately compare the charts just by looking at them.

In the Edge Equalized Treemap, the widths of all leaf rectangles are unified and the widths of all bar charts are also unified. Therefore, the values in a bar chart can correspond to both the height and the area of the bar, and thus we can compare charts by simply looking at them.

B. Margins at the top of rectangles

In Squarified Treemaps, the height of leaf rectangles does not correspond to the weight of leaf nodes. Therefore, there are some large margins at the top of the charts. In other words, the drawing area is not efficiently used.

An Edge Equalized Treemap represents the weight of a leaf node as the height of a leaf rectangle. The weight is the sum of the data drawn in the chart. Thus, the drawing area of the chart has a height corresponding to the value of the data. Because of this, there are no large margins at the top of the leaf rectangles.

C. Considerations

We compared our Edge Equalized Treemap to a Squarified Treemap, and found Edge Equalized Treemaps to be better in terms of the following aspects.

- The interval scale of the horizontal axes can be unified.
- The height of the leaf rectangle is appropriate.

These aspects relate to the ease of comparison of embedded charts in a Treemap. Because of this, the Edge Equalized Treemap is an effective representation.

We executed our programs on an Intel Core i7 CPU 920 @ 2.67 GHz with 6 GB of RAM and Windows 7 OS. It took 7861 s to represent the rainfall data in an Edge Equalized Treemap. On the other hand, the Squarified Treemap required only 0.008 s. Comparing these values, it is clear that the Edge Equalized Treemap took a long time to process. In future work, we will examine ways to speed up the construction of Edge Equalized Treemaps.

VIII. Conclusion

We focused on the representation of embedded charts in a Treemap, and developed the Edge Equalized Treemap. This representation can unify the horizontal axis widths of each chart. Therefore, we can easily compare charts just by looking at them. The height of each leaf rectangle in an Edge Equalized Treemap corresponds to its data, and no large margins are produced in the charts. Using our representation, we expect to be able to support the representation of various data in a Treemap.

Acknowledgment

The authors thank Dr. Simona Vasilache for her helpful comments. This work was partially supported by JSPS KAKENHI 22500081.

References

[1] B. Johnson and B. Shneiderman, Tree-maps: A Space-filling Approach to the Visualization of Hierarchical Information Structures, Proceedings of the IEEE Visualization, pp. 284–291, 1991.

[2] U. Dayal, M. Hao, D. Keim, and T. Schreck, Importance driven visualization layouts for large time-series data, Proceedings of the IEEE Symposium on Information Visualization, pp. 203–210, 2005.

[3] M. Bruls, K. Huizing, and J. J. van Wijk, Squarified Treemaps, Proceedings of Joint Eurographics and IEEE TCVG Symposium on Visualization, pp. 33–42, 2000.

[4] W. Wang, H. Wang, G. Dai and H. Wang, Visualization of large hierarchical data by circle packing, Proceedings of the SIGCHI conference on Human Factors in computing systems (CHI ’06), pp. 517–520, 2006.

[5] B. Shneiderman, and M. Wattenberg, Ordered treemap layouts, Proceedings of the IEEE Symposium on Information Visualization, pp. 73–78, 2001.

[6] J. Wood and J. Dykes, Spatially ordered treemaps, IEEE Transactions on Visualization and Computer Graphics, Vol. 14, No. 6, pp. 1348–1355, 2008.

[7] M. Wattenberg, Visualizing the Stock Market, CHI ’99 extended abstracts on Human factors in computing systems, Extended Abstracts, pp. 188–189, 1999.

[8] B. B. Bederson, B. Shneiderman and M. Wattenberg, Ordered and quantum treemaps: Making effective use of 2D space to display hierarchies, ACM Transactions on Graphics (TOG), Vol. 21, No. 4, pp. 833–854, 2002.

[9] T. Itoh and K. Koyamada, HeiankyoView: Orthogonal Representation of Large-scale Hierarchical Data, Proceedings of the International Symposium on Towards Peta-Bit Ultra Networks, pp. 125–130, 2003.

[10] T. Schreck, D. Keim and S. Mansmann, Regular TreeMap Layouts for Visual Analysis of Hierarchical Data, Proceedings of the Spring Conference on Computer Graphics, pp. 20–22, 2006.

[11] D. S. Johnson, Near-optimal bin packing algorithms, Ph.D. Thesis, MIT, Cambridge, MA, 1973.