Hybrid-Treemap Layouting

Sebastian Hahn and Jürgen Döllner
Faculty of Digital Engineering, University of Potsdam, Germany

Abstract
This paper presents an approach for hybrid treemaps, which applies and combines several different layout principles within a single tree map in contrast to traditional treemap variants based on a single layout concept. To this end, we analyze shortcomings of state-of-the-art treemap algorithms such as Moore, Voronoi and Strip layouts. Based on a number of identified edge cases, we propose a combination of these different layout algorithms, individually selected for and applied on each sub hierarchy of the given treemap data. The selection decision is based on the number of items to be layouted as well as the aspect ratio of the containing visual elements. Furthermore, a layout quality score based on existing treemap layout metrics (e.g., average distance change, relative direction change, average aspect ratio) has been used to evaluate the results of the proposed hybrid layout algorithm and to demonstrate its usefulness applied on representative hierarchical data sets.

Categories and Subject Descriptors (according to ACM CCS): [Human-centered computing]: Visualization—Treemaps; [Human-centered computing]: Visualization—Empirical studies in visualization

1. Introduction
Treemaps [JS91] serves as means to visualize hierarchical data sets in a scalable and space-filling way; they represent a research topic studied yet for more than two decades [Sch11]. Briefly, the visualization principle is based on representing a parent node by a finite area that is recursively subdivided into non-overlapping partitions used to visualize the child nodes. The size of the partitions depends on an application-specific defined weight, typically computed as per-node attribute data. The recursive subdivision follows a partition strategy known as treemap layout. Various layout algorithms have been published with several advantages and disadvantages with respect to their algorithmic complexity, the readability and the arrangement of the visual items in the resulting depictions [Sch11].

Besides other challenges, e.g., treemap rendering [THD13, WTLD15, LFH*16], or reduction of visual clutter [LSHD17], this paper focuses on using treemaps for data sets that change over time, e.g., for the visualization of software metric data [WL08, BD11] or business data [VvWvL06], bringing up another challenge for the layout algorithm—the layout stability. The term layout stability refers to the ability of a layout algorithm to guarantee only small changes in the layout if only small changes in the data set occur [BSW02]. Among many advantages, stable layouts are essential for usability of treemap visualization because users may memorize the layout of treemaps. If small data changes would cause fundamental layout changes, users would have to rebuild their memorized treemap.

Many applications provide a collection of treemap layout algo-

Figure 1: A Hybrid-treemap layout algorithm using a combination of Moore, Slice’n’ Dice, Strip and Voronoi treemaps.

rithms. However, they apply only one (selected) treemap algorithm for visualizing the data. Frequently, this results in several edge cases decreasing the layout quality, e.g., using a squarified treemap layout [BHWW00] for a large number of items within a narrow parent element results in highly narrow items.

In this paper, we propose a hybrid treemap layout that selects and applies several different treemap layouts for each sub hierarchy of
the given data set within a single treemap visualization (Figure 1).
The hybrid layout algorithm takes into account the aspect ratio of
the parent's representation and the number of items to be layouted.
To test and evaluate the approach, we generated a large number of
artificial data sets with only one hierarchy level and produced a
series of snapshots representing changed data sets over time (e.g.,
revisions). The data sets serve as input for eight different treemap
layout algorithms to create a decision-tree for the proposed hybrid
treemap layout. Existing layout quality measurements [BSW02,
HBD17] (average aspect ratio, average distance change, relative
direction change) were used to determine a layout quality score for
the evaluation of the test datasets.

This paper is organized as follows. Section 2 gives a brief
overview about related work in the field of treemap layout algo-
rithms as well as layout quality metrics. The used input data for
generating the decision tree and the layout metric results are dis-
cussed in Section 3. The hybrid treemap layout approach is pre-
sented in Section 4. An evaluation of the presented layout based on
test data sets is described in Section 5. The paper concludes with a
discussion and an outlook on future work in Section 6.

2. Related Work

Treemap Layout Algorithms are published for more than two
decades [Sch11]. Johnson and Shneiderman presented the initial
Slice’n dice treemap (1991) that uses a subdivision in either hori-
zontal or vertical direction, alternating based on the depth of a
hierarchical element [JS91]. This approach, especially if used for
sub-hierarchies with a large number of items, results in shapes with
high aspect-ratios and, therefore, poor readability. Bruls et al. put a
high focus on readability with Squarified treemaps (2000), using a
treemap algorithm that creates square-like shapes and, hence, it al-
 lows for average aspect ratios near one, but as a trade-off shows
poor layout stability [BHVW00]. The trade-off between nicely-
shaped regions and layout stability was first mentioned by Bed-
erston et al., introducing the Strip treemap (2002) and a first eval-
uation that takes into account the change of positions for varying
hierarchical data sets [BSW02]. Tu and Shen tried to overcome
the challenge of layout instability by using a spiral-shaped space-
filling curve, Spiral treemap (2007) [TS07], that also allows for
preserving a specific order of data in the depiction. Tak and Cock-
burn [TC13] also use a space-filling curve to compute the initial
item positions; their Hilbert & Moore treemaps (2013) creates low
mean aspect-ratio and good stability. They also introduced a new
layout metric, the location drift, which overcomes some of the dis-
advantages of the distance change metric. Nevertheless, the eval-
uation of this algorithm against other common ones did not con-
sider hierarchical data sets. In addition to the common rectangu-
lar treemap approaches, Balzer and Deussen present generalized
Voronoi- (or Power-)diagrams to create Voronoi treemaps (2005),
using random initial positions for items [BD]. The algorithm was
extended by Hahn et al. to allow for stable distributions, resulting
in treemaps that create items with low average aspect ratios and a
high visual stability [HTMD14].

Layout Stability in Treemaps is highly connected to the research
in mental maps. Misue et al. define the mental map for graphs with
a model consisting of three different aspects: orthogonal ordering,
proximity relations, and topology [MELS95]. Their definition of
topology focuses on the connections between graph nodes is not
directly applicable to implicit hierarchical visualization techniques
like treemaps. Nevertheless, the orthogonal ordering and proximity
relations propose a direction on how to evaluate the changes in
a layout with respect to a user’s mental map. A common met-
ric for evaluating treemap layout stability is the average distance
change introduced by Bederson et al. [BSW02], which only takes
into account the change in the Euclidean distance of the abso-
ute position and aspect ratios of depicted items. Several evalu-
ations were performed showing that their respective layout algo-
rithm performs best with respect to layout stability. However, ei-
ther they introduced algorithm specific metrics or used artificial or
non-hierarchical data sets [BSW02, TS07, TC13]. Kong et al. eval-
uate as a prerequisite for a good area estimation in treemaps, the
rule of nicely-shaped regions and item orientations [KHA10]. In
a controlled experiment they found, that users can hardly estimate
high aspect ratios especially with different orientations. Hahn et
al. present the relative direction change, metric that focuses on the
topology and arrangement of treemap items and show that a combi-
nation of average aspect ratio, average distance change and relative
direction can serve as significant parameters for a prediction model
for user behavior in treemap item recovering tasks [HBD17].

3. Design Decisions

The design decisions for the hybrid treemap layouting are based
on observations from multiple measurements with single-level
treemaps. These observations as well as the input datasets for the
treemaps are explained in detail in this section.

3.1. Generated Datasets

The generated data, that serves as the input for the treemap lay-
out algorithms consists of multiple varying datasets. For a better
understanding we refer to the following terminology:

• Snapshot: A single-level (one parent, multiple children) hierar-
chical dataset with an additional numeric attribute. This attribute
is percentage-wise expressed through the area of the treemap
items with respect to the sum of all items.

• Time-line: A group of snapshots with a starting snapshot and
multiple successor snapshots extracted by changing the predece-
sor to a certain degree. Each time-line has a starting point given
by a static number of child (childNumber) items created.

• Time-line group: A group of time-lines with the same starting
point (childNumber).

The numeric attribute values and their changes were created ac-
cording to [TC13]. Also, the time-lines include the deletion and
adding of items on a five percent chance to create more realistic
datasets. Due to this, the childNumber of a snapshot just refers to
its starting number of children, while the actual number of children
can increase or decrease within the time-line (no snapshots with
less than 2 items were used). The complete input data for the com-
puted treemaps contained 20 time-line groups with starting child
number from two to 500 items (2-10, 12, 15, 17, 20, 25, 35, 50, 75,
100, 250, 500). Each time-line group contained 10 different time-
lines with each 100 snapshots. This results in 20.000 data snapshots
as an input for 8 different treemap algorithms.
3.2. Layout Metrics

Eight different treemap layouts for (Slice’n’Dice, Strip, StripInverted, Squarified, Spiral, Moore, Hilbert, Voronoi) have been implemented. For each treemap the layout metrics were computed for the aforementioned single-level datasets (see Section 3.1). In addition to the varying number of child elements, 13 different aspect ratios for the parent element were used, from wide to narrow base elements (1/10, 1/2, 2/3, 3/4, 3, 4, 5, 6, 7, 8, 9, 10, 4/3, 2, 1/2).

Using these factors (childNumber and aspectRatio) three layout-quality metrics were measured:

- **Average Distance Change**: Position changes of each individual treemap items.
- **Relative Direction Change**: Changes of a treemap’s topology (arrangement and adjacency of treemap items).
- **Average Aspect Ratio**: Reflecting the readability and usefulness of the treemap.

Finally, the average of all three metrics was used to create a layout quality score. Since all metrics aim for a low number (a 1.0 reflects high changes), a lower quality score indicates less change and small aspect ratios.

3.3. Layout Quality Results

For the final results all trials from a time-line group (same childNumber) were aggregated after a normalization for each algorithm. The 2,080 aggregated measurements (13 aspectRatio × 20 childNumber × 8 layouts) and their resulting layout quality score show several patterns:

- Moore treemaps perform best for aspect-ratios from 1/2 to 2/3.
- Moore treemaps almost always perform equal or better than Hilbert treemaps (243 of 260 measurements).
- Squarified treemaps seem insufficient for all cases, Spiral treemaps perform just slightly better than Squarified.
- Narrow treemap items (up to an aspect ratio of 1/2) should be layouted as Strip (Figure 2a).
- Wide treemap items (starting from an aspect ratio of 1/2) should be layouted as Slice’n’Dice treemaps (Figure 2b & 2c).
- Very narrow treemap items (up to an aspect ratio of 1/3) with small amount of children (up to 12) should be layouted with Voronoi treemaps (Figure 2a).
- Wide treemap items (starting from an aspect ratio of 2/3) with less than 50 children should be layouted with Voronoi treemaps.
- Very wide treemap items (starting from an aspect ratio higher than 2/3) can be layouted with Voronoi treemaps as an alternative to Slice’n’Dice treemaps (Figure 2c).

The complete decision tree extracted from the results is described in detail in Section 4.

4. Hybrid-Treemap Algorithm

Using the results from Section 3.3 a hybrid treemap algorithm was implemented. Mixing traditional rectangular-treemap approaches with polygonal approaches, such as Voronoi treemaps, introduces one major prerequisite. Since a space-filling rectangular subdivision of an arbitrary shaped polygon is not possible, a polygonal approach can only be applied if the hierarchical item that is layouted either does not contain any sub-hierarchies (leafs only), or every sub-hierarchy is also layouted as a Voronoi treemap. In this approach, we decided that sub-hierarchies with leaf nodes only are possible candidates for Voronoi treemap layouts to allow for a higher flexibility in the decision for each individual sub-hierarchy.

As pointed out in Section 3, Moore treemaps generally serve as a favorable layout algorithm within a wide range of aspect ratios. To keep the number of layout switches as low as possible, the decision tree is mainly focused on a few edge-cases (Figure 3).

5. Evaluation

A quantitative evaluation was performed to investigate the performance of the presented hybrid-treemap layout approach with respect to the aforementioned layout quality score. For it, the file...
structures and development histories of eleven software systems (extracted from public github repositories) were used. The projects were randomly picked from a large base of software system data. In addition to the file structure of these systems, the number of lines-of-codes (LoC) of each file serves as the associated numeric attribute, mapped to the area of the treemap items. Futhermore, the used datasets contained multiple snapshots (min: 17, max: 28) of the file structure from different points in time. The characteristic of the snapshots with respect to the number of elements varied from very small, but fastly evolving (DeepLearning4j - min: 16, max: 1035) to large stable ones (Qt - around 20k each). All snapshots were generated at an interval of one month for each project. A pre-check guaranteed that there have been changes between each consecutive snapshot. Also, in contrast to the trials from Section 3, we did not include Slice’n’Dice treemaps in the final comparison, due to its bad readability for complete real-life datasets.

The used datasets shows hybrid treemaps performed slightly better (Mean = 258, sd = 126) than other layout algorithms (e.g., Spiral: Mean = 286, sd = 181) for the tested datasets (Figure 4). Additionally, we found that Moore treemaps (Mean = 294, sd = .137) performed better than Hilbert treemaps (Mean = .352, sd = .109). While Strip and StripInverted treemaps show highly similar performance, Squarified treemap achieves worse layout quality scores than all other rectangular approaches (Mean = .507, sd = .236). In contrast to the scores from the generated dataset trials, Voronoi treemaps perform poorly for the chosen datasets (Mean = .640, sd = .010).

6. Conclusion

We presented a treemap layout that combines the strength of several state-of-the-art treemap approaches. At the same time, the hybrid algorithm makes use of a decision tree which reduces weak-spots of treemap algorithms when used in static manner. An evaluation with real-life datasets shows the usefulness (in terms of a layout quality score) of this approach.

Even though, the hybrid treemap algorithm works sufficient for hierarchical datasets with different characteristics, there is a lot of possibility for improvements. First, decision tree, only based on one data characteristic (the number of elements to be layouted) and a layout attribute (the aspect ratio), is very simple. A more sophisticated decision could be achieved if multiple data characteristics, e.g., changes appearing on the structure or the numeric attributes, of the evolving hierarchical dataset are taken into account. Second, the hard decisions used by this approach are possible weak spots of the algorithm itself. If a sub-hierarchy is changed by an increasing or decreasing number of items and steps over the threshold, a complete change of the sub-hierarchy layout is happening.

Further investigation in the decision process of which treemap layout to use is needed and will be done as future work. Additionally, datasets from alternative domains (other than software systems) will be used for the more detailed information.

The complete supplemental material to this paper is available here

Acknowledgements

The authors would like to thank the anonymous reviewers for their valuable comments. This work was funded by the Federal Ministry of Education and Research (BMBF), Germany, within the "BIMAP" project. We also want to thank seeerne™ for providing us access to their massive data sets.
References


© 2017 The Author(s)