Influence of Container Resolutions on the Layout Stability of Squarified and Slice-And-Dice Treemaps

V. Knauthe[1], K. Ballweg[1], M. Wunderlich, T. von Landesberger[1,2], S. Guthe[1,3]

[1]TU Darmstadt, Germany
[2]Karlsruhe Institute of Technology, Germany
[3]Fraunhofer IGD, Germany

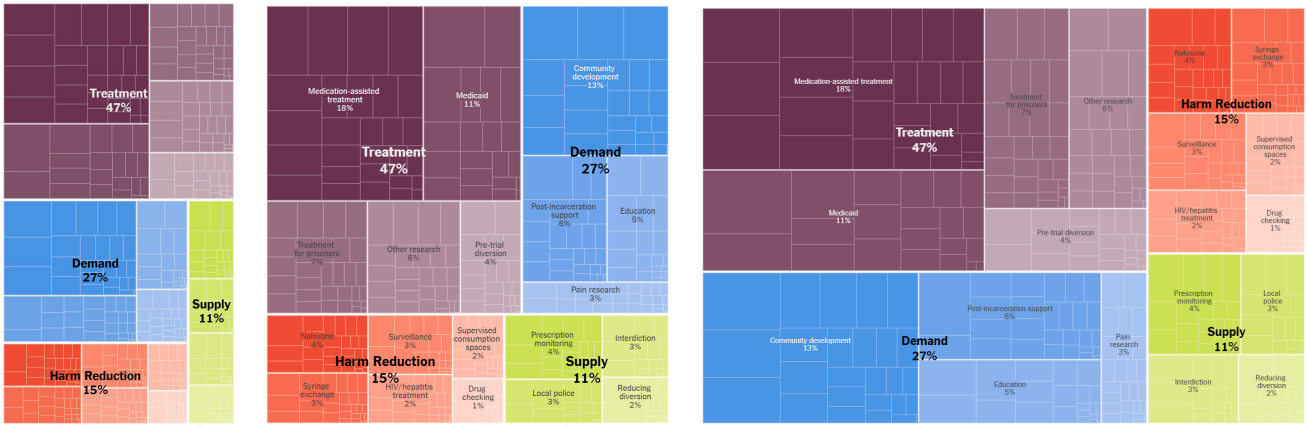


Figure 1: *[Kat18] Three different squarify treemap visualizations for the same tree data with different container resolutions (from left to right), as seen on a smartphone (660x1200), tablet with horizontal orientation (1000x1050) and tablet with vertical orientation (1200x840). Note how the layout on every level of the treemap changes when changing resolutions. The corresponding tree can be seen in Figure 2.*

Abstract

In this paper, we analyze the layout stability for the squarify and slice-and-dice treemap layout algorithms when changing the visualization containers resolution. We also explore how rescaling a finished layout to another resolution compares to a recalculated layout, i.e. fixed layout versus changing layout. For our evaluation, we examine a real world use-case and use a total of 240000 random data treemap visualizations. Rescaling slice-and-dice or squarify layouts affects the aspect ratios. Recalculating slice-and-dice layouts is equivalent to rescaling since the layout is not affected by changing the container resolution. Recalculating squarify layouts, on the other hand, yields stable aspect ratios but results in potentially huge layout changes. Finally, we provide guidelines for using rescaling, recalculation and the choice of algorithm.

CCS Concepts

• ***Human-centered computing*** → ***Treemaps;***

1. Introduction

Rectangular treemaps are a popular way to visualize tree data by recursively utilizing nested rectangles in a space efficient manner. Relative node weights, the implicit hierarchical structure and node order information contained in the tree data should be preserved in the visualisation. Treemaps are employed in a variety of fields and for an extensive range of data sets such as file systems [Shn92], financial data [Kat18], sports [Tur03], google news [MW02], energy efficiency [MTE19], and are implemented in the well known D^3 javascript library [Bos19]. While the Slice-And-Dice layout algorithm (SnD) was the first treemap layout algorithm, many new treemap layout algorithms were created to diminish existing short-

DOI: 10.2312/evs.20201055

comings of old layouts [Shn92]. However the SnD layout algorithm remained important as a baseline for evaluating new layout algorithms. The most widely used algorithm is the Squarify layout algorithm (SQ) [BHVW00].

Different container resolutions (see Figure 1) are becoming more and more relevant due to the multitude of different devices used for information presentation. This is not restricted to slight differences in computer screen resolutions but extends to screen variations as found in smartphones and tablets. Those devices do not only feature comparatively small sizes but allow for direct orientation changes. When working with multiple devices, in a group or personally, information may be shown in inconsistent ways if the visualisation is depending on screen sizes. If screen-sizes lead to fundamental layout changes (e.g. horizontal vs vertical or window resizing), user memorization and therefore mental maps of the data shown may become obsolete [MELS95]. An example for this instability can be observed for data published in the *New York Times* (*NYT*) article *How a Police Chief, a Governor and a Sociologist Would Spend $100 Billion to Solve the Opioid Crisis* in Figure 1 [Kat18]. Especially the blue *Demand* area changes its position and internal child positioning quite a lot. The treemap was created with the default D^3 SQ layout settings [Bos19].

The influence of different visualization container aspect ratios was studied by Hahn and Döllner [HD17] on seven construction algorithm variations for data that is undergoing changes. However the evaluation focussed primarily on data changes in otherwise static environments. Our work focuses on the change SQ and SnD visualizations undergo for static data but changing container resolutions.

Treemap Metrics The following paragraph will summarize Treemap metrics that were developed to evaluate how well different layout algorithms satisfy a given condition. Studied aspects that are relevant for this work include the effect of rectangle aspect ratio sizes on human perception and the effect of changes in visualisations. Kong et al. found, that it is harder for humans to compare the size of rectangles with extreme aspect ratios or diverse orientations [KHA10]. To measure how square-like a set of rectangles is, the *Average Aspect Ratio* metric was introduced. When comparing treemaps that were produced with slightly different input data, or in our case slight changes in construction parameters, layout stability is of importance. The first metric to assess how stable the layouts produced by a given algorithm are, is the *Distance Change* metric. *Distance Change* simply measures how far a rectangle travelled and how much it changed shape when comparing two layouts [SW01]. The *Corner Travel Distance* metric utilizes a shape change calculation originating in computer vision [VSC*19]. Sondag et al. proposed a change measure called *Relative Position Change* that is more focussed on human perception [SSV17]. It uses simple notions like *above, below, right and left* to calculate how the relation of rectangles to each other changes. *Relative Direction Change* uses a similar basic notion but is designed for more general shapes and not specifically made for rectangles [HBD17]. Similar, the *Angular Displacement* metric was introduced for changes in a geographic context. The *Time Varying Data* metric is using a combination of *Angular Displacement* and a variation of *Distance Change* [WD08] [CDY17]. In difference to the mentioned metrics that used two layouts for comparison, *Location Drift* uses a center of gravity to measure changes over a set of treemaps. In addition to the mentioned metrics, the maintenance of a given order in the tree, the space efficiency of layouts and special properties of geographic treemap visualisations were evaluated. However, these are not relevant for our work as the SnD and SQ algorithm are space efficient per default and do not consider geographic information [BSW02] [WD08] [TS07]. Furthermore the SnD layout algorithm maintains perfect order by default, while the SQ layout algorithm discards order per default due to node sorting by size [BSW02] [TS07]. Note, that while all visualizations in this paper use color to emphasize the changes, none of the metrics rely on this visual cue. In this paper, we evaluate how changing container resolutions affect the stability of the SnD layout and three variations of the SQ layout for constant tree data. We generated 600 random trees and evaluate layout changes for 100 different resolutions and four different layout variants. This leads to a total of 240000 treemaps for comparison.

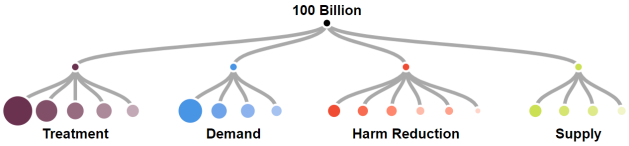


Figure 2: *Tree data from the* New York Times *use-case, as seen as treemaps in Figure 1 and Figure 3.*

2. Used Treemap Algorithms

In our work, we focus on the SnD and SQ treemap layout algorithms. This decision is based on the popularity of SQ and the common usage of SnD as a baseline for comparison. We disregarded treemap supplements such as colors, cushions, cascades, nesting or borders, as they do not influence rectangle placement or metric calculations. While the SnD algorithm has no input parameters, the SQ algorithm tries to optimize rectangles for a given target aspect ratio. Kong et al. [KHA10] found that the initially proposed target ratio of one might not be optimal. Liangfu et al. [LFH*17] alternatively proposed the golden ratio which became the default setting for the D^3 SQ implementation. In total we use the following algorithms:

- The original SnD algorithm [Shn92]
- The original SQ algorithm with a target ratio of 1 (SQ_1.0) [BHVW00]
- The D^3 implementation of the SQ algorithm which inverts the Y-axis for rectangle placing with a target aspect ratio of one (SQ_D^3_1.0) [Bos19]
- The D^3 implementation with a golden target ratio (SQ_D^3_Golden) [LFH*17]

3. Data

Use-Case: We examined the treemap data as seen in Figure 1 used by the *NYT* [Kat18] as a real world use-case to emphasise how our findings affect real-word applications. To thoroughly evaluate the effect of different container resolutions seen in Figure 1, we extracted two levels of tree data. This tree can be seen in Figure 2.

Random Data: In addition to the use-case we generated data

to compare our findings with a broad variety of treemap visualizations. The data generation was inspired by Kong et al. [KHA10] The SnD and SQ algorithms use a recursive approach for converting tree nodes into rectangles. Each decision in a recursion can influence further recursion steps. In the SnD algorithm, the current depth level of a tree determines if rectangle subdivisions are horizontal or vertical. The SQ algorithm variants use the width/height ratio of parent-containers to determine the direction of additional rectangle-rows. Therefore we chose to generate tree-data with a depth-level of two to enable decisions bases on the recursive nature of SnD and SQ layout algorithms. For the tree generation we chose two different leaf node weight sets (1-20, 5-100) and three different sets of total leaf nodes (25, 50, 100). The SQ variants use a greedy approach to optimize aspect ratios. This approach has no degree of freedom to optimize aspect ratios when only two nodes are placed in a single recursion step. Henceforth we ensured that each node, that becomes a parent node, has at least three child nodes. This ensures that the layout algorithms are forced to make layout decisions in each recursion. Each tree was initialized with a root node containing three child nodes, each containing three child nodes themselves. We then selected non-leaf nodes at random to distribute the remaining child nodes for each tree setting. This resulted in 2*3*100 = 600 unique trees.

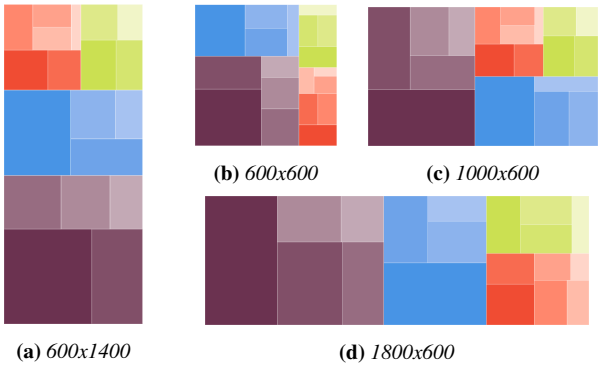
For the treemap construction, we chose ten different container side lengths: 600px, 800px, 1000px, 1200px, 1400px, 1600px, 1800px, 2000px, 2200px, 2400px. This resulted in 10*10 = 100 different combinations for bounding boxes. The screen length sizes of 800px, 1600px and 2400px were inspired by literature [EF10] and common screen sizes. The additional screen sizes were added for a finer distribution and broader evaluation. We chose resolutions over aspect ratios to reflect natural screen sizes. Using the four layout algorithm variants described in section 2, we constructed 1*100*4 = 400 treemap visualizations for our use-case and 600*100*4 = 240000 treemap visualizations from our random tree data.

4. Used Metrics

From the introduced change metrics we chose three, each with a slightly different focus. The *Distance Change* metric (treemap origin) and *Corner Travel Distance* metric (computer vision origin) measure how a rectangle changes shape and moves through space [SW01] [VSC*19]. Both metrics were adapted to calculate changes depending on rectangle sizes relative to container resolutions. Additionally, we used the *Relative Position Change* metric which is closer related to human perception of change [SSV17]. All metrics result in a single value where zero represents no change and one represents maximum change. However only *Relative Position Change* can reach the worst case of one while it is a theoretical bound for *Distance Change* and the *Corner Travel Distance*. Furthermore, we calculated the *Average Aspect Ratio*.

5. Results

The *NYT* data resulted in 100 treemaps with different resolutions for each layout algorithm. Fig. 3 shows four examples of different layouts depending on the container resolution. The original and D^3 SQ algorithms with a target ratio of one produced 42 different layouts each. Furthermore, the Y-axis inversion does not lead to



(a) *600x1400* **(b)** *600x600* **(c)** *1000x600* **(d)** *1800x600*

Figure 3: *Four different treemaps for the* NYT *use-case, constructed with the SQ algorithm and a target ratio of one.*

differences between the produced layouts, except for the expected mirroring. For simplicity we use the standard SQ algorithm to refer to both variants in the result section. The D^3 algorithm variant with a golden target ratio produced 29 unique treemaps. The SnD algorithm produced only one layout, as it does not take the container size into account, i.e. it possesses maximum stability. We chose 600x600 as a baseline for analyzing layout stability.

Figure 4a and Figure 4b show the *Corner Travel* distances of each layout of the SQ and D^3 SQ algorithms in comparison to the baseline for the *NYT* use-case. Figure 4c and Figure 4d show the aggregated results for the same metric on our generated data. The diagonals show, that recalculating without changing the aspect ratio does not affect the layout. However, rounding may still lead to unexpected results. Node weight distribution had a maximum effect of 3% and an average effect of 1% between sets. The number of leaf nodes has a maximum effect of 10% and an average effect of 3% between sets. To calculate the average effect, diagonals were ignored. An increase in leaf nodes increases the average *Corner Travel* distance. The results shown in Figure 4a-4d are not symmetrical, as the SQ algorithm regards squared resolution as wider than high. The width/height ratio determines the orientation of row placement, which results in an orientation flip when starting the layout process [BHVW00]. While a general overall tendency is visible, single treemaps can have specific resolutions where change can be negligible. E.g., the *NYT* use-case shows this for increases in width up to roughly 60% compared to the height starting from a squared container base (see the green numbers in Figure 4b). The *Distance Change* values roughly behave in the same way. While aspect ratios can be a close approximation to resolutions, we observed, that rounding lines on pixels can influence layout decisions.

In addition to the evaluation of layout stability, we analyzed the influence on changing aspect ratios. Figure 5a shows how the aspect ratio changes when using the SnD algorithm on the *NYT* data. As the SnD algorithm is not influenced by different container resolutions, recalculation and rescaling leads to the same values. Figure 5b shows how the aspect ratio of the layout produced by the SQ algorithm changes, when a finished layout is rescaled. When recalculating the average aspect ratio is 1.77, the best aspect ratio

h \ w	600	800	1000	1200	1400	1600	1800	2000	2200	2400
600	0	0.38	0.40	0.22	0.19	0.24	0.23	0.22	0.22	0.30
800	0.25	0	0.38	0.40	0.40	0.22	0.17	0.21	0.24	0.23
1000	0.24	0.27	0	0.38	0.40	0.40	0.40	0.22	0.17	0.19
1200	0.39	0.24	0.27	0	0.38	0.38	0.40	0.40	0.43	0.22
1400	0.42	0.25	0.25	0.27	0	0.38	0.38	0.40	0.40	0.40
1600	0.38	0.39	0.24	0.25	0.27	0	0.38	0.38	0.38	0.40
1800	0.40	0.39	0.25	0.24	0.25	0.27	0	0.38	0.38	0.38
2000	0.40	0.43	0.39	0.24	0.24	0.27	0.27	0	0.38	0.38
2200	0.41	0.38	0.43	0.24	0.23	0.25	0.27	0.27	0	0.38
2400	0.37	0.40	0.42	0.39	0.25	0.24	0.25	0.27	0.27	0

(a) Corner Travel Metric - NYT Data - *SQ_1.0*

h \ w	600	800	1000	1200	1400	1600	1800	2000	2200	2400
600	0	0.04	0.06	0.42	0.37	0.37	0.25	0.26	0.26	0.22
800	0.47	0	0.04	0.04	0.27	0.42	0.37	0.38	0.37	0.25
1000	0.46	0.47	0	0.03	0.04	0.04	0.27	0.42	0.37	0.37
1200	0.25	0.47	0.47	0	0.03	0.04	0.04	0.06	0.27	0.42
1400	0.29	0.35	0.47	0.47	0	0.03	0.04	0.04	0.04	0.27
1600	0.32	0.25	0.47	0.47	0.47	0	0.03	0.04	0.04	0.04
1800	0.40	0.29	0.34	0.47	0.47	0.47	0	0.03	0.03	0.04
2000	0.40	0.29	0.25	0.46	0.47	0.47	0.47	0	0.01	0.03
2200	0.39	0.32	0.29	0.34	0.47	0.47	0.47	0.47	0	0.01
2400	0.44	0.40	0.29	0.25	0.35	0.47	0.47	0.47	0.47	0

(b) Corner Travel Metric - NYT Data - *SQ_D³_Golden*

h \ w	600	800	1000	1200	1400	1600	1800	2000	2200	2400
600	0.00	0.18	0.27	0.29	0.30	0.30	0.30	0.30	0.30	0.30
800	0.38	0.00	0.16	0.23	0.27	0.29	0.29	0.30	0.30	0.30
1000	0.35	0.38	0.00	0.15	0.20	0.26	0.28	0.29	0.30	0.30
1200	0.35	0.36	0.38	0.00	0.14	0.18	0.23	0.27	0.28	0.29
1400	0.36	0.35	0.37	0.39	0.00	0.12	0.17	0.21	0.25	0.27
1600	0.37	0.35	0.36	0.38	0.39	0.00	0.11	0.16	0.20	0.23
1800	0.37	0.36	0.35	0.36	0.38	0.39	0.00	0.10	0.16	0.18
2000	0.37	0.37	0.35	0.35	0.37	0.38	0.40	0.00	0.10	0.15
2200	0.37	0.37	0.36	0.35	0.36	0.37	0.38	0.40	0.00	0.09
2400	0.38	0.37	0.36	0.35	0.35	0.36	0.38	0.38	0.41	0.00

(c) Corner Travel Metric - Random Data - *SQ_1.0*

h \ w	600	800	1000	1200	1400	1600	1800	2000	2200	2400
600	0.00	0.17	0.23	0.27	0.29	0.30	0.30	0.29	0.29	0.30
800	0.36	0.00	0.16	0.20	0.24	0.27	0.29	0.30	0.30	0.30
1000	0.35	0.36	0.00	0.14	0.18	0.22	0.25	0.27	0.28	0.29
1200	0.35	0.36	0.37	0.00	0.13	0.17	0.19	0.23	0.25	0.27
1400	0.34	0.35	0.36	0.37	0.00	0.11	0.16	0.18	0.21	0.24
1600	0.34	0.35	0.35	0.36	0.38	0.00	0.10	0.16	0.18	0.20
1800	0.35	0.34	0.35	0.36	0.36	0.38	0.00	0.10	0.15	0.17
2000	0.35	0.34	0.35	0.35	0.36	0.36	0.38	0.00	0.09	0.14
2200	0.36	0.34	0.34	0.35	0.35	0.36	0.36	0.39	0.00	0.09
2400	0.36	0.35	0.34	0.35	0.35	0.36	0.36	0.37	0.39	0.00

(d) Corner Travel Metric - Random Data - *SQ_D³_Golden*

Figure 4: *Heatmap visualizations for the* Corner Travel Metric *of our aggregated random data and the* NYT *use-case with the container resolution width on the w axis and the height on the h axis*

h \ w	600	800	1000	1200	1400	1600	1800	2000	2200	2400
600	2.08	2.33	2.71	3.15	3.64	4.15	4.66	5.18	5.69	6.21
800	2.08	2.08	2.26	2.51	2.82	3.16	3.52	3.90	4.28	4.66
1000	2.25	2.05	2.08	2.21	2.39	2.63	2.87	3.15	3.44	3.74
1200	2.49	2.15	2.04	2.08	2.19	2.33	2.51	2.71	2.92	3.15
1400	2.77	2.31	2.10	2.04	2.08	2.17	2.29	2.43	2.60	2.77
1600	3.09	2.48	2.21	2.08	2.04	2.08	2.16	2.26	2.37	2.51
1800	3.42	2.69	2.34	2.15	2.06	2.04	2.08	2.15	2.24	2.38
2000	3.76	2.92	2.49	2.25	2.12	2.05	2.04	2.08	2.15	2.22
2200	4.11	3.17	2.65	2.36	2.19	2.09	2.05	2.04	2.08	2.14
2400	4.47	3.41	2.83	2.48	2.28	2.15	2.08	2.04	2.05	2.08

(a) *Aspect Ratios for Rescaling/Reconstruction* - NYT Data - *SnD*

h \ w	600	800	1000	1200	1400	1600	1800	2000	2200	2400
600	1.87	2.09	2.38	2.80	3.23	3.67	4.10	4.54	4.98	5.43
800	1.89	1.87	2.02	2.23	2.48	2.80	3.12	3.45	3.77	4.10
1000	2.07	1.86	1.87	1.99	2.15	2.32	2.55	2.80	3.06	3.32
1200	2.35	1.96	1.85	1.87	1.96	2.09	2.23	2.38	2.59	2.80
1400	2.69	2.12	1.91	1.84	1.87	1.95	2.05	2.17	2.29	2.44
1600	3.04	2.35	2.02	1.89	1.84	1.87	1.94	2.02	2.12	2.23
1800	3.40	2.61	2.16	1.96	1.87	1.84	1.87	1.93	2.00	2.09
2000	3.77	2.87	2.36	2.07	1.93	1.86	1.84	1.87	1.92	1.99
2200	4.14	3.13	2.55	2.19	2.01	1.90	1.85	1.85	1.87	1.92
2400	4.51	3.41	2.76	2.34	2.10	1.96	1.89	1.85	1.85	1.87

(b) *Aspect Ratios for Rescaling* - NYT Data - *SQ_1.0*

Figure 5: *Heatmap visualizations for the* Aspect Ratio Metric *when rescaling/recalculating* NYT *use-case treemaps with the container resolution width on the w axis and the height on the h axis*

is 1.58 and the worst aspect ratio is 1.96. Henceforth, recalculating a SQ treemap is beneficial for aspect ratios but detrimental for layout changes. The opposite holds true for rescaling SQ treemaps.

6. Conclusion

In sum, the following guidelines can be derived:

- If recalculation is not needed and expected container resolutions vary by less than 10%, rescaling can be a viable option for the SQ layout.
- Prefer the SnD algorithm over the SQ algorithms when recalculating treemaps for different container resolutions.
- If you need to use the SQ algorithm when recalculating treemaps, evaluate if there are stable resolutions near the desired new resolution.

These guidelines are meant to keep visualizations with the same data as similar as possible for different display sizes. As there seems to be a trade off between layout stability and aspect ratio sizes, it is advisable to keep container resolutions fixed if possible.

It is always advisable to refer to a distinct treemap layout algorithm, instead of treemaps as a general visualization type. The broad range of algorithms exhibits a lot of different strengths and weaknesses that are not shared or necessarily typical for *the* treemap visualization. This is especially important, when conducting user studies, as the findings can always only be attributed to the tested algorithms.

7. Future Work

We explored the effect of container resolutions for four different treemap layout variants. To completely understand the effect of container resolutions, more treemap layout algorithms and the effect of pixel rounding need to be evaluated. Furthermore, extensive user studies are necessary to examine the interference of layout changes on the mental map of the user. E.g. how much can a layout change before a user has to completely relearn rectangle positions. In this regard, it is additionally interesting how the familiarity of a layout may influence the understanding of different layout choices. The overall goal of future work is to evaluate which current algorithm is suitable under given constraints.

References

[BHVW00] BRULS M., HUIZING K., VAN WIJK J. J.: Squarified treemaps. In *Data Visualization 2000*. Springer, 2000, pp. 33–42. 2, 3

[Bos19] BOSTOCK M.: d3-hierarchy. https://github.com/d3/d3-hierarchy/blob/master/src/treemap/squarify.js, 2019. 1, 2

[BSW02] BEDERSON B. B., SHNEIDERMAN B., WATTENBERG M.: Ordered and quantum treemaps: Making effective use of 2d space to display hierarchies. *AcM Transactions on Graphics (TOG) 21*, 4 (2002), 833–854. 2

[CDY17] CHEN Y., DU X., YUAN X.: Ordered small multiple treemaps for visualizing time-varying hierarchical pesticide residue data. *The Visual Computer 33*, 6-8 (2017), 1073–1084. 2

[EF10] ELMQVIST N., FEKETE J.-D.: Hierarchical aggregation for information visualization: Overview, techniques, and design guidelines. *IEEE Transactions on Visualization and Computer Graphics 16*, 3 (May 2010), 439–454. URL: http://dx.doi.org/10.1109/TVCG.2009.84, doi:10.1109/TVCG.2009.84. 3

[HBD17] HAHN S., BETHGE J., DÖLLNER J.: Relative direction change-a topology-based metric for layout stability in treemaps. In *VISIGRAPP (3: IVAPP)* (2017), pp. 88–95. 2

[HD17] HAHN S., DOELLNER J.: Hybrid-treemap layouting. In *Proceedings of the Eurographics/IEEE VGTC Conference on Visualization: Short Papers* (2017), Eurographics Association, pp. 79–83. 2

[Kat18] KATZ J.: How a police chief, a governor and a sociologist would spend $100 billion to solve the opioid crisis. https://www.nytimes.com/interactive/2018/02/14/upshot/opioid-crisis-solutions.html, Feb 2018. New York Times online article, accessed on 22.02.2018. 1, 2

[KHA10] KONG N., HEER J., AGRAWALA M.: Perceptual guidelines for creating rectangular treemaps. *IEEE transactions on visualization and computer graphics 16*, 6 (2010), 990–998. 2, 3

[LFH*17] LU L., FAN S., HUANG M., HUANG W., YANG R.: Golden rectangle treemap. In *Journal of Physics: Conference Series* (2017), vol. 787, IOP Publishing, p. 012007. 2

[MELS95] MISUE K., EADES P., LAI W., SUGIYAMA K.: Layout adjustment and the mental map. *Journal of Visual Languages & Computing 6*, 2 (1995), 183–210. 2

[MTE19] MTES N.: What is comstock and why is it important to utilities?, 2019. URL: https://twitter.com/NREL_MechTherm/status/1215372692838404101. 1

[MW02] M. WESKAMP D. A.: Newsmap, 2002. URL: http://newsmap.jp/. 1

[Shn92] SHNEIDERMAN B.: Tree visualization with tree-maps: 2-d space-filling approach. *ACM Transactions on graphics (TOG) 11*, 1 (1992), 92–99. 1, 2

[SSV17] SONDAG M., SPECKMANN B., VERBEEK K.: Stable treemaps via local moves. *IEEE transactions on visualization and computer graphics 24*, 1 (2017), 729–738. 2, 3

[SW01] SHNEIDERMAN B., WATTENBERG M.: Ordered treemap layouts. In *IEEE Symposium on Information Visualization, 2001. INFOVIS 2001.* (2001), IEEE, pp. 73–78. 2, 3

[TS07] TU Y., SHEN H.-W.: Visualizing changes of hierarchical data using treemaps. *IEEE Transactions on Visualization and Computer Graphics 13*, 6 (2007), 1286–1293. 2

[Tur03] TURO D.: Hierarchical visualization with treemaps: making sense of pro basketball data. In *The Craft of Information Visualization*. Elsevier, 2003, pp. 237–238. 1

[VSC*19] VERNIER E., SONDAG M., COMBA J., SPECKMANN B., TELEA A., VERBEEK K.: Quantitative comparison of time-dependent treemaps. *arXiv preprint arXiv:1906.06014* (2019). 2, 3

[WD08] WOOD J., DYKES J.: Spatially ordered treemaps. *IEEE transactions on visualization and computer graphics 14*, 6 (2008), 1348–1355. 2