Bubble Treemaps for Uncertainty Visualization

Jochen Görtler, Christoph Schulz, Daniel Weiskopf, *Member, IEEE Computer Society*, and Oliver Deussen

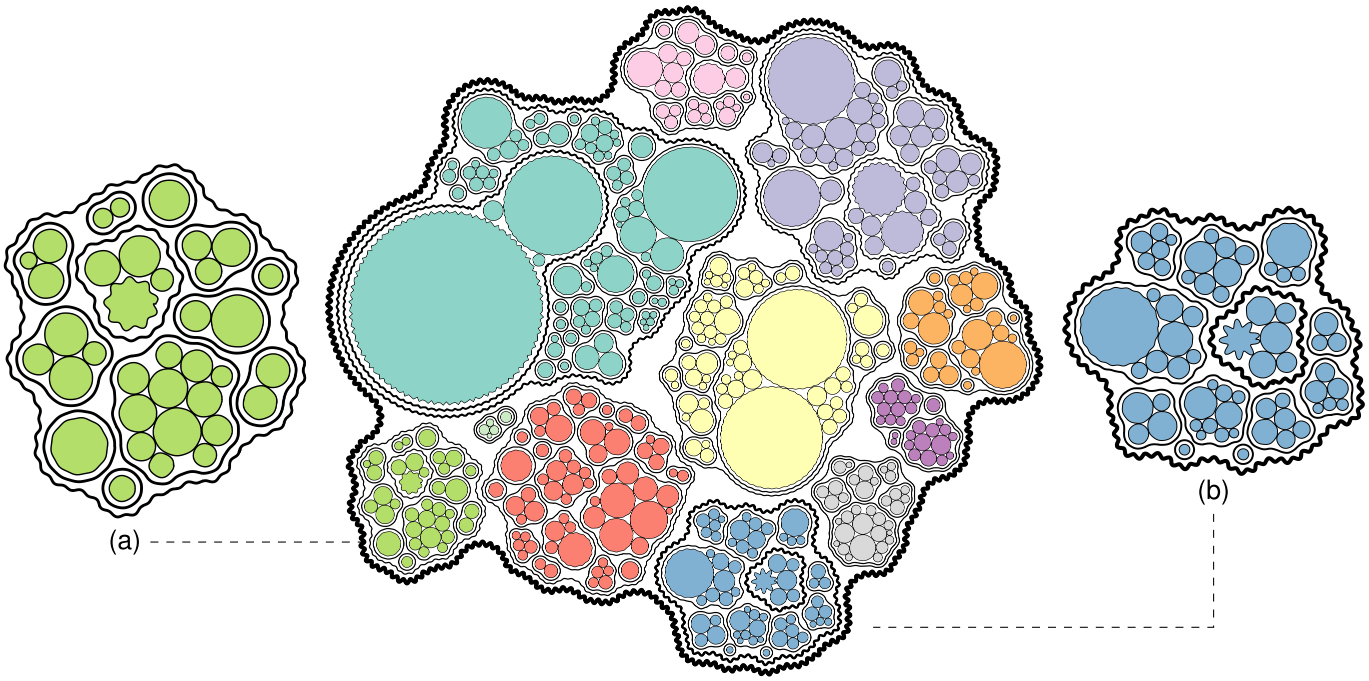


Fig. 1. Bubble Treemap of the S&P 500 index, decomposed into sectors and companies. Uncertainty arises from aggregating one week of stock data in November 2016. Each circle represents a stock, its area is proportional to the mean closing price, whereas the standard deviation is depicted using the outlines. Our visualization helps to discover a medium-sized sector with low uncertainty and assess its composition (a), as well as a sector with high uncertainty and the company that mostly introduced it (b).

Abstract—We present a novel type of circular treemap, where we intentionally allocate extra space for additional visual variables. With this extended visual design space, we encode hierarchically structured data along with their uncertainties in a combined diagram. We introduce a hierarchical and force-based circle-packing algorithm to compute Bubble Treemaps, where each node is visualized using nested contour arcs. Bubble Treemaps do not require any color or shading, which offers additional design choices. We explore uncertainty visualization as an application of our treemaps using standard error and Monte Carlo-based statistical models. To this end, we discuss how uncertainty propagates within hierarchies. Furthermore, we show the effectiveness of our visualization using three different examples: the package structure of Flare, the S&P 500 index, and the US consumer expenditure survey.

Index Terms—Uncertainty visualization, hierarchy visualization, treemaps, tree layout, circle packing, contours

1 Introduction

Hierarchical data plays a significant role in information visualization since many datasets are inherently hierarchical, or are purposefully made hierarchical. Thus, many different representation methods for hierarchies have been developed [37]. Node-link diagrams represent such structures by nodes that are connected by edges. In contrast to that, implicit representations focus on the value within each node and encode the hierarchy through inclusion. Many implicit methods, such as treemaps, divide the canvas in accordance to the relative sizes of the respective sub-hierarchies and this way optimize for space, which leads to compact and scalable representations.

In our opinion, however, there is a trade-off between compactness and readability—if a treemap is very compact, the underlying structure is difficult to grasp since the compactness does not leave space for grouping cues or other visual features. With traditional treemaps [25], it is not feasible to encode additional information such as uncertainty in a geometric way. Furthermore, inclusion via area suggests additive propagation, which might not be the case for uncertainty. In contrast, if much space is wasted, e.g., with circular treemaps [46], visual scalability suffers to a point where interaction is almost mandatory to reduce the visualized data.

We strive to find a good compromise between compactness and readability with our new visualization technique: *Bubble Treemaps*.

• *Jochen Görtler and Oliver Deussen are with the University of Konstanz. E-mail: firstname.lastname@uni-konstanz.de*
• *Christoph Schulz and Daniel Weiskopf are with VISUS, University of Stuttgart. E-mail: firstname.lastname@visus.uni-stuttgart.de.*

Manuscript received 31 Mar. 2017; accepted 1 Aug. 2017.
Date of publication 28 Aug. 2017; date of current version 1 Oct. 2017.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TVCG.2017.2743959

They allocate extra space in the layout to encode certain and uncertain information together in a geometric manner, similar to error bars. Leaf nodes of a hierarchy are encoded as circles and enclosed by an arc-based parameterizable contour. Contours of sibling sub-hierarchies are packed using a force-directed model and enclosed by another parameterizable contour. This process is recursively continued until the whole tree is traversed. This way, we can encode additional group-level information, such as uncertainty, into the visual representation of the contours. Figure 1 shows a typical example of a Bubble Treemap, emphasizing nodes with high uncertainty using deformations of the contour (amplitude and frequency) to resemble variability of box-plots while maintaining treemap-typical color coding of higher-level nodes.

We consider uncertainty as a *distribution* of possible values per node, as opposed to a single and *exact* value. Showing the mean value alone is often not sufficient to describe a distribution. Instead, we need visualizations that are capable of displaying additional statistical features that help the reader gain a better understanding of the data. Many visual variables do not work well for illustrating uncertainty [29]. As noted by Hullmann [24], uncertainty visualization is error-prone, especially, when drawing false conclusions because of bad communication regarding the underlying statistical model, e.g., confusing standard deviation with variance. Our technique offers great flexibility to choose appropriate encodings, depending on the task and underlying model—it even works well in black and white.

Our contribution is threefold: First, we propose a layout based on circle packing to use space purposely, achieving a reasonable trade-off between a compact representation of the hierarchy and its inherent information. Second, we define node contours analytically, resulting in a new parameter domain to be used for additional visual variables, in particular, for uncertainty visualization. Third, we describe different models of uncertainty and discuss their relation to hierarchal data. Our discussion is rounded up by demonstration of our technique using three example data sets. An implementation of Bubble Treemaps can be found online[1].

2 Related Work

The following section gives an overview of work related to our method. First, we review the state of the art in visualizing hierarchical data. Next, we provide a summary of recent work for visualizing set memberships, since this topic is closely related to how our method encodes the topology of an underlying tree structure. At last, we describe recent methods of uncertainty visualization for graphs.

Visualization of Hierarchical Data

There are many different methods to visualize hierarchical data; Schulz et al. [38] provide an extensive survey of implicit hierarchy visualization. Treemaps have been shown to be effective at conveying hierarchical information. Most of the traditional treemap approaches, such as Squarified Treemaps [11] or Voronoi Treemaps [4] follow a top-down strategy, recursively subdividing a given area according to the underlying hierarchy. Similarly, Auber et al. [3] describe a treemap layout algorithm that produces irregular nested shapes by subdividing the Gosper curve. The boundaries of the areas, however, are not incorporated explicitly into the layout—the contours are inlaid retroactively and not used to encode additional quantitative values.

Circle packing has been widely studied in theoretical computer science, especially for its connection to planar graphs [14]. Stephenson [40] provides a summary of the general field of circle packing. Usually, a more pragmatic approach is pursued for its application to hierarchy visualization: Wetzel [44] and Wang et al. [43] propose methods based on nesting circles in a bottom-up fashion, which is later refined by Zhao and Lu [46]. Viegas et al. [42] use a combination of circle packing together with a *balloon layout* to visualize information flow in social networks. Several domain-specific works combine dense packing of circles with layout methods from graph drawing such as Bubble Trees or radial layouts [2, 23]. We utilize the method by Wang et al. to create our initial packings (Figure 2).

[1]`https://github.com/grtlr/bubble-treemaps`

McGuffin and Robert [30] provide an extensive study on the space efficiency of different tree representation methods. They introduce a novel metric that aims to measure the distribution of area across nodes in hierarchical visualizations. One conclusion of their work is that a perfect partitioning of the space might not be ideal when additional information (for example labels) needs to be displayed. Similarly, Schulz et al. [38] argue that packing the space too tightly conceals the underlying structure and methods that deliberately leave empty space would enable a better perception of the tree structure. These findings were an inspiration for us when developing Bubble Treemaps.

Bubble charts are often used to visualize three-dimensional data. To create a bubble chart, two dimensions are mapped to the x and y axes of the plane, while the third dimension is mapped to the size of a circle at the corresponding position. Bubble charts are commonly used on websites [12, 18], often they are part of an interactive exploration tool for the data. Sometimes, however, either the category of the entities or their hierarchical structure is lost.

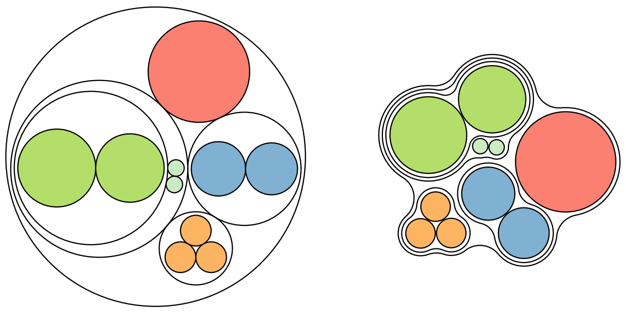

(a) Circular Treemap (b) Bubble Treemap

Fig. 2. Bubble Treemaps are initialized using a circular treemap layout and subsequently compacted using a force-based approach.

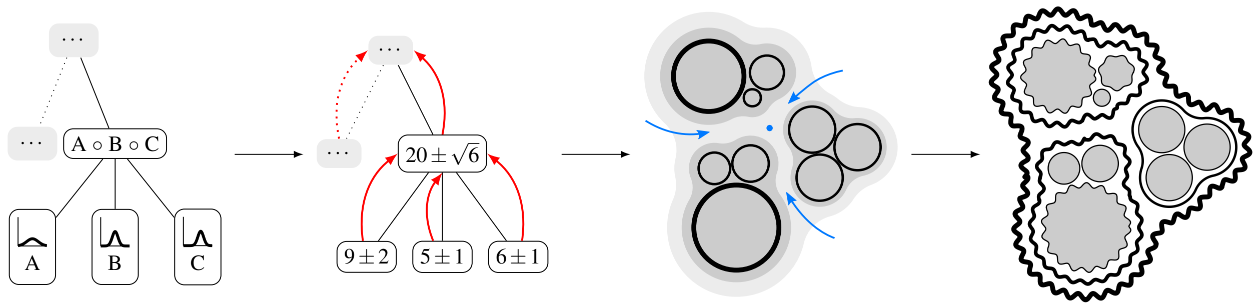
Contours and Set Membership

As mentioned above, treemaps encode a hierarchy implicitly by aggregating the areas of the child nodes into the area of the parent node. We do this similarly: all child nodes are enclosed by a contour that represents the current node. As a result, our approach shares similarities with methods that depict set membership for spatially embedded objects. For example, Bubble Sets [13] use marching squares (a 2D version of the marching cubes algorithm [28]) to draw contours around embedded objects, while we use arcs. Kelp diagrams [17] and especially the refined version Kelp Fusion [31] share more similarity, even though they were developed for geographical data. Notably, the authors mention the potential of enclosing areas by contours based on arcs but do not provide details. Riche and Dwyer [35] present a method that builds upon Euler diagrams to visualize set membership by drawing contours around objects of the same logical group while minimizing the number of crossings between contours of different groups. In the last years, several methods have been developed to improve Euler diagrams: force-directed methods are utilized to optimize their respective layout [33] and to find smoother boundaries [39]. There also exists work on drawing area-proportional realizations of Euler-like diagrams [32]. While Euler-like diagrams have similarities to our approach, they do not take hierarchical structures into account.

Our method for drawing contours shares similarities with *approximate solvent-accessible surface areas*, which come from the field of biomolecules. They model the surface area that is accessible by a probe with a fixed radius [27]. There exist efficient algorithms that compute these analytical surfaces, but they make assumptions on the structure of the molecules that do not hold for the general case of arbitrary intersecting spheres [21].

Visualization of Uncertainty

Historically, the representation of uncertainty received broad attention in scientific visualization [10, 34]. Uncertainty, however, can also be present in different data sources of information visualization [1] and visual analytics [15].



(a) Uncertain Hierarchy (b) Uncertainty Propagation (c) Layout Algorithm (d) Rendering

Fig. 3. Overview of the Bubble Treemap method. We start with measured distributions organized in a hierarchy (a). We usually only know the leaf level. By applying a suitable uncertainty model, we propagate characteristics of the underlying distribution toward the root (b). Then, we compute the treemap layout using circular arcs and a force-based model (c). Finally, we draw leaf circles and inner-node contours around each level of the hierarchy (d).

Bertin [5] and MacEachren et al. [29] study various visual variables for uncertainty regarding intuitiveness and performance in map reading tasks. These visual variables are further refined by Guo et al. [20] for graph edges. Gschwandtner et al. [19] compare different representations of uncertainty of temporal data in the form of time intervals. Hullmann [24] investigates the evaluation of uncertainty visualizations and shows that different study designs have a strong influence on the result. Apart from the visual variables described by Boukhelifa et al. [9], most visual variables are used for representing areas or volumes and cannot be used to encode uncertainty into shapes. In the context of other domains, several methods have been developed to encode uncertainty information directly into geometry. Khlebnikov et al. [26] deliberately introduce noise into multivariate volumetric rendering to blend multiple variables. Some work visualizes uncertainty in node-link diagrams. Schulz et al. [36] propose a method to perform graph layouts for probabilistic networks. The statistics package *R* can show decision trees with uncertainty in their leafs. While we apply several visual variables and ideas from explicit node-link diagrams for uncertainty visualization, we deal with implicit depiction of hierarchal data and inclusion relations.

3 Overview

Given a hierarchy of values with uncertainty in the form of additional attribute values for the leafs and an aggregation model (Figures 3a and 3b), we construct a Bubble Treemap by extracting characteristics from distributions for each level of the hierarchy, then mapping these characteristics to circles and analytically defined arc-based contours for leafs and inner nodes, respectively. To achieve a compact layout, we implement a force-directed model (Figure 3c).

A hierarchy with uncertainty is represented by a tree $T = (V, E, A)$, similar to a regular tree in graph theory, with vertices V (nodes), edges E, and attribute vectors $\mathbf{a}_i \in A \subseteq \mathbb{R}^n$ associated with each node. Typical attribute vectors of interest would be the mean and standard deviation $(\mu, \sigma)_i$. Our visualization, however, is not limited to one characteristic, instead, we aim to display multiple statistical properties of the underlying distributions at once. We achieve this by not striving for a perfect partitioning of the space, but rather purposefully allocating space that we parametrize and then use to encode such additional information.

In the next sections, we describe how to model and propagate uncertainty, construct arc-based contours, and compute our treemap layout. Afterward, we discuss the usage of visual variables for uncertainty within Bubble Treemaps, followed by three example data sets from different domains to demonstrate the usefulness of our method. Finally, we discuss implementation details and limitations of our method.

4 Models and Propagation of Uncertainty

An important factor in uncertainty visualization is understanding the underlying model. Often, uncertainty influences data, due to measurement errors, incomplete information, or inference errors. For example, to deal with measurement imprecision and show statistical significance, we often perform several measurements and aggregate them to an ensemble. Usually, we are interested in several characteristic numbers, instead of full-blown probability density functions, because numbers are easier to work with—just think of the mean and standard deviation. Regarding hierarchies, models for uncertainty dictate what information to depict, not just for leaf nodes, but also for parent nodes at each inner level. To capture and effectively visualize these characteristics, we have to have a basic understanding of different sources of uncertainty and their propagation within the hierarchy:

Probabilities We consider probability density functions (PDFs) p_i as a basic building block. Each PDF maps the value of a continuous random variable x_i to probability density p_i:

$$p_i : \mathbb{R} \to \mathbb{R}_{\geq 0}, \text{ where } \int_{-\infty}^{\infty} p_i(x_i)\, dx_i = 1 \tag{1}$$

Depending on conditional dependencies between the random variables, the joint PDF may resemble anything between the chain rule and Bayesian networks, but its integral is always one. Furthermore, the chance that a certain outcome occurs is quite abstract. For this reason, we expand our discussion using a more aggregated model that relates probability to value.

Expected Value and Standard Deviation Let us assume that we collect data for each node individually to obtain a distribution. To express characteristics of such a distribution, we usually resort to the expected value μ_i and standard deviation σ_i:

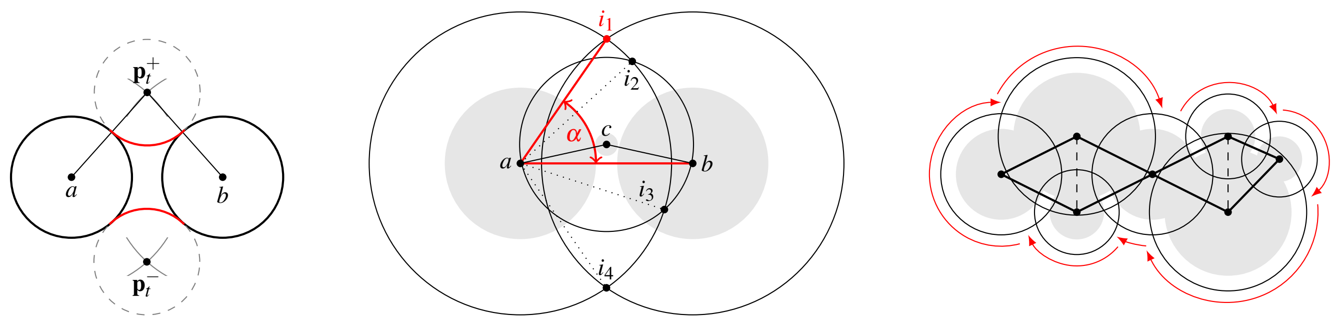
$$\mu_i = \int x\, p_i(x_i)\, dx_i, \quad \sigma_i = \sqrt{\int x^2\, p_i(x_i)\, dx_i - \mu^2} \tag{2}$$

This formula also suggests why error bars and similar techniques are so popular: they match our natural perception of fluctuation in terms of distance. We take this as another reason to encode certain and uncertain information in geometry, i.e., to maintain the relation between expected value and standard deviation.

Propagation Methods Regarding the propagation from children to parent, assuming linear aggregation and independence among the children, the expected values $\mu_{1,\ldots,n}$ add up, while the standard deviations $\sigma_{1,\ldots,n}$ aggregate using the Euclidean norm:

$$\mu_{1,\ldots,n} = \sum_i^n \mu_i, \quad \sigma_{1,\ldots,n} = \sqrt{\sum_i^n \sigma_i^2} \tag{3}$$

In this case, the propagation is quite geometrical (plain addition and length of a vector) with expected value propagation matching the



(a) The tangent arcs of two circles can be constructed from the intersection points $\mathbf{p}_t^{\pm}$ of their enlarged versions.

(b) Selection of the next circle from the envelope. The leftmost intersection point i_1 is shown in red. Note that the circle c is skipped in this configuration.

(c) The intersection graph of a set of circles. The original circles in gray are enlarged to reflect the smoothness parameter s. The red arrows show the traversal that computes the envelope.

Fig. 4. Different steps that are needed to construct the contour: (a) construction of tangent arcs, the basic primitive of our contours; (b) selection procedure for the envelope; (c) traversal order of the intersection graph.

treemap metaphor. This propagation further justifies our encoding in leaf size and contour width. The more generic and complicated approach supporting non-linear dependencies can be computed using Taylor series. Though, for complex models another approach is usually taken to work around computational and complexity issues.

Monte Carlo-based Methods To avoid a complete survey and calculation, we usually sample a carefully selected subset of the entire population to infer representative information about the entire population. We distinguish the uncertainty induced by the samples and the uncertainty of the Monte Carlo model itself. The propagation model of the latter can be used for testing conditional dependence: For example, let us assume that we have measured hierarchical geographic data and aggregate measurement errors from leafs to root. If nodes are independent, the propagated error increases with the standard deviation. If nodes depend on each other, the error might decrease, due to higher support from measurement points. To recall, non-linearities or dependencies should be ascertainable, because of the geometrically emergent properties of propagation of expected value and standard deviation.

Within the limited scope of this work, covering the all statistical characteristics and models for uncertainty out there is impossible.

5 Circular Arc Contours

In the spirit of treemaps, we recursively draw inner-node contours as arc-based entities around leaf circles to depict parent-child relationships. Instead of using an implicit description of the contour together with, for example, the marching squares algorithm for rendering, we construct the contour in the form of a parametric curve. This approach has the benefit of having an analytically defined model, which we utilize to encode an additional attribute dimension. We do not need to discretize space to draw the contour (which could lead to discontinuities), and it allows us to describe the contour directly using arc segments. Furthermore, as a small advantage, our Bubble Treemaps do not require color and can be completely described using vector graphics. In the following section, we describe the different steps that are necessary to construct the contour—Algorithm 1 describes the complete procedure.

For a given inner node $n_i \in V$, we need to find an enclosing surface that includes all circles of $leafs(n_i)$. Computing the enclosing contour consists of three parts: First, we need to find the circles that make up the envelope $E_i \subset leafs(n_i)$, which is described in Section 5.2. The envelope contains the subset of circles that are exposed at the outside of the set. By iterating over these circles, we can construct the contour as a circular arc spline defined as a sequence of biarc curves [7]. A biarc curve consists of two arc segments that share the same tangent direction at the connection point, which leads to a smooth transition (G^1 continuous). Therefore, all elements stay circular, which makes layout computation (Section 6) much easier.

5.1 Parameters

There are several parameters that describe the space requirements of the contour. These parameters can be defined separately for each node n_i. The margin m_i describes how far the contour will be placed from the underlying structure. The parameter w_i reflects the width of the contour, which is important if we want to encode additional information directly into the contour, or use the contour to emphasize the structure of the tree. At last, the padding p_i models how close adjacent objects can be placed. Please note that there exists such a tuple of parameters for each $n_i \in V$ of the tree (leafs and aggregate nodes alike). In many cases, we are only interested in the total amount of space that is required for the contour, which we define as $d_i = m_i + w_i + p_i$ for a corresponding node n_i.

In addition to the parameters inherent to a given node, each level of the tree is assigned a smoothness parameter s_i. This parameter controls how tightly the contour will fit around the $children(n_i)$ and represents the radius that is used for the tangent arc. Varying s allows us to adopt several concepts from computational geometry; the influence of this parameter on the contour will be discussed in Section 9.

The basic primitive of our method is the construction of the tangent arc. Formally, given two circles a and b with center points at $\mathbf{p}_a$, $\mathbf{p}_b$ and radii r_a, r_b and the desired radius of the tangent arc $r_t \geq \|\mathbf{p}_a - \mathbf{p}_b\| - (r_a + r_b)$, we can find the center of t by virtually enlarging a and b by r_t to obtain a' and b':

$$r'_a = r_a + r_t \quad \text{and} \quad r'_b = r_b + r_t$$

The intersection $(a' \cap b')^+$ gives us the center p_t of t. Truncating this circle (with radius r_t) to the length between the two tangent points of t with a and b gives us the desired tangent arc. Figure 4a outlines the construction with tangent arcs shown in red.

5.2 Finding the Envelope

For finding the envelope E_i, we virtually enlarge each circle of $leafs(n_i)$ by $d_i + s_i$. The result of this step is the set of circles C'. We can now compute the *intersection graph* of C', a graph that contains an edge (c_i, c_j) with $c_i, c_j \in C'$ iff $c_1 \cap c_2 \neq \emptyset$. For the sake of simplicity, we assume the graph to be connected. In case of a disconnected intersection graph, a larger smoothness parameter s_i should be chosen. Next, we find the circle with the leftmost point among all elements of C', which has to be part of E_i by construction. Starting from this element, we can traverse the intersection graph, always choosing the edge that leads to the circle with the leftmost intersection point (as shown in Figure 4c). The selection procedure is shown in Figure 4b. Here, the current circle is a and we consider the intersection points $i_1, \ldots, i_4$, comparing their respective angles to $\mathbf{v} = \mathbf{p}_b - \mathbf{p}_a$. We only need to consider the angles that are counter-clockwise to $\mathbf{v}$. From those, we choose the largest one

(α in this case). Note that it is not sufficient to find the outer face of the embedded intersection graph because there might be small circles that are skipped depending on the specified smoothness s, which is the case for circle c in Figure 4b—the proposed selection method solves this problem.

5.3 Constructing the Contour

We use E_i to construct the contour. To create the circular arc spline, we first add a tangent arc to each neighboring pair of circles. In the second step, we convert these circles to arcs. Then, we set the start angle α and the length θ of the arc segment by converting the left and right neighbors of each circle to polar coordinates centered at the current circle. It is important to note that one needs to handle the cases of *inward arcs*, which are oriented clockwise, and *outward arcs*, which turn counter-clockwise.

The described method only works if the intersection graph is connected. Additionally, the smoothness parameter s is constrained by the maximal distance $d = \|\mathbf{p}_a - \mathbf{p}_b\|$ between two circles $a, b \in C_i$, so that the contour will not intersect itself:

$$s \geq {r'_a}^2 - \frac{({r'_a}^2 - {r'_b}^2 + d^2)^2}{4d^2} \tag{4}$$

If a and b move further away from each other, the tangent arc will move into the gap in between. By constraining smoothness this way, we prohibit that the tangent arc moves across the line segment connecting the centers $\mathbf{p}_a$ and $\mathbf{p}_b$ of the two circles. Limiting s as described in Eq. (4) also covers the case where the radius of the tangent arc is too small to find a tangent point for each a and b, which is the case when a' and b' do not intersect.

Algorithm 1 Construction of the contour

```
 1: procedure CONTOUR($n_i$)
 2:     let $E$ be a sequence that represents the envelope
 3:     $C' \leftarrow$ enlarge $leafs(n_i)$ by $d_i + s_i$
 4:     $c \leftarrow$ element from $C'$ with leftmost extent
 5:     $E$.push($c$)
 6:     $c \leftarrow$ circle with leftmost intersection
 7:     while $c \neq E[0]$ and $c$ has unvisited leftmost intersection do
 8:         $E$.push($c$)
 9:         $c \leftarrow$ circle with leftmost intersection
10:     let $R$ be a sequence that will hold the contour
11:     for all adjacent pairs $c_1, c_2 \in E$ do
12:         $t \leftarrow$ tangent arc between $c_1$ and $c_2$
13:         $t_1, t_2 \leftarrow$truncate $c_1$ and $c_2$ to $t$
14:         $R$.push($t_1, t, t_2$)
15:     return $R$
```

6 Layout Algorithm

Similar to circular treemaps [43, 44, 46], our layout algorithm maps one attribute dimension, such as the expected value, to the area of the circles. The topology of the underlying tree is encoded implicitly through containment, i.e., the area of a child lies completely within the area of its parent node. To achieve a compact representation, we use an adapted version of the circular treemap algorithm to initialize our layout, similar to the one described by Wang et al. [43]. Afterward, we traverse the hierarchy bottom-up and perform a force-based circle packing while accounting for the space that is occupied by the contours of the respective sub-hierarchies.

Once we have the initial layout, we perform a post-order traversal over the circular treemap and transform it to a Bubble Treemap. We achieve this by constructing a spring-based system, as shown in Figure 5, for each sub-hierarchy. The post-order traversal that visits each node $n_i \in V$ starts from the leafs, which at first remain in the arrangement that was determined by the circular layout algorithm. In subsequent steps of the traversal, the elements C_k of each child $k \in children(n_i)$ are grouped together using a contour G_k, as explained

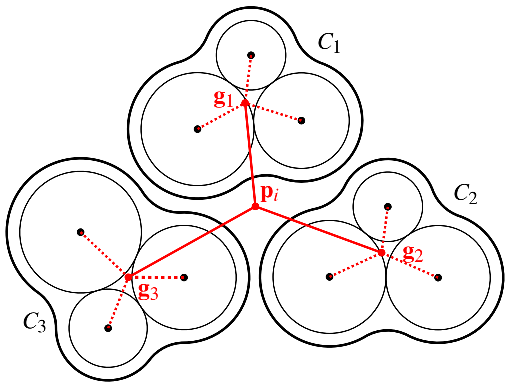


Fig. 5. Schematic of the force-based method for two levels of the hierarchy. After laying the children out in the first step (dashed), they become fixed and will be moved as a whole in the second step (solid).

in Section 5. Depending on the structure of the tree, the elements of C_k can either be circles, coming from leaf nodes, or contours that were already constructed in previous steps. We can interpret C_k as a rigid body, with mass distributed according to its area, on which external forces can be applied. Then, we define the center of the current (circular) node n_i as the center of a spring system with a fixed position $\mathbf{p}_i$. Next, we compute the center of mass $\mathbf{g}_k$ for each k and connect it to $\mathbf{p}_i$ using a spring.

Figure 5 shows an example of such a setup for two levels of a hierarchy. The springs can be seen as attractors that pull each C_k toward $\mathbf{p}_i$. We then simulate the forces in the system using a physics engine, avoiding collisions between each C_k, to create a force-based layout. In practice, we found that approximating the contours by virtually enlarging each circle of $leafs(n_i)$, to the extent of the contour, already yields good results while simplifying the configuration of the physical simulation.

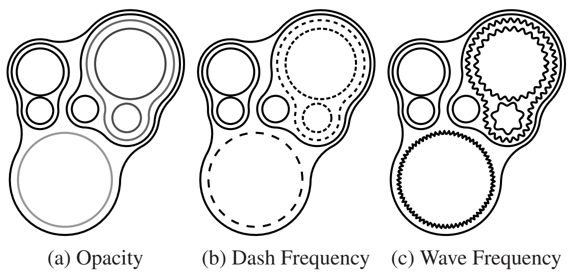
After computing such a force-based layout for n_i, the relative positions of the elements of C_k to each other are fixed and are subsequently transformed as a whole in later steps of the post-order walk. A detailed description of the algorithm is shown in Algorithm 2.

Algorithm 2 Hierarchical Bubble Treemap layout

```
Require: $n_i$ is a node of a tree with circular layout
 1: procedure LAYOUTNODE($n_i$)
 2:     $\mathbf{p}_i \leftarrow$ center of $n_i$
 3:     $W[] \leftarrow$ empty list of rigid bodies
 4:     for all $k \in children(n_i)$ do
 5:         LAYOUTNODE($k$)
 6:         $C_k \leftarrow$ list of elements for each $k \in children(n_i)$
 7:         for all $C \in C_k$ do
 8:             $G \leftarrow$ create rigid body from CONTOUR(C)
 9:             $\mathbf{g}_C \leftarrow$ center of mass of G
10:             connect $\mathbf{g}_C$ to $\mathbf{p}_i$ using a spring
11:             $W$.push($\mathbf{g}_C$)
12:     SIMULATEFORCES(W)
Require: $T$ is a tree with circles in the leafs.
 1: procedure BUBBLETREEMAPLAYOUT($T$)
 2:     root $\leftarrow$ CIRCULARTREEMAPLAYOUT($T$)
 3:     LAYOUTNODE(root)
 4:     return root
```

7 Visual Variables

Visual variables in the context of diagrams and maps have been investigated extensively [5]. The expected value (node area) simply adds up from the leafs to the root, as for all the other treemaps, and is very similar to the application of visual variables to maps. Please note that we could encode uncertainty *inside* nodes, e.g., using radial gradients like Vehlow et al. [41], at the cost of a design dimension to encode additional information. Instead, we restrict our discussion to the encoding of uncertainty on the contour.



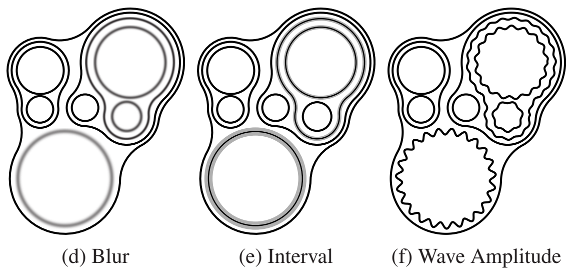


Fig. 6. Example visual variables applied to the contour. Opacity (a), dash frequency (b), and wave frequency (c) can be used if a constant contour width is desired, whereas blur (d), interval (e), and wave amplitude (f) can be used if a variable contour width is permitted.

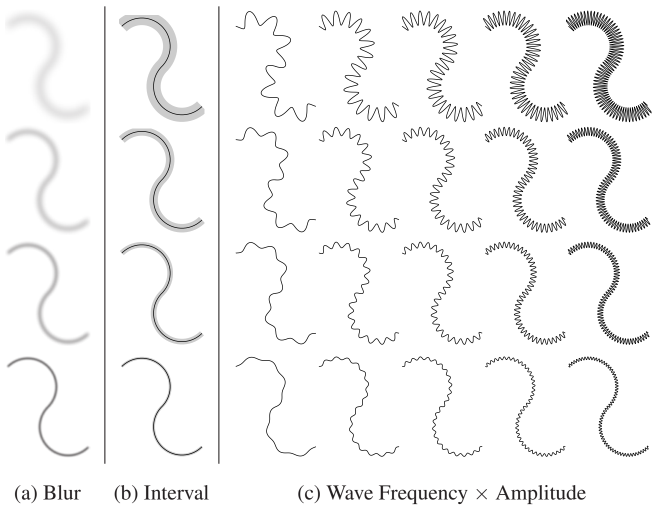


Fig. 7. Multiple levels of example visual variables for varying contour widths. Please note how frequency influences amplitude.

Our technique only requires black and white (cf. Figure 6) and offers a wide set of design choices with regards to visual variables. Usually, we desire equal saliency between certainty and uncertainty, with detection-like tasks being considered an exception. Based on work by MacEachren et al. [29], we start our discussion using opacity as baseline for uncertainty (Figure 6a). Because of the small line width, the difference between various nodes is barely visible. If contrast is an issue, experimenting with more clean and geometric visual variables for uncertainty is an obvious choice. With sketchiness being considered unprofessional [9], we went for clean waveforms on the contours. The first one is dash frequency (Figures 6b), resembling a rectangular signal, and the second one is wave frequency (Figures 6c), resembling a sinusoidal signal. As expected, both visual variables are easily readable, provide more perceivable levels and a better highlight. Despite the visual similarity to *dashing* and *sketchiness*, we refrain from judging intuitiveness based on related work, because the application is very different. Dash frequency seems to introduce high-frequent noise. Therefore, the frequency (and phase) of the dashes has to be selected carefully to avoid interferences between different lines of the same hierarchy and among siblings of sub-hierarchies.

To discuss saliency, we present a set of visual variables with varying contour width. We have implemented *fuzziness* [29] using blur to preserve color and mass of the dissolved lines. Blur (Figure 6d) is more readable than opacity and introduces much less noise and saliency than dashed lines or wave frequency. The levels of blur (Figure 7a) are difficult to distinguish, which is in line with the findings of Boukhelifa et al., who found that up to four levels of blur can be discerned [9]. Regarding intuitiveness, Correll and Gleicher [16] discuss a binning effect between certainty and uncertainty.

The next one is a representation that is inspired by error bars. To prevent confusion, we call this visual variable interval (Figure 6e). Regarding intuitiveness, we expect it to be very close to the well-known error bars. At first glance, smaller levels are more difficult to recognize whereas higher levels are easy to distinguish (Figure 7b). The last visual variable is wave amplitude at a fixed frequency (Figure 6f). Please note that there is a dependency between those two variables regarding perception, i.e., low frequencies are detrimental to distinguishability and high frequencies lead to a Moiré effect (Figure 7c). From the same figure, we have a hunch that perception of frequency could be curve-geometry depended. Nevertheless, sine waves with constant frequency and a variable amplitude seem to work well. We could only speculate that the amplitude fulfills its role as emphasis while frequency aids regarding quantitative coding. Studying their dependencies is left for future work. We suspect that differences in overall value (cf. Figure 6 and 7) shift saliency toward certainty or uncertainty, respectively. Therefore, if equal saliency is desired, we suggest counterbalancing based on value, e.g., integrating all pixels of each contour within a certain area and then compensating by adjusting the intensity.

8 Examples

This section aims at demonstrating the usefulness of our technique using exact data as well as uncertain data. In the following examples, we use color to differentiate between categories. In the FLARE data set, we colorize each group of children with the same color, whereas in the other datasets we colorize complete sub-hierarchies (children of root nodes) with the same color. Our prototype is implemented in C++, using *Box2D*[2] for the force-directed layout (8000 iterations) and *Cairo*[3] for vector graphics output. Table 1 provides a summary of the example datasets.

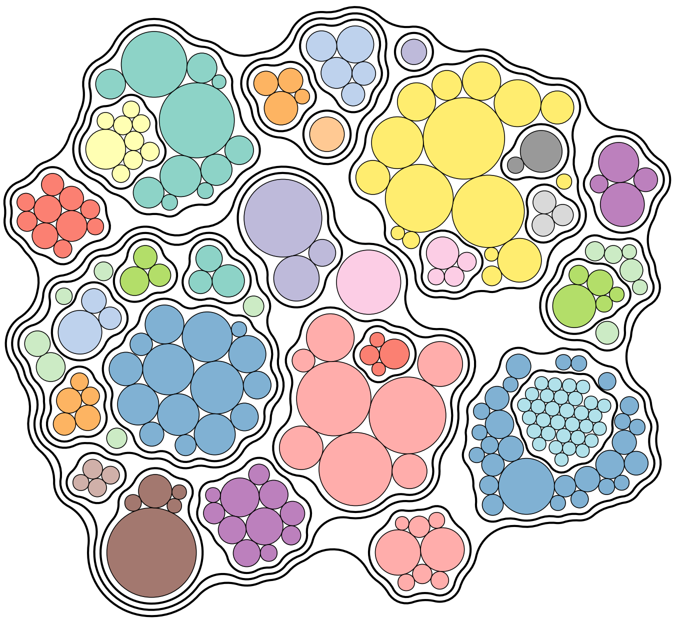
Table 1. Summary of example datasets. The runtime was measured on a desktop workstation equipped with an Intel i7-4770 CPU at 3.9 GHz.

dataset	nodes	leafs	max. depth	uncertain	time [s]
FLARE	252	220	4		3.2
S&P 500	639	503	3	✓	6.3
CES	403	295	6	✓	7.7

8.1 FLARE Package Structure

Figure 8a shows the structure of the FLARE data visualization software, which comes from the *UC Berkeley Visualization Lab*. The dataset was created by Jeff Heer and is part of the examples of *D3.js*[4]. The FLARE software consists of 10 modules that can contain further submodules. This example contains only certain data and aims to show the structural

[2]http://www.box2d.org/
[3]https://www.cairographics.org/
[4]https://bl.ocks.org/mbostock/4063582#flare.json


(a) Bubble Treemap

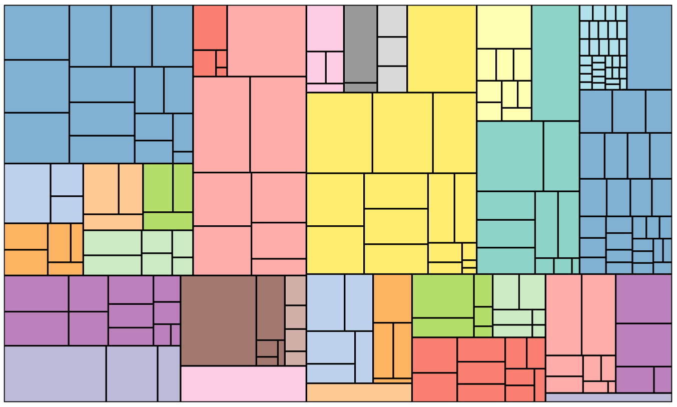

(b) Squarified Treemap

Fig. 8. Visualization of the package structure of the FLARE software: (a) our method, (b) result using Squarified Treemaps for comparison. The color coding is the same for both visualizations. However, Squarified Treemaps, in contrast to Bubble Treemaps, require color to avoid structural ambiguities.

properties of our approach. We map the size of each module to the area of the circles in the leafs. A Squarified Treemap of the same dataset [8] is shown in Figure 8b, slightly adapted to better fit our color palette. Please note that the colors of the Bubble Treemap are set to match the ones of the Squarified Treemap to allow a better comparison—which is still possible even though such non-uniform colors in sub-hierarchies impair the readability.

A particular problem of many traditional treemap approaches is that the hierarchy is hard to read and might even be ambiguous. Thus, treemaps are often colorized and use different shading styles, as shown in Figure 8b. Even though we can also use color to increase readability, it is not necessary for our method. In Figure 8a, we take the colors from Figure 8b and transfer them to our treemap. The resulting coloring is even a bit disadvantageous, since now different colors are placed in a sub-hierarchy, but our visualization remains readable due to the clear structuring of outlines and the additional space we allow for our visualization. Not being restricted by color means that we can use this strong visual cue for showing additional aspects of the data. This supports our claim that reserving some extra space offers advantages for visualizing hierarchies.

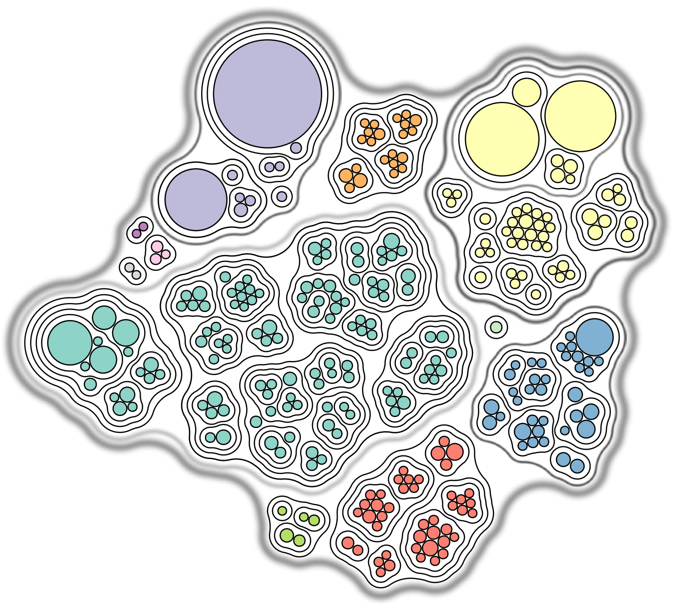


Fig. 9. Visualization of the data from the *Consumer Expenditure Survey*. The sizes of the leafs are proportional to the value of each item in the survey; the standard error is shown through the thickness of the contours. Also, contours with high uncertainty are blurred, to give the impression of uncertainty. The food category (cyan) and the housing category (yellow) have a high standard error and are therefore depicted with a stronger blur.

8.2 S&P 500 Index

Traditionally, financial data has been visualized using treemaps. Analysts are usually interested in the history of the stock in the form of a time series, since the current price of a stock alone does not give much information on how well the stock is performing. Our proposed method can be used to show the current value of a stock as well as supplemental information about its behavior over a given range of dates, providing additional context. In many cases, it is also of interest how well a sub-industry or a sector as a whole is performing. Figure 1 shows a visualization of the companies that are part of the *Standard & Poor's 500* index (S&P 500), grouped by sectors and sub-industries. For this example, we collected data using the *Yahoo Financial API*, for one week in November 2016. The size of the circles represent the mean closing prize of the stock for the given week. We use the contour to show the standard deviation σ of each stock. Even though stocks can depend on others, for our visualization purposes we assume that they behave independently. This allows us to use an uncertainty model as described in Section 4 to propagate σ toward the root.

Our visualization shows the stocks that were stable during the given period of time and others with larger variations. By looking at the waviness of the contours, it is relatively easy to identify the stock with the biggest changes, since the variance is reflected in all the contours of the respective sub-systems. In this case, the reason for the big changes were a 5-for-1 stock split, which led to single stock only having a fifth of the original value.

8.3 Consumer Expenditure Survey

The Consumer Expenditure Survey (CES) is an annual survey by the United States Department of Labor that measures the income, as well as the expenditures of *consumer units*, i.e., families or households in the US. The United States have about 109 million consumer units, out of which approximately 30,000 consumer units are sampled [6]. From these samples, the mean expenditures are estimated.

Even though the consumer units are chosen carefully to reflect the population, finding a perfect sample is impossible. The sampling error that is introduced through this method is measured using the *standard error* and gives information about the uncertainty with which the values

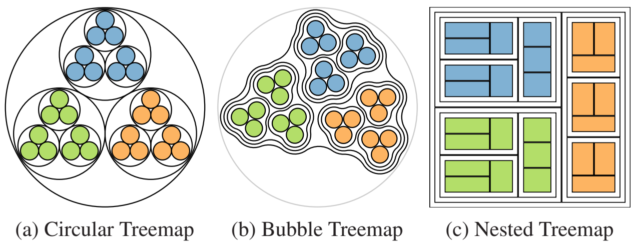


Fig. 10. Size comparison between circular treemaps (a), a corresponding Bubble Treemap (b), and a Nested Treemap (c) for a 3-ary tree where all leafs have the same value. When the branching factor of the tree increases, the difference becomes less pronounced between (a) and (b).

are afflicted. The survey uses *stratified random sampling* instead of simple random sampling. Because of this, the usual standard textbook formulas do not apply here. Blaha [6] describes the *replication methods* that were used to calculate the standard error, namely the *Balanced repeated replication*. The standard error would indicate the magnitude of the variability if the survey were to be repeated with different samples of consumer units [6].

We use our method to visualize the diary survey of the 2014 dataset[5], the result can be seen in Figure 9. We map the value of each item to the radius of the circles. For our visualization, we use a combination of thickness and blur to show the uncertainty. Each of these visual variables alone would suffer from deficiencies: Blur might get hard to read quickly, since the contour would become too light, whereas thickness alone would be perceived counter-intuitively (uncertain values would appear very thick). Blur can only be perceived correctly up to four levels [9], hence, we map the uncertainty to four levels of thickness and set the blur proportional to each level. Our visualization shows two categories that are afflicted with a high standard error: the *Food* category, shown in cyan, and the *Housing* category, colored in yellow. Within the housing category, the subcategory with the highest standard error is *Fuel and Utilities*.

9 Discussion and Limitations

In comparison to circular treemaps, we can use space more efficiently, especially for k-ary trees with a small branching factor. Figure 10 shows this problem for a 3-ary tree with four levels and leafs of equal size. In the most extreme case, namely, a binary tree where the left and right children have equal sizes r_c, the parent circle has to have a radius $r_p = 2r_c$. When the circles of the children are drawn on the inside of the parent, only 50% of the area of the parent circle is used. Rectangular treemaps are perfectly space-efficient, e.g., Squarified Treemaps (Figure 8b) and Nested Treemaps (Figure 10c). Please note that the leafs can appear unequal when the aspect ratio is not the same.

Arc Primitives The main visual characteristic of Bubble Treemaps is that the inner structure is defined by an arrangement of leaf nodes that is reflected on the outside by a contour. Nesting contours leads to parallel curves that capture the underlying tree structure. This effect would be difficult to achieve with energy-based contouring methods such as implicit curves or splines. Using circles and arcs as basic primitives of our method has several advantages: The users already have a good intuition of how to interpret Bubble Treemaps, since circle primitives are used throughout many different visualizations already. Furthermore, adding labels to circles and biarc curves should be simple and visually pleasing because of their clean geometry. For example, labels can be added by either allocating more space per node or, if suitable, using segments of the contours. Another compelling reason to build our method upon circles are the clear visual outlines that can be achieved, which leads to an engaging visualization. Other circle-based visualizations [22, 45] show that arrangements of circles are judged as aesthetic.

[5]`https://stats.bls.gov/cex/programs/r14.zip`

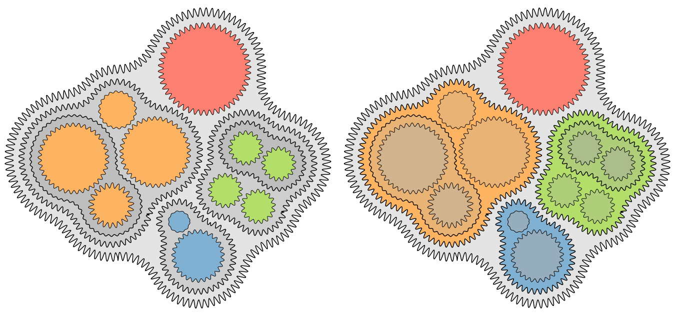


Fig. 11. Comparison of neutral shading (a) and colored (b) inner node areas. While filling contour areas allows us to encode additional information, it also introduces bias regarding area perception: In (a) the sum of the green leafs correctly appears smaller, while in (b) the sum of the green area erroneously appears bigger, than the red node.

Area Perception Like circular treemaps, Bubble Treemaps do not reflect the aggregate size of a sub-hierarchy in the inner node area of the enclosing contour. Filling the inner node areas with color would allow us to encode information, such as the topology of the hierarchy. There is, however, a risk of shifting saliency regarding aggregation of expected values. This effect is illustrated in Figure 11, where the topology of the tree is emphasized using different shades of gray (Figure 11a) and color (Figure 11b). Filling the area of each level in a neutral color improves the perception of the groups, as well as the depth of the hierarchy. In Figure 11b, each sub-hierarchy was assigned a color, and different levels of depth are emphasized by decreasing the saturation. This highlights the group structure of each node but introduces a bias in area perception: The aggregation of the green group now falsely appears larger than the single red node. The potential error amplifies with increasing contour thickness, which further aggravates the interpretation under the presence of uncertainty. Therefore, we advise against filling inner node areas if visual aggregation of leaf nodes is desired.

Hierarchy Perception In general, the efficiency of treemaps decreases with increasing tree depth, since the implicit representation tends then to hide the underlying structure. Bubble Treemaps share this characteristic to some degree: If deeper sub-hierarchies are placed toward the center of the visualization that are not directly adjacent to contours, including the root node, grasping depth becomes difficult. In all other cases, e.g., the green node to the very left in Figure 9, reading the depth is done by counting the number of contours on the outside of a group. As shown in Table 1, the datasets in this paper have a depth that ranges from three to six levels and up to about 500 nodes. For such hierarchies, our proposed method works well, but we expect that deeper hierarchies will pose greater challenges.

Depending on the chosen visual variables and their parameters, uncertain regions can appear more salient then certain regions. While this might be desired in some scenarios, i.e., when searching for categories with high uncertainty, in the general context of uncertainty visualization this might be confusing. One way to deal with this problem is to additionally adjust the opacity of these uncertain regions, giving the user better visual cues to interpret the visualization, while retaining the quantitative encoding. Regarding the encoding, we expect frequency and amplitude to behave similarly to gradient as a visual variable in that only a certain amount of levels can effectively be perceived. Our intuition is that a fine granular distinction should be possible (cf. Figure 7).

Computational Complexity and Runtime The computational complexity of constructing the contour strongly depends on how the smoothness s is set. If s is much greater than the biggest radius $max(r_i)$, this will lead to the degenerate case of the intersection graph where each circle intersects each other circle. To identify the successor of the

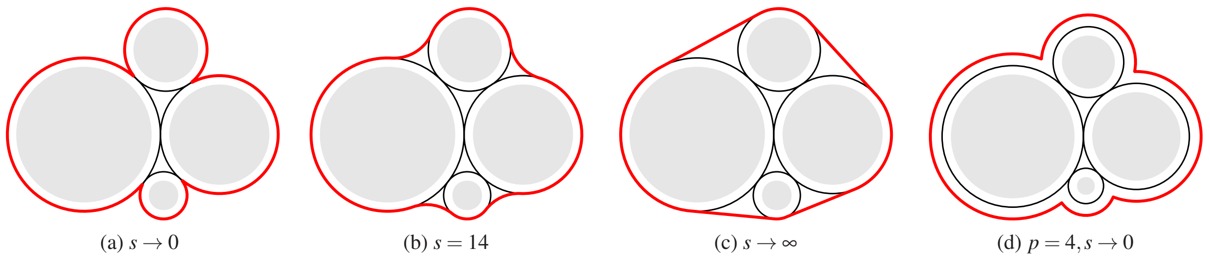


Fig. 12. Setting the smoothness factor s toward infinity yields the convex hull of the set of circles. The other parameters of the contour are constant. We can use the same construction to obtain the offset polygon of the circles by introducing an additional padding for each circle.

current circle, we need to find the circle with the leftmost intersection point. While traversing the intersection graph, we have to consider all other circles when searching for the next element. This leads to a computational complexity of $O(n^2)$.

The runtime of the layout step of our method is mainly bound by the setup of the simulation, especially how many iterations are needed to achieve a good layout. Therefore, one needs to find a good compromise between quality and runtime and predicting the exact number of iterations in advance is difficult. For the datasets that we show in this paper, we have found that 8000 iterations usually lead to good results.

Table 1 shows runtimes for different datasets, as well as additional information about each dataset. Even though the force-based simulation mainly determines the performance of our algorithm, there are several characteristics that influence the runtime of the computation. The overall number of nodes is the most obvious one, but since the physical simulation is performed for each sub-hierarchy, the maximum depth of the tree and the overall breadth of the tree are also important factors. This is reflected in the runtime of the CES dataset, which has fewer nodes than the S&P 500 dataset, but due to the higher maximum depth still takes longer to compute.

Parameters of the Contour The way we construct the contour around the tightly packed circles leads to a general notion of a contour. By adjusting parameters that define the contour, namely the smoothness s and the padding p, we can emulate different concepts of computational geometry. As described in Section 5, s controls the radius of the tangent arcs and can be used to steer how closely the contour will cling to the underlying circles. The effects of the parameters on the contour are also shown in Figure 12. When $s \to 0$, we obtain the concave hull of the set of circles (Figure 12a). Increasing s will relax the contour, therefore decreasing its total perimeter (Figure 12b). Finally, for $s \to \infty$, our algorithm computes the convex hull of the set of circles (Figure 12c). Independent of s, we can also adjust the padding p of the contour. This controls how far the contour will be offset from the original circles. By additionally setting $s \to 0$, we simulate the offset polygon (Figure 12d). Adjusting these parameters per level, s in particular, can be used to further improve readability. For the provided examples, we slightly reduced s with each level. This leads to contours that resemble isocontours, a representation that many users might already be familiar with from topographic maps or 2D contour plots.

We imagine that the generality of the description of the contour makes it possible to be used in other contexts of visualization as well. Lately, advances have been made in the field of visualizing set membership for objects embedded in the plane [13, 31]. Our method for constructing a contour around objects is not restricted to our use case but could be applied to draw contours around arbitrary objects that are embedded in the plane. For this to work, it should be possible to define a bounding circle for each object and to know the maximum distance that the elements of each cluster have, to choose a good value for the smoothness parameter. Bubble charts that visualize hierarchical or categorical data could also benefit from the proposed contour method.

10 Conclusion

We have presented Bubble Treemaps, a novel method that allows us to visualize hierarchical data afflicted with uncertainty. The main idea of our visualization is to deliberately allocate extra space that can be used to encode additional information. For this, we have presented a method to group circles together using contours based on circular arc splines. We have described a hierarchical force-based layout algorithm that we use to transform a circular treemap into a more compact representation. Since encoding uncertainty into visualizations is difficult, our method tries to leave as many design choices as possible. We show how several visual variables can be used to effectively convey uncertainty information, while still showing the original underlying structure.

In future work, we want to incorporate temporally changing data into our visualization. Since we use the circular treemap algorithm as an initialization for our method, to some degree, we inherit its properties, namely that the initial layout is not stable. After performing the force-based layout, however, our algorithm could be used for dynamic or interactive representation by simply repeating the force-based layout procedure, if the changes in the leafs are not too large. We are confident that the subsequent adjustment should allow user interaction. Furthermore, we are interested in incorporating appearing and disappearing hierarchies. For this, we imagine that the initialization could be changed to space-filling curves, which would create some robustness against changes. Visualizing changes in the topology is especially relevant for depicting structural uncertainty.

Acknowledgments

We would like to thank the German Research Foundation (DFG) for financial support within project A01 of SFB-TRR 161. We also thank Hendrik Strobelt for the fruitful discussions and active feedback.

References

[1] C. Aggarwal and P. Yu. A survey of uncertain data algorithms and applications. *IEEE Transactions on Knowledge and Data Engineering*, 21(5):609–623, 2009.
[2] G. Aisch and D. McCandless. Bubble tree (Open Knowledge Foundation). `http://okfnlabs.org/bubbletree/`, 2011. Last Accessed: 2017-03-26.
[3] D. Auber, C. Huet, A. Lambert, B. Renoust, A. Sallaberry, and A. Saulnier. GosperMap: Using a gosper curve for laying out hierarchical data. *IEEE Transactions on Visualization and Computer Graphics*, 19(11):1820–1832, 2013.
[4] M. Balzer, O. Deussen, and C. Lewerentz. Voronoi treemaps for the visualization of software metrics. In *Proceedings of the ACM Symposium on Software Visualization*, pages 165–172, 2005.
[5] J. Bertin. *Semiology of Graphics*. University of Wisconsin Press, 1983.
[6] J. L. Blaha. Standard errors in the Consumer Expenditure Survey. `https://www.bls.gov/cex/anthology/csxanth5.pdf`, 2003.
[7] K. Bolton. Biarc curves. *Computer-Aided Design*, 7(2):89–92, 1975.
[8] M. Bostock. Treemap. `https://bl.ocks.org/mbostock/4063582`, 03 2017. Last Accessed: 2017-03-29.
[9] N. Boukhelifa, A. Bezerianos, T. Isenberg, and J.-D. Fekete. Evaluating sketchiness as a visual variable for the depiction of qualitative uncertainty. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2769–2778, 2012.
[10] K. Brodlie, R. A. Osorio, and A. Lopes. A review of uncertainty in data visualization. In *Expanding the Frontiers of Visual Analytics and Visualization*, pages 81–109. 2012.

[11] M. Bruls, K. Huizing, and J. J. van Wijk. Squarified treemaps. In *Proceedings of the Joint EUROGRAPHICS and IEEE TCVG Symposium on Visualization*, pages 33–42, 2000.

[12] S. Carter. Four ways to slice Obama's 2013 budget proposal. https://www.nytimes.com/interactive/2012/02/13/us/politics/2013-budget-proposal-graphic.html, February 2012. Accessed: 2017-03-24.

[13] C. Collins, G. Penn, and S. Carpendale. Bubble sets: Revealing set relations with isocontours over existing visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1009–1016, 2009.

[14] C. R. Collins and K. Stephenson. A circle packing algorithm. *Computational Geometry*, 25(3):233–256, 2003.

[15] C. Correa, Y.-H. Chan, and K.-L. Ma. A framework for uncertainty-aware visual analytics. In *Proceedings of the IEEE Symposium on Visual Analytics Science and Technology*, pages 51–58, 2009.

[16] M. Correll and M. Gleicher. Error bars considered harmful: Exploring alternate encodings for mean and error. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2142–2151, 2014.

[17] K. Dinkla, M. J. van Kreveld, B. Speckmann, and M. A. Westenberg. Kelp diagrams: Point set membership visualization. *Computer Graphics Forum*, 31(3):875–884, 2012.

[18] FINVIZ.com. Stock market bubbles. http://finviz.com/bubbles.ashx, 2017. Last Accessed: 2017-03-24.

[19] T. Gschwandtner, M. Bögl, P. Federico, and S. Miksch. Visual encodings of temporal uncertainty: A comparative user study. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):539–548, 2016.

[20] H. Guo, J. Huang, and D. H. Laidlaw. Representing uncertainty in graph edges: An evaluation of paired visual variables. *IEEE Transactions on Visualization and Computer Graphics*, 21(10):1173–1186, 2015.

[21] D. Halperin and M. H. Overmars. Spheres, molecules, and hidden surface removal. *Computational Geometry*, 11(2):83–102, 1998.

[22] M. Hlawatsch, M. Burch, and D. Weiskopf. Bubble hierarchies. In *Proceedings of the Workshop on Computational Aesthetics*, pages 77–80, 2014.

[23] M. Hlawatsch, M. Burch, and D. Weiskopf. Visual analysis of eye movements by hierarchical filter wheels. In *Proceedings of the 19th International Conference on Information Visualisation*, pages 107–113, 2015.

[24] J. Hullman. Why evaluating uncertainty visualization is error prone. In *Proceedings of the 6th Workshop on Beyond Time and Errors on Novel Evaluation Methods for Visualization*, pages 143–151, 2016.

[25] B. Johnson and B. Shneiderman. Tree-maps: A space-filling approach to the visualization of hierarchical information structures. In *Proceedings of the 2nd Conference on Visualization*, pages 284–291, 1991.

[26] R. Khlebnikov, B. Kainz, M. Steinberger, and D. Schmalstieg. Noise-based volume rendering for the visualization of multivariate volumetric data. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2926–2935, 2013.

[27] B. Lee and F. Richards. The interpretation of protein structures: Estimation of static accessibility. *Journal of Molecular Biology*, 55(3):379–400, 1971.

[28] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, pages 163–169, 1987.

[29] A. M. MacEachren, R. E. Roth, J. O'Brien, B. Li, D. Swingley, and M. Gahegan. Visual semiotics & uncertainty visualization: An empirical study. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2496–2505, 2012.

[30] M. J. McGuffin and J.-M. Robert. Quantifying the space-efficiency of 2D graphical representations of trees. *Information Visualization*, 9(2):115–140, 2010.

[31] W. Meulemans, N. H. Riche, B. Speckmann, B. Alper, and T. Dwyer. KelpFusion: A hybrid set visualization technique. *IEEE Transactions on Visualization and Computer Graphics*, 19(11):1846–1858, 2013.

[32] L. Micallef and P. Rodgers. eulerAPE: Drawing area-proportional 3-venn diagrams using ellipses. *PLoS ONE*, 9(7):1–18, 07 2014.

[33] L. Micallef and P. Rodgers. eulerForce: Force-directed layout for euler diagrams. *Journal of Visual Languages & Computing*, 25(6):924–934, 2014.

[34] A. T. Pang, C. M. Wittenbrink, and S. K. Lodha. Approaches to uncertainty visualization. *The Visual Computer*, 13(8):370–390, 1997.

[35] N. H. Riche and T. Dwyer. Untangling Euler diagrams. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1090–1099, 2010.

[36] C. Schulz, A. Nocaj, J. Görtler, O. Deussen, U. Brandes, and D. Weiskopf. Probabilistic graph layout for uncertain network visualization. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):531–540, 2017.

[37] H. J. Schulz. Treevis.net: A tree visualization reference. *IEEE Computer Graphics and Applications*, 31(6):11–15, 2011.

[38] H. J. Schulz, S. Hadlak, and H. Schumann. The design space of implicit hierarchy visualization: A survey. *IEEE Transactions on Visualization and Computer Graphics*, 17(4):393–411, 2011.

[39] P. Simonetto, D. Archambault, and C. Scheidegger. A simple approach for boundary improvement of euler diagrams. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):678–687, 2016.

[40] K. Stephenson. *Introduction to circle packing: The theory of discrete analytic functions*. Cambridge University Press, 2005.

[41] C. Vehlow, T. Reinhardt, and D. Weiskopf. Visualizing fuzzy overlapping communities in networks. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2486–2495, 2013.

[42] F. Viégas, M. Wattenberg, J. Hebert, G. Borggaard, A. Cichowlas, J. Feinberg, J. Orwant, and C. Wren. Google+Ripples: A native visualization of information flow. In *Proceedings of the 22nd International Conference on World Wide Web*, pages 1389–1398, 2013.

[43] W. Wang, H. Wang, G. Dai, and H. Wang. Visualization of large hierarchical data by circle packing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 517–520, 2006.

[44] K. Wetzel. Pebbles — using circular treemaps to visualize disk usage. http://lip.sourceforge.net/ctreemap.html, 2003. Last Accessed: 2017-03-31.

[45] T. Wu, L. Zhang, and J. Yang. Automatic generation of aesthetic patterns with cloud model. In *Proceedings of the 12th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery*, pages 1077–1084, 2016.

[46] H. Zhao and L. Lu. Variational circular treemaps for interactive visualization of hierarchical data. In *Proceedings of the IEEE Pacific Visualization Symposium (PacificVis)*, pages 81–85, 2015.