

GosperMap: Using a Gosper Curve for Laying out Hierarchical Data

David Auber, Charles Huet, Antoine Lambert, Benjamin Renoust, Arnaud Sallaberry, Agnes Saulnier

Abstract—The emergence of very large taxonomies that result from the increase in available data raises many problems of visualization and navigation through their corresponding hierarchies. On datasets of such scale, classical graph drawing methods do not take advantage of certain human cognitive skills such as shape recognition. These cognitive skills could make it easier to remember the global structure of the data. In this paper, we propose a method that is based on the use of nested irregular shapes. We name it GosperMap as we rely on the use of a Gosper Curve to generate these shapes. By employing human perception mechanisms that were developed by handling, for example, cartographic maps, this technique facilitates the visualization and navigation of a taxonomy. An algorithm has been designed to preserve region containment according to the taxonomy and to set the leaves' sizes proportionally to a property, in such a way that the size of non-leaf regions corresponds to the sum of their children's sizes. Moreover, the input ordering of the hierarchy's nodes is preserved, i.e., the areas that represent two consecutive children of a node in the hierarchy are adjacent to one another. This property is especially useful because it guarantees some stability in our algorithm. We illustrate our technique by providing visualization examples of the repartition of tax money in the U.S. over time. Furthermore, we validated the use of the GosperMap in a professional documentation context. and show the stability and ease of memorization for this type of map.

Index Terms—Treemap, Gosper curve, Hierarchical data visualization.



1 INTRODUCTION

The wealth of data that is available requires us to use efficient techniques to access the information that it holds. The manner in which files are stored on a computer is a prime example of how a good classification, in this case, the folder hierarchy, helps one to locate what one is looking for. Many other examples, such as the Dewey decimal book classification, lead us to believe that hierarchical data classification is the most efficient and intuitive way to organize knowledge.

Many methods have already been proposed for visualizing hierarchies, the most prominent examples of which are tree representations (see Treevis.net for an overview). Reingold and Tilfort [?] presented the first approach to visualizing through trees in a node link diagram, positioning each node over its children, which can also produce a dendrogram. Since then, more representations have been proposed, such as a radial layout [?] or a bubble tree algorithm [?]. Node link diagrams lack the ability to easily map the nodes' size and also have the drawback of poorly filling the space between nodes.

Another visual metaphor is to visualize regions, either with containment (i.e., children are enclosed by their parents), such as Treemaps [?], or without containment, including icicle plots [?]. Icicle plots have a similar drawback to node link diagrams because they tend to leave a large amount of unused space. Furthermore, as demonstrated in [?] treemaps are not an intuitive representation because the hierarchical structure is not as clear as in a conventional tree drawing. Here, we propose a novel visual metaphor that attempts to overcome these shortcomings.

The cartographic map metaphor has been in use for a while because it is a very intuitive way to represent information. Many examples have been demonstrated to use this metaphor, such as the *temperance map* designed by W.M. Murrell [?] in 1846, the *European Economic and Social Committee map* ¹ and a recent XKCD cartoon ² in a lighter tone.

This background has led us to propose a method that is based on the use of irregular shapes. Our approach is motivated by Fabrikant and Skupin [?], [?], they claim that such an approach would appeal to the cognitive skills that have been developed by anyone who has ever read a map. These skills include the recognition of region containment (e.g., North America contains the United States, which is divided into states that contain counties; Europe contains Germany, which is composed of 16 Länder) and the pattern recognition of regions areas (e.g., a country that is shaped like a boot is

-
- David Auber and Antoine Lambert are with the Université Bordeaux 1, the CNRS UMR 5800 LaBRI and the INRIA Bordeaux Sud-Ouest, France, E-mail: david.auber@labri.fr, antoine.lambert@labri.fr.
 - Charles Huet is with the CNRS UMR 5800 LaBRI and the INRIA Bordeaux Sud-Ouest, France, E-mail: charles.huet@labri.fr.
 - Benjamin Renoust is with the Université Bordeaux 1, the CNRS UMR 5800 LaBRI, the INRIA Bordeaux Sud-Ouest and the Institut National de l'Audiovisuel, France, E-mail: brenoust@ina.fr.
 - Arnaud Sallaberry is with the University of California at Davis and Pikko, USA, E-mail: asallaberry@ucdavis.edu.
 - Agnes Saulnier is with the Institut National de l'Audiovisuel, France, E-mail: asaulnier@ina.fr.

1. <http://www.eesc.europa.eu/?i=portal.en.self-and-co-regulation-cartography>
2. <http://xkcd.com/802/>

easily remembered). Such visualization also allows the use of all the cartography semiotics [?]. Additionally, we want our visualization to explicitly make known some properties of the tree. To achieve these goals, a region must have an area that is proportional to the sum of the area of its children. There is also a need to visualize the evolution of taxonomies, and the stability of the layout is a prevalent feature when visualizing dynamic trees. Finally, a significant issue with respect to a visualization is its aesthetic aspect. Indeed, studies in ergonomics have underlined the interesting influence of aesthetics on usability and adoption [?], [?], [?]; thus, specific attention must be devoted to creating pleasant shapes and attractive visualizations when addressing a large audience or readership.

The contribution of this paper arises in three forms. First we propose an algorithm for creating treemap-like representations of trees that have irregular shapes. This solution is rapid and can be compared to the first treemap algorithm [?], although no space is devoted to displaying labels. The second contribution of our paper is to propose an improvement of the first algorithm to separate the boundaries of nested regions. This algorithm helps to visualize the hierarchy, and the boundaries can be used to display labels. The last main contribution of this paper is to propose an algorithm to display labels in concave shapes.

2 BACKGROUND AND RELATED WORKS

We first focus on works that are related to the cartographic map metaphor, and we demonstrate that the existing methods do not fulfill all of the previously specified requirements. We then focus on the visualization of trees as regions, and present their properties.

2.1 Cartographic Maps Metaphor

Geographical maps as an input: Redrawing geographical maps with additional constraints has already been well studied in the literature. For example, distortion techniques have been used to highlight a focused entity while preserving the context [?]. Cartograms represent geographic maps in which the sizes of the regions depend on a given value, where regions are deformed to the desired sizes and adjacency can be preserved [?], [?], or regions are represented as rectangles [?], [?] and adjacency is lost. These methods require pre-existing spatial data, making them irrelevant to our problem. Rooke *et al.* [?] proposed an example of abstract data visualization based on real geographical maps. These authors used a map of the world to visualize a hierarchy of commands.

From Self-Organizing Maps to Voronoi Diagrams: *Self-organizing maps* (SOM) [?] are based on an unsupervised learning algorithm that produces a two-dimensional map in which similar objects are close to each other. Skupin [?] proposed a technique based on SOMs, Voronoi diagrams, and clustering that produces

images that look like country maps from texts; in this technique, how related the maps are is based on the common concepts that they contain. We are unaware of a method that could transform the input of our method to the format that these methods use. Gansner *et al.* [?] displayed graphs as geographic-like maps, using a method based on graph drawing, Voronoi diagrams, and clustering. Our data could be transformed to use this method; however, this approach does not guarantee that the regions will be connected.

Representing data as landscapes: Another approach to generating geographic-like maps is to display the data as landscapes. *Themescapes* [?] are abstract, three-dimensional landscapes of information. GraphSplatting [?] is a related technique that transforms a graph into a two-dimensional scalar field. The scalar field is rendered by a color-coded map, a height dimension or a set of contours. While these methods produce nice landscape-like maps, they cannot be adapted to trees without creating disconnected regions.

Addressing fractals: Generating virtual maps based on fractal models is often used in movies and computer games [?]. Keim *et al.* [?] first proposed a method based on the use of recursive patterns in the field of Information Visualization. In their work, the elements to be visualized are mapped to small identical patterns that fill a plane. The first visualization based on fractals was developed 10 years later, as a result of the preliminary work of Wattenberg [?], in which he indicated the idea of using space-filling curves. Muelder and Ma [?] presented a method that is based on this idea. These authors first hierarchically clustered the nodes of a graph and then extracted an ordering of the graph nodes according to a planar layout of the tree of clusters. Finally, they placed the nodes according to the ordering on a space-filling curve such that the gap between two nodes corresponds to their proximity in the hierarchy. Two consecutive nodes of the ordering will be closer to each other if they belong to the same cluster. Because this method is devoted to very large graph layouts, the authors are not interested in revealing the underlying clustering and are not interested in making the visualization appear like a map.

2.2 Trees visualized as regions

The visualization of trees as regions has been thoroughly discussed in the literature, and many variations on the subject, all of which bring different highlights, exist. For a detailed comparison, please refer to previous work [?]. In this paper, we will consider two main categories.

2.2.1 Adjacent regions

The earliest method of adjacent regions is the Icicle plot [?], which is constructed by placing the children next to the parent such that the depth and siblings are mapped on the X- and Y-axis, respectively. Alternatively, the nodes can have radial coordinates, whereby the

angle and depth are mapped on the siblings, on the distance, respectively. This method was introduced as “information slices” [?]. Sunburst [?] is a variation of this method that uses the whole circle and displays a deeper hierarchy in a single image. However, a node’s area is not proportional to the sum of the areas of its children.

The benefits of these methods are that labels can be placed inside each region without overlapping with the children regions, and that the depths of the nodes are easily identifiable.

2.2.2 Region containment

The most common form of region containment visualization is treemaps, which recursively divide the plane by walking the tree from the top to the bottom. A detailed overview of treemaps by Ben Shneiderman, updated by Catherine Plaisant, can be found here³. The first treemap method, “slice and dice” [?] is constructed by recursively dividing the plane alternatively horizontally and vertically. As a result rectangles that have a high aspect ratio (*width/height*) can exist. Some methods have improved on the original treemaps, such as the squarified treemap [?], which has an aspect ratio that is very close to 1, but does not account for the order of the children, which would make the layout more stable over time. Strip layout [?] is a compromise between the squarified and the slice and dice; the aspect ratio is not as close to 1 as with the squarified treemap but the order of the nodes is preserved. Quantum treemaps [?] uses rectangle widths and heights that are multiples of the same fixed number, which eases the size comparisons. The mixed treemap [?] uses the slice and dice for the upper level and the squarified treemaps for the lower levels. Voronoi treemaps [?], [?], [?] recursively split the space similar to the aforementioned methods and use a Voronoi tessellation to create convex polygons instead of rectangles. BubbleMaps also [?] produce non rectangular shapes by positioning adjacent squares to represent leaves having the same parent. The City layout [?] uses a street map metaphor, in which leaves are represented as buildings, located in districts, which are contained in larger districts, up to the root of the tree.

Finally, two works address the problem of transforming maps into Treemaps [?], [?]. Even if it is not directly useful for our purposes, we think that this approach is nevertheless related to tree visualizations and cartographic maps.

2.2.3 Properties

While considering the many properties of layouts that impact the visibility of a tree, we chose the layouts that pertain to region-based maps : **Region Containment**, i.e., whether or not children are placed inside their parents; **Aspect Ratio** of the smallest rectangle that encloses the region, i.e., the *width/height*, where 1 is the ideal value, to afford an easier area comparison; **Area Correlation**,

	Region Contmt.	Aspect Ratio	Area Correl.	Stab.
Icicle plots			X	X
Info. Slices			X	X
Sunburst				X
City Layout	X			/
Slice & Dice TM	X		X	X
Squarified TM	X	X	X	
Strip TM	X	/	X	/
Quantum TM	X	/	X	/
Mixed layt. TM	X	/	X	/
Voronoi TM	X	X	X	X

TABLE 1

Summary of the properties for the methods that aim to visualize trees as regions. An X indicated that the property is fully respected, a forward slash represents that the property is partially respected, and an empty cell implies the property is not respected. TM represents treemap.

i.e., how correlated the area of the leaves is to a given property, and the non-leaves are correlated to the sum of the value of their children; and **Stability**, i.e., how much a node moves on consecutive visualizations.

Table 1 sums up these properties for the aforementioned methods. This table is only given as an indication, and according to the tasks to be performed, the optimal design is not necessarily the design that maintains all of these properties. Moreover, other useful properties have not been mentioned. In this paper, our purpose is to focus on the properties mentioned, and our technique guarantees all of them.

As far as we know, no labeling technique can be applied to the methods that are displayed in Table 1 without some loss of one of these properties. Our technique offers to at least partially fulfill these properties and, in addition, to supply an efficient labeling method.

3 DRAWING A GOSPERMAP

Fractals are curves or shapes that have self-similar patterns. Intuitively, they can be split into parts such that each part has a pattern that is similar to the other parts and to the first part. Thus, the parts can be used to visualize categories, sub-categories and so on, while providing boundaries that resemble regions that are found in cartographic maps (a part of the Brittany coast resembles the whole Brittany coast [?]). The main purpose is to find how we can fit the region containment constraint into a fractal model. We propose a solution that is based on 2D space-filling curves, which are often used to generate fractals. Looking forward at the generation of these curves shows us that the process is composed of recursively subdividing a straight line segment.

To use a 2D space-filling curve, we must find an order for our taxonomy’s leaves. This order must be such that all of the nodes in a category are consecutive. This structure allows us to represent internal nodes by

3. <http://www.cs.umd.edu/hcil/treemap-history/index.shtml>

containment. This property has already been discussed [?], [?]. The entire process is detailed in Figure 1.

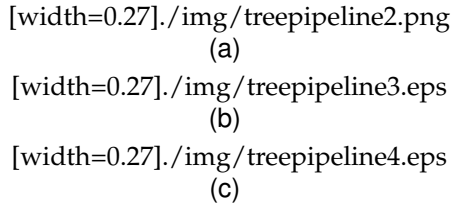


Fig. 1. Algorithm overview: (a) We find a linear arrangement on the leaves (a string of red nodes), using the structure of the tree. (b) We project this linear arrangement with a 2D space-filling curve. (c) For each internal node, we create a region that contains all of its descendants.

3.1 Curve selection

Our algorithm is closely tied to the curve because it must produce a result that resembles a map of nested irregular shapes while preserving the tree structure. The curve must be simple (i.e., without crossings) for each region to contain its children. To not restrict the amount of data that we can handle as input, a 2D space-filling curve that can expand indefinitely is required. The curve can be defined in a more formal way as a function $h : N \rightarrow R^2$; such that the Euclidean distance between $h(i)$ and $h(i+1)$ is 1. According to this function, we can define the *Worst-Case Locality* [?] value as follows:

$$WL = \lim_{k \rightarrow \infty} \sup_{i, j \in N_k} \frac{d(h(i), h(j))^2}{|i - j|}$$

where N_k denotes the set of k -digit base- n numbers and $d(h(i), h(j))$ is the Euclidean distance in R^2 . We square this term because, if the maximal distance in N is $O(k^2)$, then the maximal distance in R^2 is $O(k)$. WL indicates how much the nodes that are close in 1D space are close in 2D space. If this value is finite, then this curve has the *locality property* that is defined by: $d(h(i), h(j)) < c|i - j|^{1/2}$ where $c = WL^{1/2}$.

This property guarantees that children of a single node will be close to one another. With a curve that does not hold the locality property, two children of the same node can be far apart from each other (e.g. in the case of a spiral, up to its diameter).

We have analyzed many curves, and the curve that appears to be the most suitable is Gosper's flowsnake curve, which is also known as the Gosper island [?]. Indeed, this approach divides the plane into hexagons. Then, angles of the polygons that define the boundaries of the regions are $2\pi/3$ or $4\pi/3$. Obtuse angles make the boundaries smoother, which allows us to obtain regions that are similar to those that can be observed in cartographic maps. Gosper's flowsnake curve has the locality property, with $c = \sqrt{6.35}$ [?].

3.2 Positioning the leaves and dividing the space

The leaves' order is extracted by a depth-first search, which can be seen in Figure 1.b. We place the points along Gosper's curve, as shown in Figure 1.c. Each curve requires a specific way of dividing the plane, to make each node represented by a polygon; using a Voronoi diagram to perform this operation makes it generic, although it is slightly time-consuming ($o(n \cdot \log(n))$). In this specific case, the regions produced are hexagons, as can be seen in ??a and, thus, can be generated in $o(n)$. The size of a region that represents a node can be augmented by attributing several consecutive points of the curve to this node.

We are now able to create a region for each node of the hierarchy from the bottom to the top of the tree; a region of a node is obtained by merging the regions of its children, as shown in Figure ??. Notice that this method follows the region containment principle because Gosper's curve is planar.

Using the aforementioned method already produces treemaps with irregular shapes such as in Figure 3 in a very short time (less than one second for the graph shown, of over 6,000 nodes). The parent regions are drawn over their children, and are applied with some transparency so that the children are visible. This process does not allow the display of more than 3 levels, because the borders are overlapping. If a leaf is the only child of its parent, then their boundaries will overlap, hiding the depth of the leaf. To show more levels, we could hide all the current top-level regions, but we would lose the ability to distinguish two regions of the same depth in the tree that do not have the same parent. Another solution would be to drill down in a specific region, showing only the subtree of this region, which would lose the contextual information. We added a post-process to our method, which solves this problem (see section 4.1).

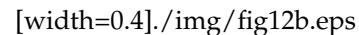


Fig. 3. Filesystem hierarchy of the Tulip (<http://www.tulip-software.org>) open-source software [?], on which we have implemented our prototype. The levels display changing boundary sizes according to the depth of the corresponding nodes in the hierarchy and provide a transparency value to the regions.

4 IMPROVING THE MAP

Differentiation and identification of the hierarchical levels in the map are required for the map to be visually analyzed. The boundary computations and the labeling can benefit from the map construction operations.

4.1 Boundary computations

An efficient way to detect the region containment for each level of the hierarchy could consist of separating

the lines that delimit the regions from one another. As we can see in Figure 1.c, the boundary that defines the region $\{6, 7, 8\}$ does not overlap its parent's boundary (region $\{5, 6, 7, 8\}$). As a result, it is easy to determinate a region's hierarchy because a region is contained inside of its parent regions (i.e., nodes in the path from this node to the root of the tree). Similar to in a cartographic map, we can see such boundaries as contour lines that demarcate regions that have the same altitude (i.e., the same level in the hierarchy).

We use the hexagonal grid that was created by the Voronoi diagram, and we replace each vertex of each hexagon by a node. Then, we add edges between the leaf and each of the vertices of its grid cell, effectively creating an internally triangulated graph, as in Figure 4.a.

[width=0.44]./img/hexagons6.eps
(a)

[width=0.44]./img/hexagons9-nontriangulated.eps
(b)

[width=0.44]./img/hexagons72.eps
(c)

[width=0.44]./img/hexagons9.eps
(d)

Fig. 4. (a) Vertices of the hexagon generated by the Voronoi diagram are replaced by a new node, which is linked to the leaf, effectively creating an internally triangulated graph. (b) The boundary of the purple and blue nodes, as computed by the first step of the boundary computation and other boundaries recursively computed. (c) Corresponding boundaries; the spacing between the boundaries decreases exponentially according to the number of nested boundaries. (d) Triangulated graph constructed to apply our boundary expansion method.

Next, we add boundaries to the regions, proceeding from the bottom to the top of the tree. In our example, we first create the boundary of the purple node. We process the edges that are crossed by the boundary clockwise, and each of these edges is replaced by a node that is positioned at the middle of the edge. These new nodes are linked to the target and the source of the removed edge by two new edges and to the following and preceding new nodes, to create the purple boundary that we see in Figure 4.b. We recursively apply this process to create all of the boundaries. In contrast to the method described in 3.2, the boundaries that are generated do not overlap with one another.

We could directly use these new boundaries to display our map. However, our technique consists of recursively splitting the edges of the triangulated graph, which makes the distance between two boundaries decrease exponentially as we create more boundaries. Figure 4.c highlights this phenomenon, in which the distance between the yellow boundary and the green boundary is

two-times smaller than the distance between the green boundary and the purple boundary. A straightforward way to solve this problem would be to simply divide the space between the borders by the number of layers. We opted for a more complex solution that separates the borders clearly and smoothes the borders to produce and enhance the aesthetics of our map. This method also possesses the advantage of maintaining a better area correlation for deeply nested nodes. This goal is accomplished by first transforming the graph shown in Figure 4.b into the internally triangulated (only the outer face is not a triangle) graph shown in Figure 4.d.

We then apply Tutte's algorithm [?] to find the node positions, such as the distance between consecutive boundaries, which is approximately the same. This step is to iteratively position every node that is not on the outer face at the barycenter of its neighbors. Each of these iterations runs in linear time. According to Tutte, this algorithm converges to a stable planar configuration in at least a linear number of iterations. Our graph is triangulated, and we fix the outer face of our graph. Thus, the solution is unique and preserves the initial embedding of our graph, which guarantees the region containment property. In our case, the graph is already planar, and the nodes are very close to their final positions, which considerably shortens the running time of this step. Figure 5.a shows the graph that is positioned by using this method, and Figure 5.b shows the same graph, but the nodes and edges that are not in the boundaries are removed. Notably, the barycenters can be weighted to allocate a greater area to the boundaries of the regions that represent the upper nodes of the hierarchy.

[width=0.44]./img/hexagons9-bary.eps
(a)

[width=0.44]./img/hexagons9-bary-clean.eps
(b)

Fig. 5. (a) Applying Tutte's algorithm to Figure 4.d enables us to transform the initial triangulated graph that was obtained by our method. This action solves the exponential problem on our boundaries, as shown in Figure 4.c, and additionally smoothes the boundaries of the regions, creating a more visually appealing visualization. (b) The results obtained after removing all of the dummy elements.

At the end of the boundary computations, we obtain a smooth, non-overlapping boundary for each internal node of the tree. Furthermore, for each internal node, we also obtain a list of triangles that define a closed region (border region) in which nothing is displayed. The node boundary is used to draw a filled polygon, and the border region is utilized to draw a textured contour that enhances the region containment visualization. Figure 6 shows the entire process on a file system of 200 nodes extracted from the VisWeek 2011 electronic

proceedings. To enable visualization of the size of the files as well as the size of the directories, we add a number of dummy children to each leaf, for which the number is proportional to the size (i.e., the file size in that example). The dummy nodes are then removed. Figure ?? shows an example of the final maps using the Python standard library directory on a Linux system as the input hierarchy.

Now that we have found a correct embedding of the nodes, we must display the labels to direct the user's navigation. Displaying all of the labels induces occlusion problems. The next section is devoted to an efficient method that is adapted to the layout.

4.2 Labeling

We will first give a method that shows the labels of all of the nodes at a certain depth, which provides a good overview of the data. Then, we will demonstrate a method that repeats a region's label along its border, allowing us to identify regions even when they do not appear in full.

4.2.1 Labeling regions of a given depth

This method aims to enhance the map by displaying labels that are inside of the regions. To achieve this goal, we display the label of every node at a given depth.

Displaying rectangular labels over the regions without accounting for their shapes was not efficient. Indeed, because of the concavity of many of these regions, some labels were not on the corresponding areas. Thus, we only use this method for the last level of the hierarchy (see Figure ??b). We then experimented with rectangular labels that were inscribed in the regions and aligned along the horizontal or vertical axis (see Figure 3). However, the irregularity of the regions causes the rectangle's size to be very small, whereas spreading the label over the region makes them larger and more readable.

We attempted to find a method that would generate labels that follow the region, such as the labels that Skupin presented [?]. The main idea is to locate a path that is included in a region on which it would be possible to draw text. This technique's principle is similar to the interaction of path text tools that are present in modern imaging software.

The regions in which we want to display labels are possibly concave polygons without holes. For each polygon, we compute a Voronoi diagram of its vertices, and then, we remove the vertices of the Voronoi diagram that are not included in the polygon. We then reconnect the Voronoi vertices to the closest polygon's vertices (i.e., the sources of their adjacent cells in the Voronoi diagram). That operation creates a tree structure that is close to a medial axis of our polygon [?]. Each path of that tree can be used to display a text that is at an equal distance from each of the polygon's borders. Let n be the number of vertices in a simple polygon; then, the complexity for the tree construction is $O(n \cdot \log(n))$.

The challenge is to select one path in the set of possible paths in the tree that we have constructed. We first attempt to select the longest path, as shown in Figure 4.2.1. This technique has the drawback of curving the labels a large amount, which hinders the readability. This effect results from the number inflection points (see "clustering" in Figure 4.2.1). Furthermore, differences in size along the medial axis makes choosing a proper font size difficult (as shown in Figure 4.2.1 on "import" and "distance"). The evaluation that we obtained in a professional environment (see section 5.2) using that type of labeling demonstrates that a tradeoff between the axis-aligned labels and the curved labels is necessary.

The evaluation indicates that, for the users, when it comes to the labels, readability takes priority over size. To attempt to better satisfy this constraint, we devised a new approach, that uses the longest path in the tree that we constructed. We selected the longest subpath that has a similar thickness (i.e., the distance to the border of the polygon) all the way and that is mostly straight. We find this path in $O(n)$ [?], and we select the thickness with a threshold, which can also be accomplished in $O(n)$. Future work on the labeling includes curve smoothing and size homogenization (the size can be constrained by the depth of the node to which it is attached).

[width=0.9]./img/labelsinmap_new.eps

Fig. 9. Labels on the children of the *plugins* region, which are the third level of the hierarchy. The box highlights the labels on the border of the nested regions.

4.2.2 Labeling of the whole hierarchy

The second labeling method helps the user to maintain the contextual information while he is zoomed in. This method is to apply textures that contain the labels along the regions' boundaries, in a similar fashion as the contour lines are applied on the cartographic maps. An example of this technique is highlighted in Figure 4.2.1. The leaf *Makefile* is contained in the *plugins* region, which itself is contained in an *arena*, child of the root node *ALEX*. *PluginSample* is clearly on the same level as *plugins*, and *CVS* is on the same level as *Makefile*.

5 EVALUATION

To evaluate our GosperMap (G.M.), we performed a scientific evaluation benchmark and a user evaluation in a professional environment. The objectives of these two evaluations are first to provide a quantitative evaluation according to a set of measures and then to evaluate the accuracy of the GosperMap for solving hierarchy analysis tasks.

5.1 Numerical Evaluation

We measured several parameters on the "What We Pay For" dynamic (time-stamped) dataset. Furthermore,

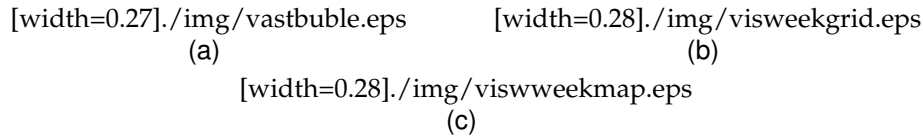


Fig. 6. Examples of the embedding process: (a) Initial tree; (b) Triangulated graph embedded using Tutte's algorithm; and (c) Final map with a texture applied on the paths that correspond to the region boundaries.

we compared the results that were obtained by our technique with two of the most popular methods. We chose the Sunburst (S.B.) [?] for its stability and the Squarified Treemap (S.T.) [?] for its good aspect ratio. Unfortunately, we were not able to compute these measures on Voronoi treemaps [?], [?], but according to the results of the original paper [?] and those of the fast Voronoi treemap algorithm [?], Voronoi treemaps should be at least as good as our technique. For each technique, we measured the computation time, the aspect ratio of each generated region, the correlation between the value mapped on the elements and the on-screen area, and the displacement of the elements between two consecutive states. Figure 10 gives a sample of the results that we obtained on the evolution of that weighted hierarchy for the years 2007, 2008, 2009, and 2010. Table 2 summarizes all of the measures that we computed on these different visualization techniques.

Benchmarking all of the techniques for the visualization of hierarchical data [?], [?], [?], [?], [?], [?], [?], [?] is outside the scope of this article. This evaluation is given only as an indication that our technique possesses the properties that make the treemaps efficient when visualizing the hierarchy.

[width=0.98]./img/resultstable.eps

TABLE 2

(time) Computation times in seconds, (aspect ratio) $width/height$ ratio of the generated regions, (area) the correlation between the value mapped on the elements and the area used on the screen, and (stability) the standardized displacement of elements between two states. The standard deviations are noted in red

Dataset

The data comes from <http://www.whatwepayfor.com>, and represents how much of a taxpayer's money goes to which accounts. The data used a yearly income of \$50,000 for a single person. We decided to visualize the evolution of the income/outcome according to the "function", "subfunction" and account classification. Initially the "What We Pay For" dataset is a directed acyclic graph. To visualize it we have duplicated leaf nodes ("accounts") that are connected to several internal nodes ("subfunction"). The resulting tree contains 990 nodes. To visualize all of the accounts, whether they are positive or negative, we use the absolute value of the "mycost" parameter.

Measures

Timing: In Table 2 one can see that the main drawback of our technique is its computational cost. S.B. and S.T. only need to sort elements, contributing to their speed and ease of implementation. Because of the generation of a Voronoi diagram and for our boundary computation phase, our technique requires working with a triangulated graph that is significantly larger than the input tree. Thus even if the algorithms that we use are reasonably fast (Voronoi, barycenter, polygon merging), the size of our triangulated graph is the bottleneck. For example, in Figure 10 we generated a triangulated graph of 33,059 nodes and 88,846 edges to obtain the final visualization. If we only use the thickness of the boundaries (see Figure 3) to visualize our dataset, then we spare ourselves the cost of generating the triangulated graph. Without the boundaries computation, our running time has the same order of magnitude as the other techniques.

Aspect Ratio: Table 2 summarizes the aspect ratio that is obtained for each visualization. We define the aspect ratio as the minimum of the width and height of the smallest rectangle that encloses a region divided by the maximum of the width and height of that rectangle which results in a value between 0 and 1. The aspect ratio of a visualization is the average of all the aspect ratios of regions that are used to represent the internal nodes and leaves. Our measures show that the G.M. surprisingly outperforms the S.T. in terms of the aspect ratio. Looking carefully at the S.T. results, one can see that the aspect ratio could be far from the expected ratio, when the size of the elements varies significantly. The G.M. uses a Gosper curve, for which the fractal nature induces an aspect ratio that is almost constant independently from the length of the curve or the position of the subpart. Curves that are smaller than the kernel (≤ 8 elements for a Gosper's curve at order 1) do not respect this property. The irregularity of the concave regions makes it more difficult to accurately compare areas, even though the aspect ratio is better.

Area correlation: Table 2 summarizes the correlation between the area of the regions and the value that is assigned to a node in the original tree. For each visualization we have computed the exact area of each polygon that is induced by a region boundary. As can be seen, the three methods have a significant correlation between the desired area and the measured area. The S.B.'s low score results from the fact that the algorithm ensures a correlation of 1 with the size of the angular sectors, if we had used a variable radius for each level of

[width=0.99]./img/casestudy2.eps

Fig. 10. Results for the “What We Pay For” dataset. **GosperMap** : For each year, the surface of each region is correlated (0.99) with the “mycost” value of the dataset. Observing the “Commerce and Housing Credit” region, even if the “mycost” parameter evolves significantly year after year, this evolution is easy to follow. Furthermore, new elements are always inserted at the bottom left corner. **Squarified Treemap** : For each year, the surface of each region is correlated (0.97) with the “mycost” value of the dataset. Observing the “Commerce and Housing Credit” region, because of the ordering that is used by the algorithm, the “function” region significantly moves. Nodes with a higher weight are easy to find because of the Squarified Treemap ordering. **Sunburst** : For each year, the surface of each region is correlated (0.92) with the “mycost” value of the data set. The evolution of the “Commerce and Housing Credit” region is easy to follow because of the high stability of the technique. However because the surface that is used for displaying one ring is constant, the addition of new elements (or small elements) is not always perceptible. This phenomenon can lead to evolution being unnoticed.

the S.B., as in [?], we assume that we would have found similar results as in the S.T. or G.M. layout. The scores of G.M. and S.T. are not 1 because of the algorithms that are used to free space for the labels. If we remove these algorithms (see section 3.2), then the areas are perfectly correlated with the node sizes.

Stability: Table 2 summarizes the element displacement data when the weights of the tree’s elements are modified. To measure these displacements, we computed the center of the bounding box of each region. Then for each time stamp (from 2006 to 2010), we measured the standardized displacement between the elements that are present in the two considered years (the 2006 layout was used, which is not given here, to compute the 2006-2007 measure). The displacement is standardized using the diagonal of the bounding box of each layout (i.e., the maximum possible displacement of an element). As expected, the S.B. technique has obtained the best result during our experimentation and the S.T. has the worst result. The S.T. result arises from the ordering of the elements needed for the packing algorithm that optimizes the aspect ratio. For the G.M. one can see that the stability of the algorithm is between the stability of the S.T. and the S.B.. Even if the G.M.’s algorithm does not change the order of the elements, changing the size of an element at the beginning of the Gosper curve can shift all of the other nodes and can introduce a large number of small displacements. The same situation is not the case with S.B. because the angular sectors are redistributed on a circle. However, after analyzing the small standard deviation of the G.M., which is almost similar to that of S.B., the displacement of the nodes should not affect the readability of our visualization.

Discussion

The initial objective of this work was to create containment visualizations that resemble cartographic maps. During all of the presentations of this work to users, we only received positive feedback. We also observed true enthusiasm for the use of such type of visualizations. Users especially found them to be very aesthetically pleasant. Attractiveness is a significant asset, especially for a large audience or readership. Moreover, for other

visualizations, the use of decorations and embellishments can have a positive influence on comprehension and memorability [?]. A potential advantage of our design is therefore to improve some user performance tasks.

We ran the measurements presented in subsection 5.1. The data indicate that, when using our technique, we can obtain similar quality and even outperform other algorithms. This evaluation allows us to claim that our technique respects the most important requirements of a treemap representation, not that it is superior to other treemaps.

The application of our technique to time-stamped datasets demonstrates that even if the number of displacements is higher than for the S.B. algorithm, it is straightforward to track the evolution, because the amplitude of the displacement is small. For example, in Figure 10, if we observe the “Commerce and Housing Credit” region, we can see that even if the “mycost” parameter evolves significantly over time, the evolution can easily be followed. Thus, G.M. can be used in tasks that require tracking the evolution of the data, which is similar to the spatially ordered treemaps [?].

Furthermore, because we use complex shapes to represent our regions, we noticed that such a representation helps when memorizing the produced map of the dataset. Regarding the “What We Pay For” dataset, part of the diagram was referred to using geographical terminology. For example, considering the yellow region in Figure ??, we immediately used the term “China”. Compared to S.T., in which all of the regions have a similar shape (almost square), our technique appears to be able to use cognitive skills. To confirm that observation, we ran a user evaluation in a professional environment (see Section 5.2).

5.2 Usability Study

We led a user study in a professional environment at the French National Audiovisual Institute (Ina) whose mission is to archive French TV and radio. The user study is based on the visualization of the Ina thesaurus which is a hierarchy of keywords that are used for document annotation. Based on a previous similar approach

[?], we believe that most advantages of the GosperMap would appear after long-term use and would be difficult to measure. The study is not meant to formally or quantitatively measure the benefits of the GosperMap, but focuses on its usability in a professional application context. We then tested and gathered the first impressions on the use of the GosperMap visualization of the thesaurus for annotation purposes.

Objectives

Evaluation of information visualization techniques is based on testing both visual representation and interaction mechanisms [?]. According to the norm ISO 9241-11[?], [?] we analyze the usability of the system based on its interactivity and visual features: effectiveness, efficiency, and user satisfaction [?], [?]. In addition, we are interested in testing the memorization and ease of learning. We also analyze the system according to the norm ISO 9241-12 on the information presentation, testing the graphic user interface.

Dataset

The data is the Ina thesaurus, which consists of 9420 terms represented as a hierarchy. Annotators use a set of terms to label a document, to ease its retrieval. The team in charge of the thesaurus wants to make it more accessible for annotators and newcomers, to maintain the high quality of the Ina archives. Currently, annotators can ask for the presence of a term in the thesaurus, but do not have the possibility of seeing its context. The system features are designed to fit the users professional needs.

Interaction

Because we only display the label that is described in 4.2.1 on one level at a time, to provide a clear decomposition of the different levels of the hierarchy, we implemented an additional simple interaction method to change the depth that we are focusing on. We used the algorithm described in [?] to implement a zoom and pan, to travel the hierarchy more easily. As shown in Figure 4.2.1, we added a lifeline, that displays as many boxes as the tree is deep and colors them depending on which subtree has the focus and at which depth we are positioned. The leftmost green boxes represent the parents of the currently focused region, and the rightmost boxes show in brown the deepest local subtree (additional gray boxes indicate the global deepest tree if it stands in a different branch). For more details on interaction techniques with hierarchical graphs, see [?].

Methodology

We set user tests based on professional tasks to meet the Ina users motivations. Evaluating the system's discovery and its adequacy with the users' needs, the

tests measured the task completion only in terms of the answers. The validation of the users operations (the quality of their answers) was assessed with the help of an additional expert annotator, who helped us design the tasks.

The users, 3 males and 3 females, were chosen considering 3 groups based on their thesaurus knowledge:

- 2 experts: they are the managers of the Ina thesaurus and fully know it.

- 2 annotators: they have a partial knowledge of the thesaurus knowing some keywords but not the structure linking them.

- 2 researchers: they are experts in visualization prototypes but have no knowledge about the thesaurus.

Each test requires approximately 2 hours, including a presentation of the system and the evaluation, some learning manipulations and 3 task achievements. Two additional questionnaires complete the test. The first questionnaire related information on readability and presentation. The other questionnaire regarded the ease of use, the performance, the relevance of the system and the user satisfaction.

Tasks

The tasks are set as "high level" tasks [?], [?], because they do not aim to only find one element in the tree and instead effectively produce an analysis by using the map.

Task 1, the tree topology analysis: This task concerns the readability and understanding of the visualization. The user should compare and order the branches or nodes according to their size and their depth. The objectives are: to see whether the tree is balanced; to find whether there is a specific pattern; to order several terms by their level; and, for a given term, to locate and observe its neighborhood, its level, and the level of its children.

Task 2, annotation (find a node by its context): We choose 5 web pages of news articles that contain an image. The user should navigate in and interact with the treemap to find the best keywords to annotate the image on the page. This request means to navigate the map and choose a term, checking at its level or below to determine whether there is a more specific word that could be a better match. The images were chosen so that the annotation should contain at least 3 terms from distinct branches of the thesaurus.

Task 3, ease of memorization: We tested whether the term's characteristics and location are easy to remember. We first asked the users to find the terms that were used in the previous tasks using the navigation system on a map with no label. Finally the users then must be asked to put back the first level labels (representing 10 nodes) on the blank map.

Results

Observations: On Task 1, all of the users were able to successfully finish their tasks without major errors;

however, some approximations in their answers existed. The expert users immediately recognized the different elements of the thesaurus on the map by analyzing the tree structure, recognizing branches by their graphic attributes, and picking specific patterns (the largest branch in terms of the total number of leaves, deep branches with a small number of leaves, or shallow branches with a large number of leaves). The representation allowed the users to have good global readability, while annotators found it difficult to read the depth of a branch in the map (but easy with the lifeline). They found different depth perceptions depending on the colors that were used for the boundaries and the leaves. The depth-label representation gave the users difficulty in reading the map topology especially when the labels were too long. The depth and size perception for a specified node is approximate but sufficiently good for this application.

On Task 2, all of the users succeeded in achieving their tasks. Notably, in spite of some differences between the chosen sets of keywords, all of the annotations were considered to be valid. The work load was increased by the interaction because the users could see far away neighbors on the map; the users wanted to reach them but were required to come back to the closest ancestor before getting to them. The visual metaphor is good enough that users want to navigate this map the same way they navigate a cartographic map. There is also no possibility to jump back to an ancestor that is farther back than the node's father.

On Task 3, every user was able to immediately find a term the second time. We observed two main behaviors: users remembered the path to a specific node, and users remembered the actual location of a specific node on the map. Most of the users could remember the labels and, then, place them exactly on the map; the other users remembered their locations after being given the labels.

Discussion

We confirmed that all of the users actually achieved their tasks and found all of the elements that are required to conduct an analysis based on the map, validating its interaction effectiveness. With respect to the visual effectiveness, the users were able to read all of the tree elements. The exact depth perception of the deepest elements is more confusing (perhaps because of the choice of colors), but exploring them with the lifeline perfectly fills this lack. Instead of the usual number of direct children, the map shows within eyesight the number of leaves under a tree node and its lineage, which is a useful parameter for the thesaurus manager to assess the tree balance.

The users easily found the context of any term while searching it, without any additional manipulation or getting lost in the thesaurus, which validates the efficiency of the system. The hierarchical representation allows the users to easily compare the size of the areas of different shapes. The labels in their difference of sizes confuse the

users, with the largest labels attracting them regardless of their node characteristics.

Most of the users have been eager to use this system. The users have expressed a substantial amount of positive feedback, but also recommended potential improvements, showing their immediate appropriation of the tool. The users have expressed very concrete advice on interaction and visual encoding improvements. Moreover, the users want to enrich the map with different types of information such as the occurrences of each term in the archive. The thesaurus also contains relationships between terms; the users would like to see whether the layout can be constrained by these relationships (term 1 is near term 2 because they share a relationship).

6 CONCLUSIONS AND FUTURE WORKS

We have presented a method for visualizing hierarchical data as a map through the use of a 2D space-filling curve. Elements of the hierarchy are displayed as nested shapes following the region containment paradigm of the treemaps. The algorithm presented in section 3.2 that regards the layout is the first contribution of this paper. Although the solution is rapid, the hierarchy is not easily readable. The second contribution of our paper is the improvement of the first algorithm to separate the boundaries between nested regions. This improvement helps to more comfortably distinguish the hierarchy and the boundaries can eventually be used to display the labels. The final contribution of this paper is the labeling algorithm for concave shapes. This solution is not only useful for our technique but also can be applied to other applications in the cartography field.

Such visualization ensures area correlation and the stability of the layout. The input ordering of the nodes is preserved, and represents itself as an additional visual variable that is expressed through node proximity (even though our evaluation did not take such property into account). Moreover, the GosperMap presents an interesting and almost unique shape for each non-leaf node of the tree (which we suspect can influence long term memorization of the map), and pleasant aesthetic shapes (which improve the prospects of users adopting the interface).

We have validated the use of the GosperMap in a professional documentation context. The limitations mentioned by the users concern only a few minor implementation aspects. Users even proposed many useful improvements in both the interaction (offering a dual exploration that is both tree-based and GosperMap-based) and in the visualization (the relief effect could be improved by a better color choice, and the layout could be constrained with term relationships). We have shown the stability and the ease for memorization of this type of map, which are two qualities that are required for rapid modification detection.

Following the interest of professional users for this visualization, a more complete implementation integrating the temporal evolution of the hierarchy is underway,

together with constraints on the node ordering so that the spatial positioning relates to transversal proximities. We are also planning to improve the labeling readability, and the boundary relief effect. Our future work will focus on to what extent GosperMap visualizations are preferred over other treemaps and what influence they have on long term memory and mass adoption.

David Auber received his PhD degree from the University of Bordeaux I in 2003. He has been an assistant professor in the University of Bordeaux Department of Computer Science since 2004. His current research interests include information visualization, graph drawing, bioinformatics, databases, and software engineering.

[width=1in,height=1.25in,clip,keepaspectratio]img/auber.eps

Charles Huet received his Masters degree in 2009 at University of Bordeaux I, Software Engineer at INRIA Bordeaux Sud-Ouest since 2010. His interests are Software Engineering and Information Visualization.

[width=1in,height=1.25in,clip,keepaspectratio]img/charles.eps

Antoine Lambert received his PhD degree from the University of Bordeaux I in 2012. He is currently a research engineer at the University of Bordeaux I and the LaBRI. His research interests are information visualization and graph drawing.

[width=1in,height=1.25in,clip,keepaspectratio]img/antoine.eps

Benjamin Renoust did his graduate studies in both the University of Technology at Belfort (UTBM - France) and the Korea Advanced Institute of Science and Technology (KAIST - South Korea) in Computer Science and Engineering. He started preparing his PhD in 2009 at both the LaBRI (University of Bordeaux I) and the National Institute for Audiovisual (Ina - Paris) with interests in graph analysis and visualization.

[width=1in,height=1.25in,clip,keepaspectratio]img/renoust.eps

Arnaud Sallaberry received his PhD degree from the University of Bordeaux I in 2011. He is currently a post doctoral researcher at the University of California, Davis, in the ViDi research group. His current research interests include information visualization, network analysis and graph drawing.

[width=1in,height=1.25in,clip,keepaspectratio]img/arnaud.eps

Agnès Saulnier is a Researcher/Engineer in the research department of INA. She received her Engineering degree from the Ecole Nationale Supérieure des Télécommunications (TELECOM Paris) in 1992, and a Master of Mathematics in the University of Paris VI in 1990. She was firstly involved in research and development in image analysis. Her current research includes evaluation methodologies.

[width=1in,height=1.25in,clip,keepaspectratio]img/agnes.eps