Nmap: A Novel Neighborhood Preservation Space-filling Algorithm

Felipe S. L. G. Duarte, Fabio Sikansi, Francisco M. Fatore, Samuel G. Fadel, and Fernando V. Paulovich

Figure 1. Nmap visual representation of ten years of scientific papes of the IEEE Infovis conference and some other papers frequently cited by them. Each rectangle represents a group of papers with related content. Similar groups are placed close to each other and the dissimilar ones are far apart positioned, obeying the distance-similarity metaphor. This representation allows users to browse document collection into different levels of abstraction, starting with a high-level view and successively detailing the areas of interest until the documents are presented, speeding-up this exploratory process.

Abstract—Space-filling techniques seek to use as much as possible the visual space to represent a dataset, splitting it into regions that represent the data elements. Amongst those techniques, Treemaps have received wide attention due to its simplicity, reduced visual complexity, and compact use of the available space. Several different Treemap algorithms have been proposed, however the core idea is the same, to divide the visual space into rectangles with areas proportional to some data attribute or weight. Although pleasant layouts can be effectively produced by the existing techniques, most of them do not take into account relationships that might exist between different data elements when partitioning the visual space. This violates the distance-similarity metaphor, that is, close rectangles do not necessarily represent similar data elements. In this paper, we propose a novel approach, called *Neighborhood Treemap (Nmap)*, that seeks to solve this limitation by employing a slice and scale strategy where the visual space is successively bisected on the horizontal or vertical directions and the bisections are scaled until one rectangle is defined per data element. Compared to the current techniques with the same similarity preservation goal, our approach presents the best results while being two to three orders of magnitude faster. The usefulness of Nmap is shown by two applications involving the organization of document collections and the construction of cartograms, illustrating its effectiveness on different scenarios.

Index Terms—Space-filling techniques, treemaps, distance-similarity preservation

1 Introduction

Space-filling techniques compose a family of algorithms that seek to use as much as possible the available visual space to represent a dataset. Examples include the Sunburst [16], Quantum Bubblemaps [2], Voronoi Treemap [1, 9], and the rectangular Treemap [6, 14] and its variants [3, 22, 2, 24, 21, 18]. In common, all techniques seek to split the visual space into regions that represent the data elements, with areas proportional to some attribute, or weight, associated with them.

Amongst them, rectangular Treemaps, or simply Treemaps, have received wide attention due to their simplicity, reduced graphical complexity, and compact use of the available visual space [24]. Several different algorithms have been proposed, however the core idea is the same, to split the visual space into rectangles following some criteria. Examples of techniques that implement different criteria are the Squarified and Cluster Treemaps [3, 22], which seek to keep as much as possible the aspect ratio of the rectangles close to one, and the Quantum Treemap [2], which creates rectangles with fixed sizes. Although both criteria can successfully generate more pleasant layouts if compared to the original technique, they were not designed to take into account relationships that might exist between different elements when partitioning the visual space. This violates the distance-similarity metaphor [24], that is, close rectangles do not necessarily represent similar data elements which is an important aspect recognized by previous work [4, 7, 24].

Some attempts have been made in order to incorporate similarity relationships on the produced layouts. Most of them impose a one-dimensional ordering to the data elements and arrange the produced rectangles based on that. Although a better preservation of similarity relationships is attained, mapping one-dimensional information to a two-dimensional space still violates the distance-similarity metaphor in most cases. Currently, few rectangular Treemaps techniques are capable of mapping two-dimensional ordering to the two-dimensional visual space. Nervertheless, even those techniques violate the distance-similarity metaphor depending on the distribution of the

• *Felipe S. L. G. Duarte, Fabio Sikansi, Francisco M. Fatore, Samuel G. Fadel, and Fernando V. Paulovich are with the Institute of Mathematics and Computer Science, University of São Paulo, São Carlos, SP, Brazil. E-mail: {felipelageduarte, fhenrique, fmfatore, samuel.fadel}@usp.br and paulovic@icmc.usp.br*

Manuscript received 31 Mar. 2014; accepted 1 Aug. 2014. Date of publication 11 Aug. 2014; date of current version 9 Nov. 2014.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org.
Digital Object Identifier 10.1109/TVCG.2014.2346276

weights associated with the data elements.

In this paper, we introduce a novel algorithm to create rectangular Treemaps, called *Neighborhood Treemap (Nmap)*. Nmap is based on a slice and scale strategy where the visual space is successively bisected horizontally or vertically and the bisections are scaled. This process is repeated until one rectangle is defined per data element. Our tests demonstrate that Nmap outperforms the existing techniques in terms of preserving the two-dimensional ordering when creating a layout, while being two to three orders of magnitude faster.

2 Related Work

Different approaches have been proposed to create rectangular Treemaps. In this section, we focus on strategies that take similarity relationships into account when defining and positioning the rectangles.

Jigsaw [23] is one example of technique that seeks to preserve similarity relationships. It splits the available space into a regular grid and employs both a one-dimensional ordering based on a pivot data element and space-filling curves to traverse the grid. Although it successfully creates regions proportional to given weights, it defines irregular shapes (non-rectangular), making it more difficult to visually compare different areas [21]. Tak and Cockburn [18] tackle this problem by recursively splitting the visual space into quadrants, therefore rectangular areas, which are positioned following a Hilbert or Moore curve of level zero. The created regions are rectangular, however, one-dimensional ordering information is lost in the process.

The Spiral Treemap [21] and the One-dimensional Ordered Treemap (OOT) [2] present better results in terms of the one-dimensional ordering preservation. The former technique uses a one-dimensional ordering of the input data positioning the Treemap regions from the center of the visual space to its border following a circular arrangement. The latter technique is an adaptation of the Squarified Treemap technique [3] changing the order the Treemap regions are processed, following a given one-dimensional ordering. When two-dimensional positions associated to the Treemap regions are given as input, the one-dimensional ordering is defined by calculating the distance from each position to the left corner of the available space and using this to order the regions. Although the continuity of the ordering can be preserved, the two-dimensional similarity relationships are not guaranteed, thus violating the distance-similarity metaphor. Differently, our approach does not rely on one-dimensional orderings to preserve the similarity relationships. Instead, we use two-dimensional information to define the placement of the rectangles on the final layout, rendering a better approximation of the distance-similarity metaphor.

One of the few approaches that seek to overcome this problem is the Spatially-Ordered Treemap (SOT) [24]. It is also an adaptation of the Squarified Treemap [3] but that seeks to preserve two-dimensional orderings into the two-dimensional space. This is accomplished, as in the OOT technique, by carefully defining the order the elements are processed. SOT processes the elements ordering them so that the next element to be processed is the most similar element to the previously processed one. When two-dimensional positions are associated to the Treemap regions, this similarity is calculated considering these positions, therefore resulting on layouts that preserves the relative spatial position of a given input. Although this renders better results in terms of the distance-similarity relationships preservation, the quality of the results depends on the distribution of the weights of each rectangle and the ratio between the largest and smallest weights. The more different from a constant distribution with small weight ratio, the worse the result. Our technique is less sensitive to this, attaining better results in terms of preserving the given two-dimensional information, producing less elongated rectangles in a fraction of the required time.

Other attempts on creating space-filling techniques that preserve two-dimensional similarity relationships have been made. The Self-Sort Map [17] is one example. It is similar to a sorting algorithm in two dimensions, organizing the data elements into a structured layout. The drawback in this case is that all regions present the same size on the produced layout, impairing its applicability when different weights are assigned to different data elements. The same happens with IncBoard [12], besides wasting too much visual space on the process of partitioning and placing the visual elements. We also seek to preserve the similarity relationships on the produced layout in our approach. However, all the available space is used, and the rectangles can present different areas.

3 Slice and Scale Treemap Algorithm

Neighborhood Map (Nmap) tackles the problem of preserving distance-similarity relationships when constructing a Treemap using an approach that consecutively bisects the available area and scales the resulting bisections. Let $\mathscr{D} = \{d_1, \ldots, d_n\}$ be the data elements with $P : \mathscr{D} \to \mathscr{P}$ a function that assigns weights $\mathscr{P} \in \mathbb{R}$ (a selected data attribute) to each element. Let also R be the rectangle enclosing the available area to display the Treemap and $\mathscr{X} = \{(x_1, y_1), \ldots, (x_n, y_n)\} \in \mathbb{R}^2$ the Cartesian coordinates assigned to points inside R representing the data elements. The reasoning behind our algorithm is to apply this slice-scale process until one rectangle is defined per element in $\mathscr{D}$. These rectangles and their positions will present areas proportional to the weights $\mathscr{P}$, preserving the similarity relationships in $\mathscr{X}$, that is, close points in $\mathscr{X}$ should define near rectangles, and far apart points should define distant rectangles on the final layout.

The rectangles can be vertically or horizontally split. On the horizontal bisection, a vertical segment b_v is defined splitting R into two rectangles R_A and R_B so that $R_A \cup R_B = R$ and $R_A \cap R_B = \emptyset$. Weights p_A and p_B are associated to R_A and R_B computed as the summation of the weights of the elements in each rectangle, that is, $p_A = \sum_{d_i \in R_A} P(d_i)$ and $p_B = \sum_{d_i \in R_B} P(d_i)$. Based on that, R_A and R_B are horizontally rescaled in order to present areas proportional to p_A and p_B, respectively.

Let w_{R_A} and w_{R_B} be the widths of R_A and R_B, the widths of the transformed rectangles R'_A and R'_B can be computed as a function of p_A and p_B and the total weight of R ($p_A + p_B$)

$$\begin{aligned} w_{R'_A} &= \tfrac{p_A}{p_A+p_B} \cdot w_R \\ w_{R'_B} &= \tfrac{p_B}{p_A+p_B} \cdot w_R \end{aligned} \tag{1}$$

where w_R is the width of R. Using this information and considering (x_R, y_R) the Cartesian coordinates of the top left corner of R, the non-rotational rigid transformation in homogeneous coordinates to transform R_A and R_B into R'_A and R'_B horizontally rescaling each area according to their weights are

$$\begin{aligned} \mathbf{H}_{R_A} &= \begin{bmatrix} \frac{w_{R'_A}}{w_{R_A}} & 0 & x_R(1-\frac{w_{R'_A}}{w_{R_A}}) \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \\ \mathbf{H}_{R_B} &= \begin{bmatrix} \frac{w_{R'_B}}{w_{R_B}} & 0 & (x_R+w_R)(1-\frac{w_{R'_B}}{w_{R_B}}) \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{aligned} \tag{2}$$

Since these are linear transformations, the similarity relationships between the points inside each rectangle are preserved. These matrices are applied to the points and the rectangles, changing the x-coordinates of them. The matrices $\mathbf{V}_{R_A}$ and $\mathbf{V}_{R_B}$ for the vertical transformation are similarly obtained, only changing the bisector segment to be horizontal (b_h) and obtaining scale factors to change the y-coordinates.

$$\begin{aligned} \mathbf{V}_{R_A} &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{h_{R'_A}}{h_{R_A}} & y_R(1-\frac{h_{R'_A}}{h_{R_A}}) \\ 0 & 0 & 1 \end{bmatrix} \\ \mathbf{V}_{R_B} &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{h_{R'_B}}{h_{R_B}} & (y_R+h_R)(1-\frac{h_{R'_B}}{h_{R_B}}) \\ 0 & 0 & 1 \end{bmatrix} \end{aligned} \tag{3}$$

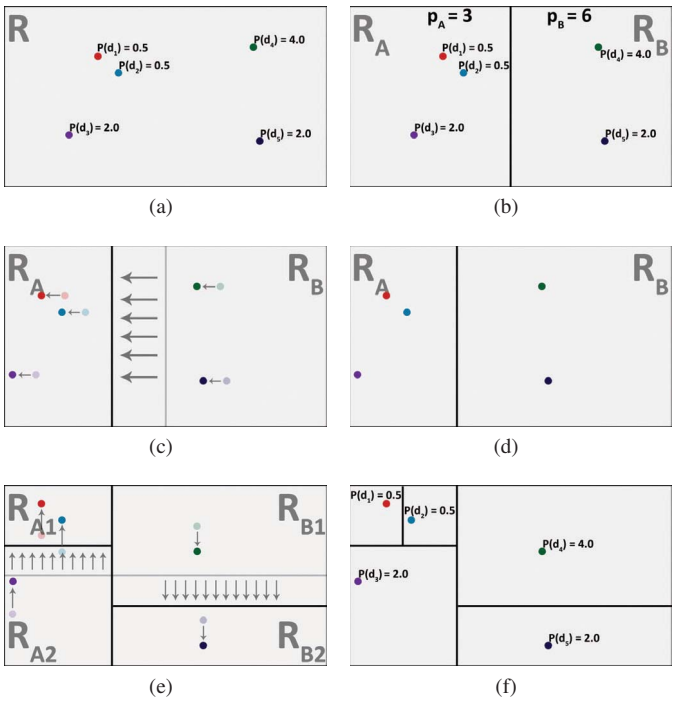


Figure 2. Overview of the slice and scale process. (a) Initially, points representing the data elements are positioned inside the enclosing rectangle. (b) Since the rectangle's width is larger than its height, the first bisection is horizontal. (c,d) Based on the summation of the weights of the elements on each resulting rectangle, scales are executed so that the rectangles present areas proportional to that. (e) This process of bisection and scaling is executed alternating horizontal and vertical cuts (f) until each element is assigned to one rectangle with area proportional to its weight.

where h_R, h_{R_A}, h_{R_B}, $h_{R'_A}$, and $h_{R'_B}$ represent the height of R, R_A, R_B, R'_A, and R'_B, respectively.

Figure 2 illustrates this process. The first step is to create one rectangle R enclosing the area to display the Treemap and to assign positions to the points representing the data elements (Figure 2(a)). In this example, the dataset is composed by 5 data elements. After that, R is bisect into two rectangles R_A and R_B and the weights p_A and p_B are calculated (Figure 2(b)) for each rectangle. In this example, we are performing a horizontal bisection. Next, the rectangles and the points belonging to them are scaled (Figure 2(c)) creating new rectangles R'_A and R'_B with areas proportional to p_A and p_B (Figure 2(d)). This process is then applied to R'_A and R'_B (Figure 2(e)). This is repeated until each rectangle contains only one point (Figure 2(f)). Algorithm 1 depicts the complete process.

3.1 Defining the Bisectors

Different variations of this algorithm can be attained changing the direction of the bisections and the positions of the horizontal (b_h) and vertical (b_v) bisectors. In this paper, we devise and evaluate two different strategies: *Alternate Cut (Nmap-AC)* and *Equal-Weight (Nmap-EW)*.

The *Nmap-AC* is the simplest one. In this strategy, the horizontal and vertical bisections are alternated. If R is horizontally split, R_A and R_B will be vertically split, and vice-versa. The direction of the first bisection is decided based on the width (R_w) and height (R_h) of R. If $R_w > R_h$ the bisection is horizontal, vertical otherwise. The positions of bisectors b_h and b_v are defined so that R_A and R_B present the same number of data elements (if the number of elements is odd, R_A will contain one more element than R_B). To define these positions, the elements are initially ordered in ascending order considering the x-coordinates, for the horizontal bisection, and y-coordinates, for the vertical bisection (two orderings are created). Based on that, the first half of the elements is assigned to R_A and the remaining ones to R_B. The position of the bisector is calculated as the average of the larger coordinate of the elements in R_A and the smallest coordinate of the elements in R_B. Since non-rotational linear transformations are applied to R_A and R_B, the order of the elements on both directions does not change after the transformation. Thereby, the initial ordering can be used to split R_A and R_B and so on.

Algorithm 1 Nmap algorithm.

```
function SLICESCALE(ℛ, R)
    if |R| = 1 then                          ▷ R contains one instance
        ℛ ← ℛ ∪ {R}                          ▷ add R to the list of rectangles
    else
        dir ← DIRECTION(R)                   ▷ get the direction to bisect
        (R_A, R_B) ← BISECT(R, dir)          ▷ bisect R given a direction
        if dir is horizontal then
            Calculate H_{R_A} and H_{R_B}    ▷ see Equation(2)
            SLICESCALE(ℛ, H_{R_A} · R_A)
            SLICESCALE(ℛ, H_{R_B} · R_B)
        else
            Calculate V_{R_A} and V_{R_B}    ▷ see Equation(3)
            SLICESCALE(ℛ, V_{R_A} · R_A)
            SLICESCALE(ℛ, V_{R_B} · R_B)
        end if
    end if
end function
```

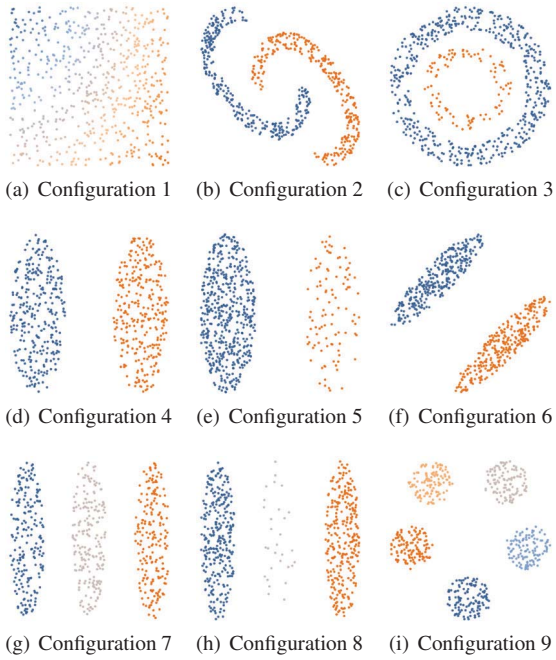
The second strategy, the *Nmap-EW*, seeks to improve the aspect ratio of the attained rectangles. Different from the first strategy, R is split into R_A and R_B so that $p_A \approx p_B$. The direction of each bisection is decided based on the width and height of R. If $R_w > R_h$ the bisection is horizontal, vertical otherwise. To find the position of the bisector, the elements are ordered in ascending order considering the x-coordinates, for the horizontal bisection, and y-coordinates, for the vertical bisection (two orderings are created). Then, they are added one-by-one to R_A from the smallest coordinate to the largest one until the difference $|2 \cdot p_A - p|$ is minimised, where p is the weight of R. The remaining elements are assigned to R_B. The position of the bisector is calculated as the average of the largest coordinate of the elements in R_A and the smallest coordinate of the elements in R_B. Again, the initial ordering can be used to split R_A and R_B and so on.

3.2 Initial Placement

The aim of our algorithm is to preserve, amongst the Treemap rectangles, the similarity relationships that exist between the positions $\mathscr{X} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ assigned to the data elements inside R. If the goal is to preserve the similarity relationships between the data elements $\mathscr{D} = \{d_1, \ldots, d_n\}$, $\mathscr{X}$ should reflect that. One solution is to apply a multidimensional projection technique [19] which maps multidimensional data elements to the plane preserving, as much as possible, the similarity relationships amongst them. If $\delta(d_i, d_j)$ represents the distance between two data elements, and $d((x_i, y_i), (x_j, y_j))$ the distance between the points on the plane representing them. A projection technique seeks to minimize $|\delta(d_i, d_j) - d((x_i, y_i), (x_j, y_j))| \; \forall \; d_i, d_j \in \mathscr{D}$, such as classical scaling [20]. The choice depends on the data domain.

4 Results and Evaluation

In order to evaluate Nmap, we compare the strategies depicted in this paper against state-of-art rectangular Treemap techniques that seek to preserve similarity relationships. One is based on one-dimensional ordering, the One-dimensional Ordered Treemap (OOT) [24], and another one is based on two-dimensional ordering, the Spatially-Ordered Treemap (SOT) [24]. This comparison is carried-out considering three different measures, aspect-ratio, displacement, and neighborhood preservation. Aspect-ratio is the average of the aspect-ratio of all

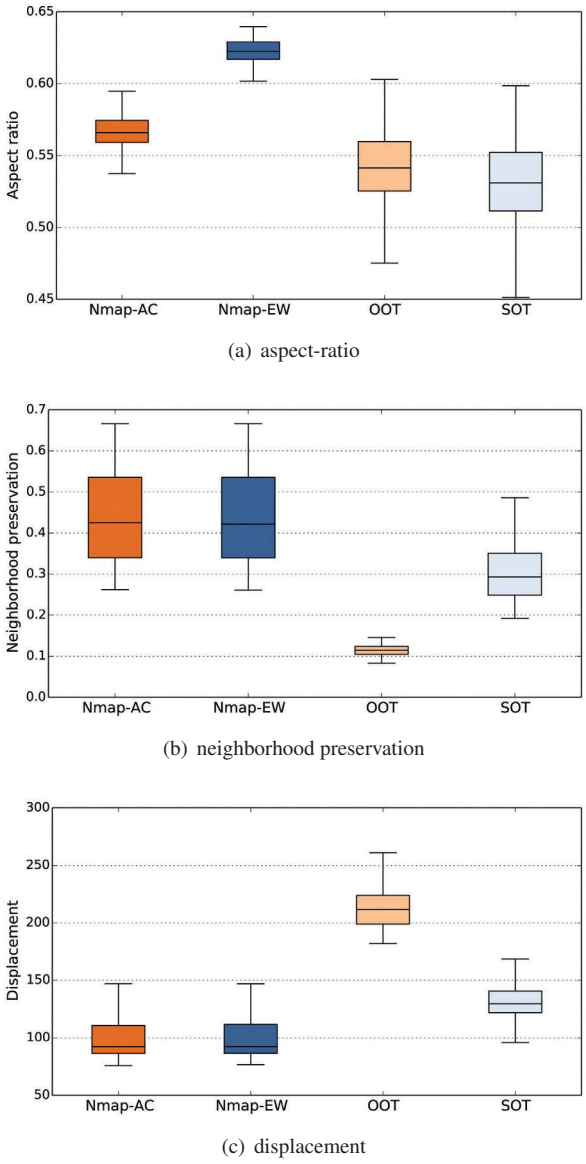


(a) Configuration 1 (b) Configuration 2 (c) Configuration 3

(d) Configuration 4 (e) Configuration 5 (f) Configuration 6

(g) Configuration 7 (h) Configuration 8 (i) Configuration 9

Figure 3. Initial configurations used to evaluate the techniques. These placements were designed to verify different properties of each technique, such as the ability to handle scenarios presenting non-linear relationships, non-orthogonal directions, and with different number of groups of elements and densities.



(a) aspect-ratio

(b) neighborhood preservation

(c) displacement

Figure 5. Boxplots of aspect-ratio, neighborhood preservation, and displacement measures. In all these aspects, the Nmap strategies surpass current state-of-the-art techniques, indicating their quality on obeying the distance-similarity metaphor.

obtained rectangles, where the aspect-ratio of one rectangle R is calculated as $min(R_w/R_h, R_h/R_w)$. In fact this is a normalized version of the aspect-ratio ranging on the interval]0..1]. The larger the value the better the result. The displacement is calculated as the average of the displacement from the original positions $\mathscr{X} = \{(x_1,y_1),\ldots,(x_n,y_n)\}$ to the final positions of each rectangle considering their centroids. Finally, the neighborhood preservation is calculated as the average of the k-nearest neighbor index, varying k. The k-nearest neighbor index is calculated as the average of the percentage of the k-nearest neighbors in $\mathscr{X} = \{(x_1,y_1),\ldots,(x_n,y_n)\}$ that are preserved on the obtained rectangles considering their centroids. Neighborhood preservation is the way we use to evaluate the similarity relationship preservation. All the results were generated in an Intel® Xeon CPU E7-2870 2.40GHz, with 32GB of RAM. All techniques are implemented in Java. We use the original codes, made available by the authors, of the OOT and SOT techniques.

We have employed 9 distinct initial configurations in our experiments, allowing the analysis of the techniques in scenarios with different properties. Figure 3 shows these initial configurations. We split the initial configurations into 3 groups. The first one (first line of Figure 3) contains configurations to measure the capacity of the techniques to preserve the shape of initial placements considering linear and non-linear relationships. The second group (second line of Figure 3) contains configurations to check the capacity of the techniques to keep on the final layout the separation between two groups of elements, varying their densities and orientations. And the third group (third line of Figure 3) is composed by configurations with more than two groups with different densities. All configurations contain 600 points.

Figure 4 presents the results for each one of these initial configurations using the aforementioned techniques. In these layouts, the weights of each data element were randomly defined following a Gaussian distribution. The points' colors of the original placement are used to color the attained rectangles. For the group of shape preserving verification, the OOT presents good results for the configuration 1, but it fails for configurations 2 and 3. This is caused by the one-dimensional ordering imposed by this technique, considering the distances from the points of the initial configuration to the layout's left-upper corner. The good results for the configuration 1 is attained since the color gradient varies from the left-upper corner to the right-bottom corner, so the rectangles close to the diagonal are inherently one-dimensional ordered. The SOT presents good results for configuration 1, but configurations 2 and 3 present final shapes that do not correctly match the original positions. Nmap-AC and Nmap-EW present the best results in terms of preserving the original shapes. The color gradient of the rectangles match the original gradient of the points, and the highly non-linear configurations are consistently preserved on the final layouts.

For the group of configurations that aim to evaluate different densities and directions, the SOT, Nmap-AC, and Nmap-EW techniques attained good results for configurations 4 and 5. The OOT presents the worst results for both configurations in terms of preserving the original shape. In fact, the OOT will always fail when the direction of largest variance of the groups are (closely) parallel to the enclosing rectangle axis. This is due to the one-dimensional ordering that is used which considers the top-left corner of the enclosing rectangle. When this

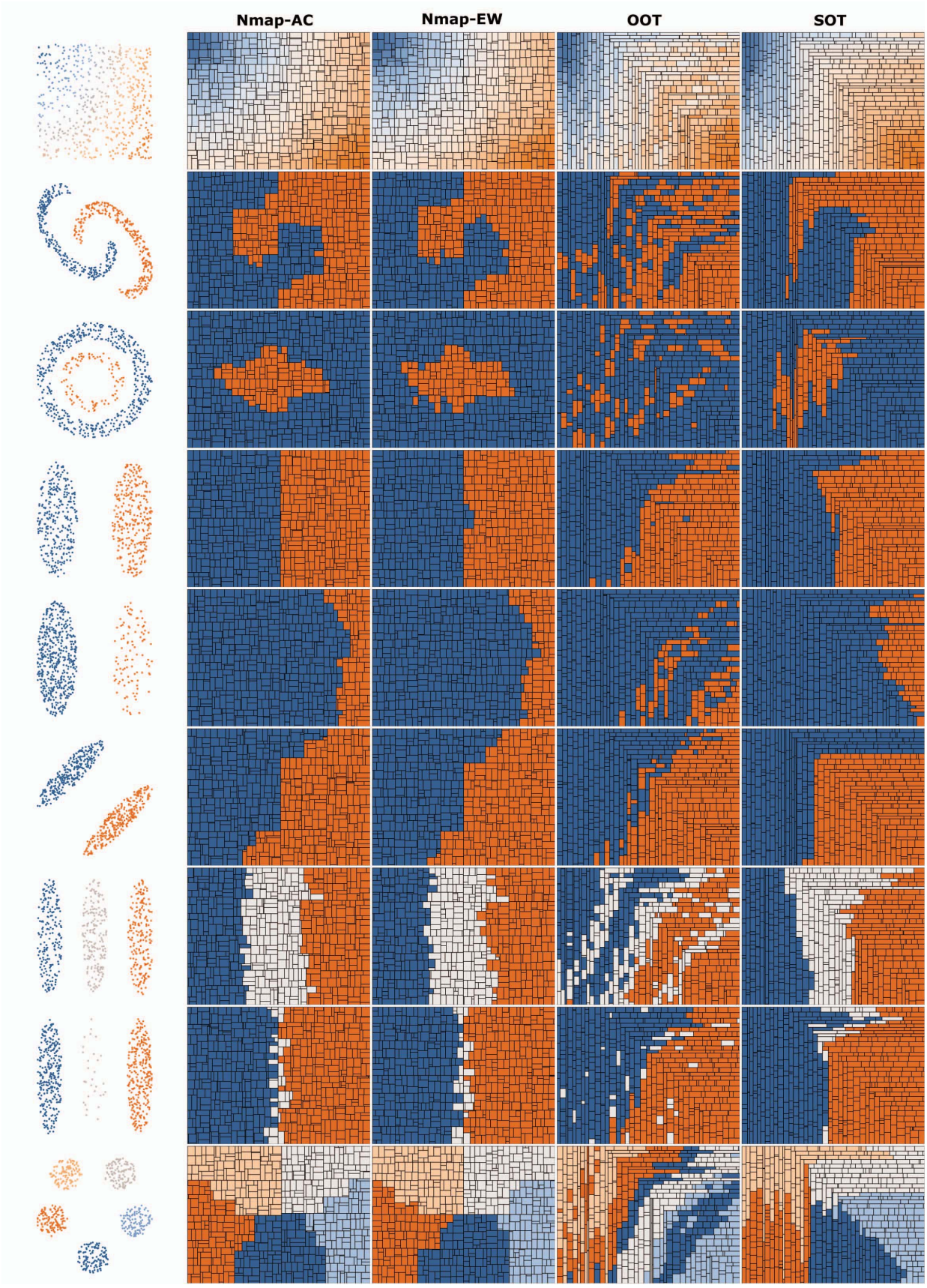


Figure 4. Results of applying Nmap-AC, Nmap-EW, OOT, and SOT techniques on different initial configurations, respectively. The color of the initial points are used to color the produced rectangles. In general, the Nmap strategies produced layouts that best preserves the shape and neighborhoods of the initial configurations, achieving a more reliable representation of the distance-similarity metaphor.

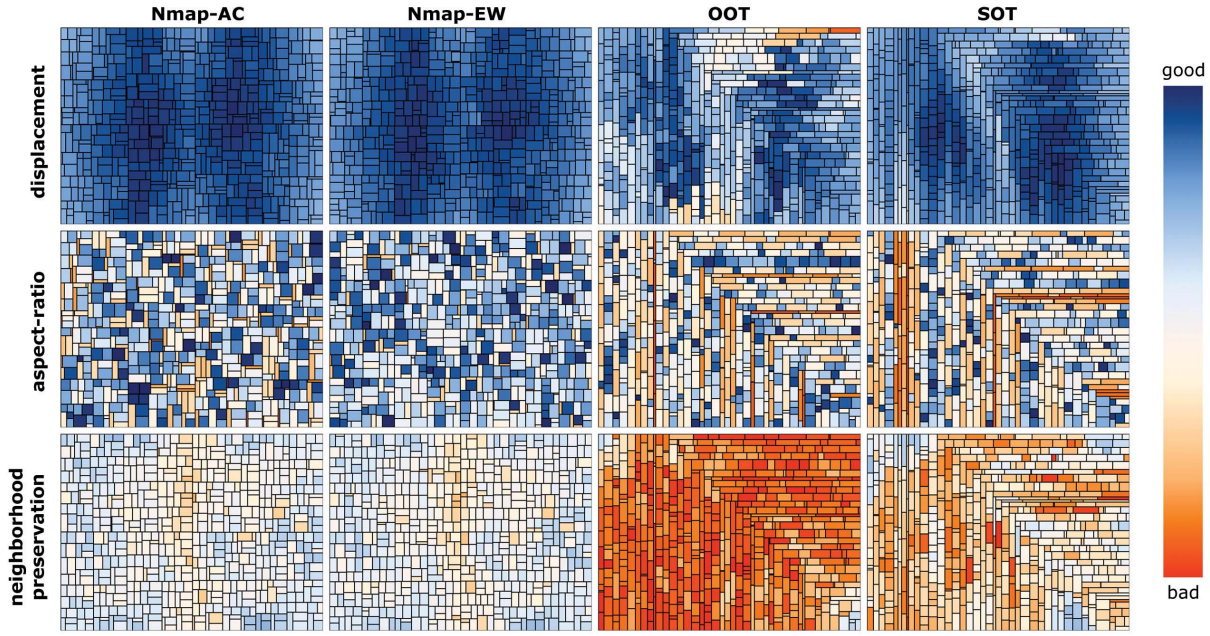


Figure 6. Comparison between SOT, OOT and the Nmap strategies for the configuration 4. The rectangles are colored according to the displacement, aspect-ratio and neighborhood preservation measures. In all these aspects the advantage of the Nmap strategies is evident, not only producing less elongated rectangles, but also better preserving the neighborhood relationships, showing the good compromise of them with the distance-similarity metaphor.

direction is aligned to the rectangle diagonal, the OOT can preserve the initial shape (configuration 6). For the configuration 6, Nmap-AC and Nmap-EW also attained good results, succeeding on presenting some diagonal direction. For this configuration, the SOT technique presents the worst result, almost completely losing the information about the original diagonal placement, violating the distance-similarity metaphor.

The tests with the last group, that seeks to evaluate the techniques when there exist more than two groups of elements, corroborate the quality of Nmap-AC and Nmap-EW strategies. Again, the OOT technique presents the worst results, not being able to preserve on the final layouts the groups and structures present on the initial placement. The SOT technique has problems when a group is surrounded by two denser groups (configuration 8). Even the Nmap strategies being based on consecutive bisections, multiple groups with different densities are successfully captured on the produced layouts. In addition, the groups arrangement is well preserved (configuration 9).

In general, Nmap-AC and Nmap-EW visually present the best results. In order to quantitatively assess that, we have changed the weights of the data elements following a Gaussian distribution considering different ratios between the smallest and largest weights, $1/1$, $1/10$, $1/50$, $1/100$, $1/150$, $1/200$, and $1/250$. Each technique was executed 30 for each ratio. This allows the analysis into different scenarios, varying from rectangles with balanced weights until rectangles with very dissimilar weights. Figure 5 summarizes the results in terms of the aspect-ratio, neighborhood preservation and displacement of the produced layouts. The Nmap strategies obtained better results in terms of the aspect-ratio (Figure 5(a)), being closer to one than the OOT and SOT techniques. Although a simple approach, the Nmap surpass techniques that are adaptations of an algorithm (Squarified Treemap) which is know to render rectangles with very good aspect-ratio. Comparing Nmap-EW to Nmap-AC, the first one presents better results as expected, indicating that the produced rectangles have almost the same aspect-ratio and that the number of elongated rectangles (poor rectangles) is minimized by the process.

In terms of neighborhood preservation (Figure 5(b)), Nmap-AC and Nmap-EW present the best results, while OOT, due to its one-dimensional nature, presents the worst. In this test, we run the k-nearest neighbors varying k from 5 to 20. The same good results are attained on the displacement measures (Figure 5(c)), indicating that the Nmap strategies change their initial placements less than the other techniques when constructing the final layouts. This clearly shows the quality of the Nmap strategies on reliably creating layouts that obey the distance-similarity metaphor.

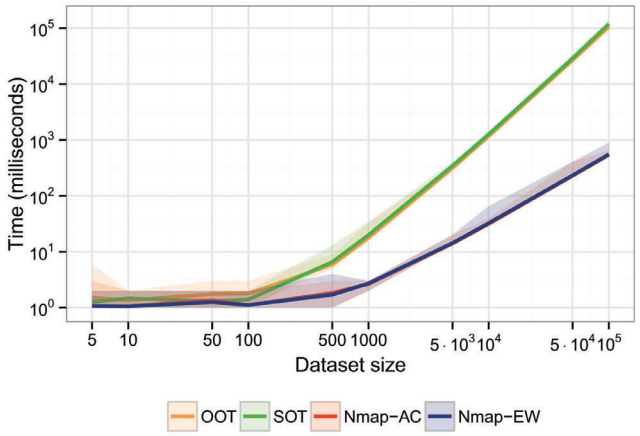


Figure 7. Plot "datset size versus running times". On average, the Nmap strategies present almost the same performance, being two to three orders of magnitude faster than the other techniques.

Figure 6 presents a comparison between the layouts created for the configuration 4 with the rectangles colored according to the displacement, aspect-ratio and neighborhood preservation measures. As in Figure 4, we use a Gaussian distribution to define the weights. The first row presents the result of the displacement. The Nmap strategies not only produce the best displacement on average, but also the deviation is smaller than in the other techniques. Similar results are obtained for the configurations with groups of elements with balanced

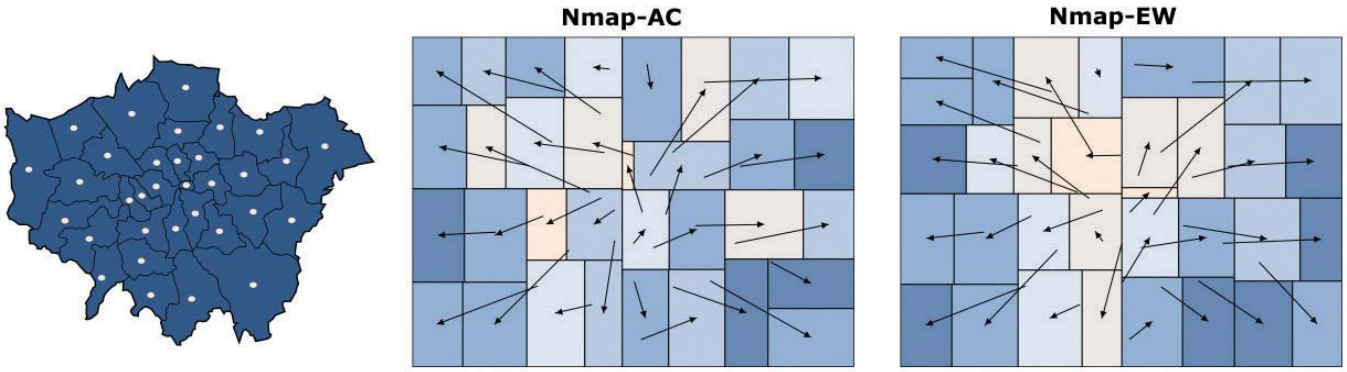


Figure 8. Cartograms from the city of London boroughs. The area of each rectangle indicates the district population, their colors represent the neighborhood preservation and the arrow map the displacement. The small displacement and good neighborhood preservation attained by the Nmap strategies are desirable properties when creating cartograms.

densities. For the configurations with groups with uneven densities, the displacement is larger on the borders between them (configuration 5). This is expected since the points on the group with larger density have to move more to occupy the area of the less dense group. The second row presents the results of the aspect-ratio. The Nmap-EW and Nmap-AC rendered the best results, producing considerably less elongated rectangles than the OOT and SOT techniques, with a clear advantage to the Nmap-EW. Notice that the weight ratio to produce these layouts is $1/50$. Decreasing this, the advantage of Nmap strategies over OOT and SOT is even more evident. The last row of Figure 6 presents the neighborhood preservation. The Nmap strategies present noticeably better results than OOT and SOT, not only on average but also considering the deviation. Similar results are obtained for the other configurations.

The computational complexity of the Nmap strategies are dictated by the bisection phase. The process of the Nmap-AC involves to order the x and y-coordinates once and split the rectangles so that each obtained rectangle has (approximately) the same number of elements. Therefore, it is $O(n \log n)$, where n is the number of data elements. This is the complexity to order the initial points if the coordinates are represented as float numbers. If integers are used, this complexity can be reduced to $O(n)$ using, for instance, the radix sort algorithm. The complexity of Nmap-EW is $O(n^2)$ due to the process which involves traversing all the data elements on a rectangle to find the bisection position where the summation of the weights are (approximately) the same. However, the worst case happens only in one special situation. The weights of the points have to be ascending ordered from the left to the right on the x-coordinate and from the top to the bottom on the y-coordinate. In addition, in all rectangle bisections, the weight of the last element on the ordered list should be larger than the summation of the remaining elements belonging to the rectangles. In fact, this is a very unusual situation, and on average Nmap-EW is much faster than that. Figure 7 presents the plot "dataset size versus running times". In this test, we create randomly positioned datasets with different distributions of weights, varying from 5 to $100,000$ elements, and run 200 times each technique for each dataset. The thin lines indicate the average case and the shadows represent the smallest and largest running time of each technique. Notice that, on the average, the values of the Nmap-EW strategy are practically the same of the Nmap-AC, since the worst case scenario is unusual to happen. However, in some cases the Nmap-EW is slower than the Nmap-AC, and both approaches are much faster than the OOT and SOT techniques, two to three orders the magnitude faster.

In summary, the OOT and SOT techniques obtained better results in terms of the aspect-ratio, an expected outcome since they are based on a technique that is known to produced good rectangles in that sense. For the displacement and neighborhood preservation measures, Nmap strategies produced noticeably better results in a fraction of the required time for large datasets. This demonstrates their ability on handling large datasets keeping a good compromise with the distance-similarity metaphor when placing the rectangles on the final layout.

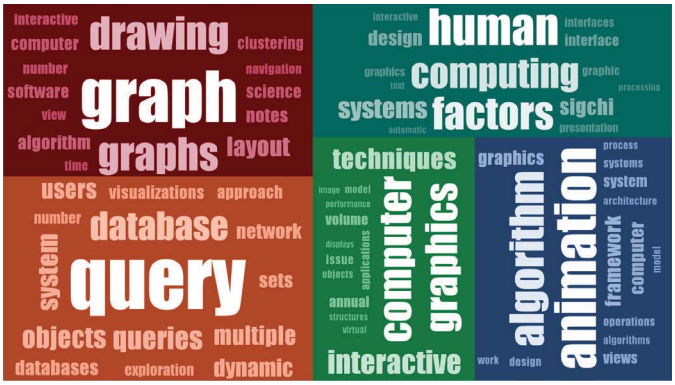


Figure 9. High level view of a document collection composed by scientific papers on visualization. Each rectangle represents a group containing similar papers. Tag-clouds are used to support users on identifying the content of each group.

5 Application

Taking advantage of the small displacement and good neighborhood preservation attained by the Nmap strategies, different applications can be devised. One example is on creating cartograms. Cartograms are maps whose cartographic coordinate space is distorted to accommodate some non-geometrical property [15]. Figure 8 presents the map of the London boroughs and the cartograms generated using the Nmap strategies. These cartograms were generated using the centroid of each borough as the initial placement and their populations as the attribute used to define the rectangles areas. We use the data available at the *London Datastore* web site[1]. The color of each rectangle represents the neighborhood preservation and the arrows indicate the displacement from the original to the final positions (the rectangles centroids). The local neighborhoods are well preserved, and the size of the arrows and the reduced crossings between them indicate the small displacement attained. This good balance between neighborhood preservation and displacement gives evidences that Nmap strategies are good candidates on designing cartograms.

Another application that can take advantage of the Nmap features is the browsing of document collections. In order to apply the Nmap strategies to create this visual representation, firstly all documents are converted into multidimensional vectors following the bag-of-words

[1]London Datastores web site `http://data.london.gov.uk/`

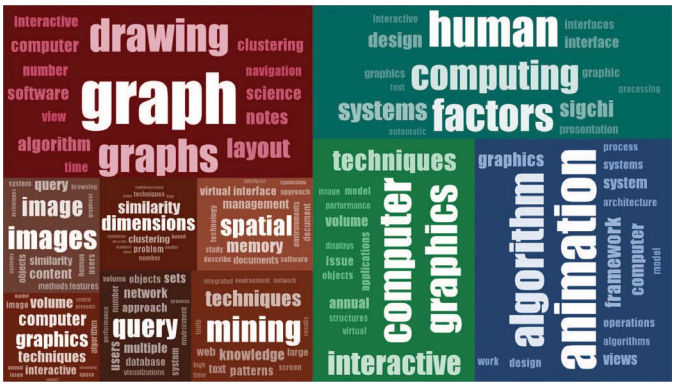


Figure 10. Expanding one group of documents of Figure 9 into subgroups. A more detailed view is provided upon user request.

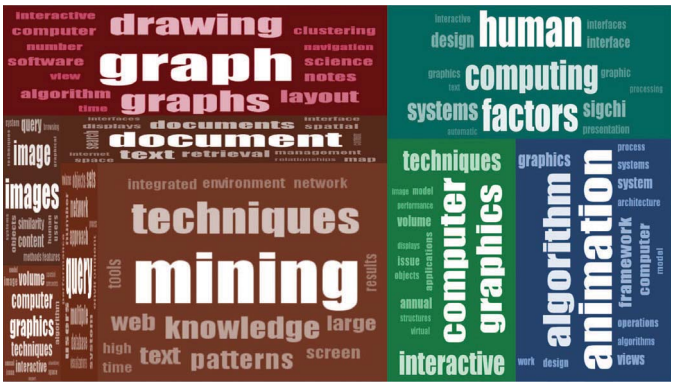


Figure 11. A zoom-in of a sub-group of Figure 10 to improve its readability. Taking advantage of the transformation matrix formulation of the Nmap strategies a zoom operation that preserves the context is easily implemented only changing the weight of the focused rectangle.

representation [13]. Based on that, a clustering algorithm is applied, and the cluster's multidimensional centroids are calculated. The initial placement for the Nmap is obtained projecting these centroids on the plane using a multidimensional projection technique (see Section 3.2). Here, we use the least-square projection [10] since it is indicated to handle document collections. Finally, to represent the content of a rectangle we use the tag-cloud metaphor, making the size of the words proportional to their importance. We use the algorithm proposed in [11] to compute such importance and arrange the words inside the rectangles. Figure 9 shows the result of applying this approach to visualize the IEEE Infovis 2004 contest dataset [5]. It is composed by papers of ten years of the InfoVis conference and some other papers frequently cited by them. In this example, five different groups of papers were created to compose the initial layout. This is a parameter that the user can change, according to his/her expertise about the dataset under analysis, using some metadata information available, or employing an algorithm to automatically estimate the number of groups/clusters [8].

The obtained rectangles can be refined, allowing to browse the document collection in more details. To accomplish that, the documents belonging to a selected rectangle are re-clustered and their centroids are projected inside its area. This is then used as the initial placement for the Nmap. In our application, this refinement process can be executed by the user until a certain minimum number of documents inside a rectangle is reached. When this happens, a list of the documents is displayed. Although this process of creating clusters inside cluster defines a hierarchy of groups and subgroups based on a clustering algorithm, the Nmap strategies can also be used to represent any given hierarchy, being independent of this cluster-based organization. Figure 10 presents the result of expanding one rectangle (cluster) of Figure 9. This reveals the second level of the hierarchy under one node, presenting rectangles (clusters) with more refined tag clouds. Figure 1 shows the result of expanding all rectangles of Figure 9, thereby presenting the entire second level of the cluster hierarchy. The good neighborhood preservation attained by Nmap ensures that similar groups of papers are near placed on the final layout according to the the bag-of-words representation. Every time such representation matches the user expectation, useful visual representations are created.

Taking advantage of the formulation of the Nmap strategies, a zoom-in operation that preserves the context can be easily done. The way the Nmap was proposed, a binary tree of transformation matrices in homogeneous coordinates representing the scales can be created. Thereby, a zoom-in can be implemented increasing the weight of the focused rectangle and updating the scale matrices on the tree. This guarantees not only to keep the context when focusing, but also the proportionality between the remaining rectangles (not on focus). Figure 11 presents the zoom-in of one sub-group of Figure 10 resulted from expanding one group of Figure 9. All the words of the corresponding tag-cloud is easier to read without changing the adjacencies of the rectangles.

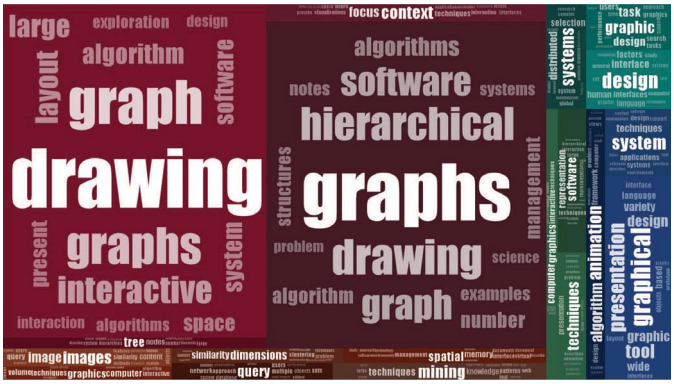


Figure 12. The resulting layout when the rectangles areas are proportional to the relevance of keywords "graph or graphs". Different attributes can be used to define the rectangles areas (weights) adding flexibility to the process of exploring document collections.

On the previous examples (Figures 9 and 10), the areas (weights) of the rectangles are initially defined proportional to the number of documents each group of documents contain. Nevertheless, other attributes can be used, such as, the importance of a given keyword, the number of documents citations and so on, adding flexibility to the process of exploring the collection. On Figure 12 the importance of keywords are used to define the rectangles weights. In this case the keywords are "graph or graphs" and the weight of a rectangle is calculated considering the relevance of the keywords given by the tag-cloud.

6 Conclusions

In this paper we proposed a novel rectangular Treemap algorithm called Neighborhood Treemap (Nmap). Nmap employs a slice and scale approach, creating visual representations that preserve similarity relationships amongst data elements. Considering the design goal of "preserving neighborhood, filling all available space with rectangles of different sizes with aspect-ratio as close as possible to one, performing the smaller displacement, and being able to handle large datasets", the set of comparisons we provided shows that Nmap outperforms other state-of-the-art rectangular Treemaps techniques in all these aspects, being two to three orders of magnitude faster. Thereby, not only better representing the distance-similarity metaphor and reducing problems associated on violating that, but also being able to handle large datasets in real time. Moreover, the flexibility and ease of implementation render Nmap one of the the most attractive Treemaps techniques to create cartograms and visual representations of document collections.

ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers for their constructive comments. This work was supported by the São Paulo Research Foundation (FAPESP) (grants #2012/21022-0, #2013/14650-7 and #2011/22749-8) and CNPq-Brazil.

REFERENCES

[1] M. Balzer, O. Deussen, and C. Lewerentz. Voronoi treemaps for the visualization of software metrics. In *Proceedings of the 2005 ACM Symposium on Software Visualization*, SoftVis '05, pages 165–172, New York, NY, USA, 2005. ACM.

[2] B. B. Bederson, B. Shneiderman, and M. Wattenberg. Ordered and quantum treemaps: Making effective use of 2d space to display hierarchies. *ACM Transactions on Graphics*, 21(4):833–854, Oct. 2002.

[3] M. Bruls, K. Huizing, and J. van Wijk. Squarified treemaps. In *In Proceedings of the Joint Eurographics and IEEE TCVG Symposium on Visualization*, pages 33–42. Press, 1999.

[4] K. Buchin, D. Eppstein, M. Löffler, M. Nöllenburg, and R. I. Silveira. Adjacency-preserving spatial treemaps. In *Proceedings of the 12th International Conference on Algorithms and Data Structures*, WADS'11, pages 159–170, Berlin, Heidelberg, 2011. Springer-Verlag.

[5] J.-D. Fekete, G. Grinstein, and C. Plaisant. Ieee infovis 2004 contest, the history of infovis. `http://www.cs.umd.edu/hcil/iv04contest`, 2004.

[6] 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 '91*, VIS '91, pages 284–291, Los Alamitos, CA, USA, 1991. IEEE Computer Society Press.

[7] F. Mansmann, D. Keim, S. North, B. Rexroad, and D. Sheleheda. Visual analysis of network traffic for resource planning, interactive monitoring, and interpretation of security threats. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1105–1112, Nov 2007.

[8] G. Milligan and M. Cooper. An examination of procedures for determining the number of clusters in a data set. *Psychometrika*, 50(2):159–179, 1985.

[9] A. Nocaj and U. Brandes. Computing Voronoi treemaps: Faster, simpler, and resolution-independent. *Computer Graphics Forum*, 31(3pt1):855–864, June 2012.

[10] F. V. Paulovich, L. G. Nonato, R. Minghim, and H. Levkowitz. Least square projection: A fast high-precision multidimensional projection technique and its application to document mapping. *IEEE Transactions on Visualization and Computer Graphics*, 14(3):564–575, 2008.

[11] F. V. Paulovich, F. M. B. Toledo, G. P. Telles, R. Minghim, and L. G. Nonato. Semantic wordification of document collections. *Computer Graphics Forum*, 31(3pt3):1145–1153, 2012.

[12] R. Pinho, M. C. F. de Oliveira, and A. de A. Lopes. Incremental board: A grid-based space for visualizing dynamic data sets. In *Proceedings of the 2009 ACM Symposium on Applied Computing*, SAC '09, pages 1757–1764, New York, NY, USA, 2009. ACM.

[13] G. Salton. Developments in automatic text retrieval. *Science*, 253:974–980, 1991.

[14] B. Shneiderman. Tree visualization with tree-maps: 2-d space-filling approach. *ACM Transactions on Graphics*, 11(1):92–99, Jan. 1992.

[15] A. Slingsby, J. Dykes, and J. Wood. Configuring hierarchical layouts to address research questions. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):977–984, Nov 2009.

[16] J. Stasko and E. Zhang. Focus+context display and navigation techniques for enhancing radial, space-filling hierarchy visualizations. In *Information Visualization, 2000. InfoVis 2000. IEEE Symposium on*, pages 57–65, 2000.

[17] G. Strong and M. Gong. Self-sorting map: An efficient algorithm for presenting multimedia data in structured layouts. *IEEE Transactions on Multimedia*, 16(4):1045–1058, June 2014.

[18] S. Tak and A. Cockburn. Enhanced spatial stability with Hilbert and Moore treemaps. *IEEE Transactions on Visualization and Computer Graphics*, 19(1):141–148, 2013.

[19] E. Tejada, R. Minghim, and L. G. Nonato. On improved projection techniques to support visual exploration of multidimensional data sets. *Information Visualization*, 2(4):218–231, 2003.

[20] W. S. Torgeson. Multidimensional scaling of similarity. *Psychometrika*, 30:379–393, 1965.

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

[22] M. Wattenberg. Visualizing the stock market. In *CHI '99 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '99, pages 188–189, New York, NY, USA, 1999. ACM.

[23] M. Wattenberg. A note on space-filling visualizations and space-filling curves. In *Information Visualization, 2005. INFOVIS 2005. IEEE Symposium on*, pages 181–186, Oct 2005.

[24] J. Wood and J. Dykes. Spatially ordered treemaps. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1348–1355, Nov. 2008.