Noname manuscript No.
(will be inserted by the editor)

A Spiral-like Method to Place in the Space (and Interact with) too Many Values

Yannis Tzitzikas · Maria-Evaggelia Papadaki · Manos Chatzakis

the date of receipt and acceptance should be inserted later

Abstract Modern information systems have to support the user in managing, understanding and interacting with, more and more data. Visualization could help users comprehend information more easily and reach conclusions in relative shorter time. However, the bigger the data is, the harder the problem of visualizing it becomes. In this paper we focus on the problem of placing a set of values in the 2D (or 3D) space. We present a novel family of algorithms that produces spiral-like layouts where the biggest values are placed in the centre of the spiral and the smaller ones in the peripheral area, while respecting the relative sizes. The derived layout is suitable not only for the visualization of medium-sized collections of values, but also for collections of values whose sizes follow power-law distribution because it makes evident the bigger values (and their relative size) and it does not leave empty spaces in the peripheral area which is occupied by the majority of the values which are small. Therefore, the produced drawings are both informative and compact. The algorithm has linear time complexity (assuming the values are sorted), very limited main memory requirements, and produces drawings of bounded space, making it appropriate for interactive visualizations, and visual interfaces in general. We showcase the application of the algorithms in various domains and interactive interfaces.

Keywords Visualization, Visual Interfaces

1 Introduction

Visualization is important for understanding data, and this concerns almost every data management task: from schema visualization, to query answer visualization, analytics, data mining, data cleaning, and workload visualization. Moreover, modern information systems should enable the user to understand, and interact with

Yannis Tzitzikas (E-mail: tzitzik@ics.forth.gr) · Maria-Evaggelia Papadaki (E-mail: marpap@ics.forth.gr) · Manos Chatzakis (E-mail: chatzakis@ics.forth.gr)
Institute of Computer Science, FORTH-ICS, GREECE, and Computer Science Department, University of Crete, Greece

larger number of objects than in the past (e.g. the results of summarization, sentiment analysis, dataset discovery and search, etc). However the bigger the data is, the more difficult it is to visualize and understand it. In this paper we focus on a fundamental problem: how to visualize *a set of values in the 2D* (or 3D) space. Quite often we have to visualize a function f, i.e. a set of $(x, f(x))$ pairs where x ranges a finite domain X, and commonly a plot is used where all x values are placed in the x-axis and $f(x)$ are presented as dots in the (x, y) coordinate. If f represents functions like frequency, size, popularity, wealth, etc, then the $(x, f(x))$ pairs are *ordered* in descending order with respect to $f(x)$ and then are plotted. However if X is *big* and $f(x)$ follows *power law* then the resulting plot is not suitable for inspection by humans: its long heavy tail makes almost invisible the first points, i.e. the big elements (let alone the small elements). The commonly used approach for such cases is to make a plot in the *log-log* scale. In such cases the points tend to form a line and it can be approximated with various functions [22]. However such drawings are not convenient for interactive visualization systems: the user cannot easily inspect and interact with each value.

In this paper we introduce a complementary approach that is based on a *circular drawing*. We introduce the algorithm *Concentric Spiral* that yields a plot that is (a) more compact than a plain plot, and (b) more informative in comparison to a log-log plot. It is like "coiling" the big tail of the normal plot. To grasp the idea, Figure 1 shows the populations of the 1000 biggest cities in descending order[1] in five forms: (a) *normal plot*, (b) *log-log plot* and (c) *Concentric Spiral*, (d) *tree-maps* and (e) *sunburst* diagrams. Notice that *tree-maps* and sunburst diagrams do not show each individual value (the small values are collapsed to continuous areas); in general they cannot scale to large numbers while showing each individual element (something that is important for inspection and interaction purposes).

The drawings produced by *Concentric Spiral* can be considered as aesthetically pleasing probably because (i) spiral is a very natural (frequently occurring) shape, and (ii) the drawing does not leave unnecessary blank spaces. Note that the derived spiral-like layout, is not the *Archimedean spiral* (where the distances between the turnings are constant), nor the *logarithmic spiral* (where the distances between the turnings increase in geometric progression); instead the spiral follows the variations of the data, i.e. the width of each "circle" of the spiral depends on the sizes of the objects that are placed in that circle. Another important merit of the algorithm is that it has linear time complexity (if the values are not sorted then we have to sort them first, i.e. $O(n \log n)$ in that case) and very limited main memory requirements, making it appropriate for the placement of too many objects in the space.

We build upon the preliminary ideas for the visualization of the Linked Open Data Cloud [39], and in this paper we refine and extend the algorithm, we introduce several extensions and variations of it that differ on how they manage the empty internal space, we analyze its properties, we show how to combine several such layouts for visualizing more than one functions, we propose extensions allowing the visualization of *millions* of objects, we elaborate on its suitability for visualizing values that follow *power-law*, we discuss efficiency and implementation, and finally we report feedback from users.

[1] Data retrieved by querying the SPARQL endpoint of Wikidata `https://query.wikidata.org/` on May 22, 2019.

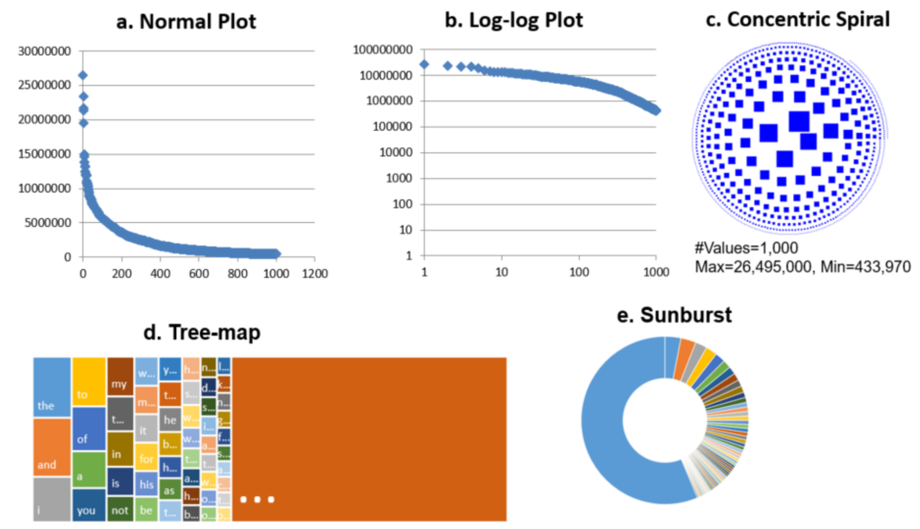


Fig. 1: The populations of the 1000 biggest cities using (a) normal plot, (b) log-log plot, (c) Concentric Spiral, (d) tree-map, (e) sunburst diagram

In a nutshell, the contribution of this paper is that it describes and analyzes a family of layout algorithms for producing compact circular diagrams, with linear time complexity (or $n \log n$ if the values are not sorted), minimal main memory requirements, subject to several variations. We prove that the algorithm yields drawings with no collision, and in case of power laws with exponent greater than one the occupied space is finite (even if the number of elements is infinite). Moreover the paper provides examples of applications from various domains. Overall, this layout can be exploited in a plethora of visualization cases and frameworks either static or interactive. The rest of this paper is organized as follows. Section 2 discusses requirements and background, Section 3 discusses related work, Section 4 presents the main algorithm, while Section 5 presents extensions, and Section 6 presents variations of the algorithm. Section 7 analyzes the space occupied by the produced diagrams, Section 8 demonstrates applications, and finally Section 9 concludes the paper.

2 Requirements and Background

2.1 Requirements

From our experience the last years in designing and building interactive information systems, that visualize values of various kinds as shapes in the 2D space, in environments that allow the user to interact with these shapes, we identified the following desired features, that we shall hereafter call basic requirements: (a) visualize values as *distinct shapes*, each with an *area analogous* to the corresponding value, enabling in this way the *visual comparison* of the values, (b) place bigger values should be placed at the centre for making them evident as well as their *relative size*, (c) collisions should *never* occur, (d) there should be *no big empty spaces*, (e) ability to visualize *thousands* (or even millions) of values very fast, (f) visualize values with *aspect ratio 1* for aiding readability and interaction (e.g. for building virtual 2D/3D VR or AR worlds where each value corresponds to an area/volume with which the user can further *interact* for getting more information or for performing a particular task).

These requirements (or desired features) are quite generic and relevant to various contexts including visualization frameworks (e.g. [19]), faceted search systems [47] (e.g. for visualizing the counters of the available filters), big data exploration [10, 23], visualization of large answers ([6, 3]) OLAP interfaces in general [43, 36], visualization of query workloads [28], data mining [35], data cleaning [20], dataset search [14], new initiatives related to digital libraries [54], and others.

2.2 Background: Power-Laws

Since we are interested in not only small or medium-sized collections, it is worth considering the distribution that a large number of datasets follow[2]. The distributions of a wide variety of physical, biological, and man-made phenomena approximately follow a *power law* over a wide range of magnitudes [15], e.g. word frequencies (Zipf's law), city populations, wealth, the web [1], as well as Linked Data. As regards the latter, such distributions appear at schema level (as studied in [44]), but also on data level (i.e. the sizes of datasets in RDF triples [37, 24]).

The most commonly used approach for visualizing a function that follows power-law, is to use a plot in the log-log scale. Indeed, this is the approach that is followed by papers that reveal and measure power-law distributions (e.g. see [25]) and such log-log plots are offered by software packages that assist revealing such distributions (like [2]). The reason is that in case of power-laws the points tend to form a line and can be approximated with various functions, as it is well elaborated in [22]. Note that the identification of power laws, is also useful for graph drawing, e.g. [29] at first divides the nodes of a graph into power and non-power nodes and then it applies a force-directed placement algorithm that emphasizes the power nodes which results in establishing local neighborhood clusters among power nodes.

Formally, a power-law [15] is a function $f : X \subseteq \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}_{\geq 0}$ of the form: $f(x) = \alpha x^{-\beta}$, where α, β are constants, with $\alpha, \beta \in \mathbb{R}_{\geq 0}$ This means that $f(x)$ can be drawn as a line in the *log-log* scale with a slope equal to $-\beta$. Note that the above definition excludes the value $\beta = 0$, since in this case $f(x)$ would be equal to α for all values of x, i.e. f would be constant. Loosely speaking, uniform distributions can be regarded as a trivial case of *power-law* distributions. Intuitively, the value of β is a measure of the skewness of the distribution, i.e. $\beta = 0$ implies no skewness. A well known example of *power-law* is the Zipf's law that states that given some corpus of natural language utterances, the frequency of any word is inversely proportional to its rank in the frequency table; here $\beta = 1$, i.e. $f(x) = \alpha x^{-1}$, although values greater than 1 (for β) have been measured too.

[2] Roughly, we could consider that small-sized corresponds to a few dozens objects, medium-sized to a few hundreds, and large-sized to thousands and more.

3 Related Work

In the context of Linked Data (see [38] for a survey) several methods have been proposed for the visualization of RDF graphs (i.e. the graph that is composed by a set of RDF triples) e.g. see [9], as well as the surveys [11] and [17]. Visualization is also very important for *data analysis* [4] and there are various tools that offer visualizations for big data analytics [42, 32, 48, 49, 26]. The more related visualizations to our work are described below.

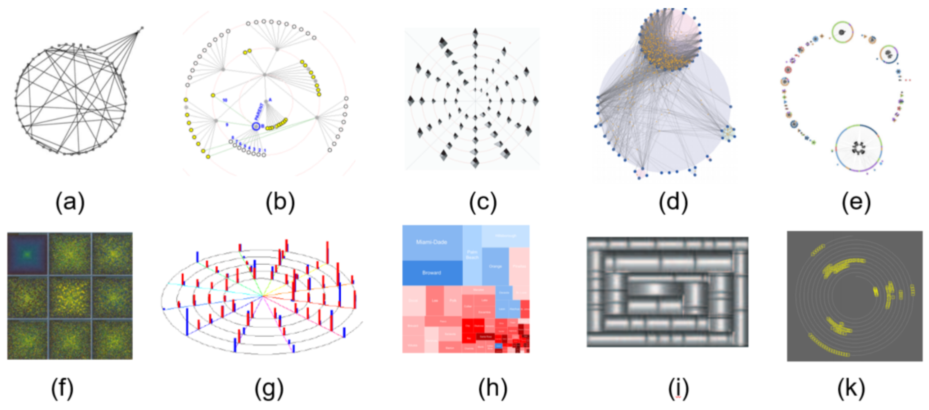


Fig. 2: Related radial, spiral and tree-map visualizations: (a) circular drawings of *graphs* [40], (b) dynamic graphs with *radial layout* [55], (c) exploratory analysis of *cyclic multivariate* data [52], (d) clustered bipartite graphs in *multi-circular style* [30], (e) visual maps for data-intensive ecosystems [34], (f) snake-spiral visualization [33], (g) visualization of serial periodic data [13], (h) *tree-maps* [31], (i) *spiral tree* maps [46], (k) spiral visualization for *time series* [53].

There are several methods for producing circular-like diagrams, see [21] for a survey of *radial methods* for information visualization. To start with, there are algorithms for *circular drawings of graphs*, like [40], where the graph nodes are placed in a circle and emphasis is given on reducing edge crossings, e.g. see Figure 2(a). However in such algorithms the nodes are points, *not shapes*, therefore the notion of *size* is not supported. There are also algorithms for radial layouts, mainly for *trees* [55], e.g. see Figure 2(b), which again do not support different sizes. From the survey [51], the most related visualization seems to be [52], a method for exploring *multivariate data* that may exhibit *periodic* behavior. However such drawings can host values with *small variation* and the number of objects in a circle is *fixed*, e.g. see Figure 2(c), therefore they cannot host too many values, nor values that vary a lot. There are also spiral displays enhanced with coding and interaction like that in [45], however in that work every "concentric zone" has the same width, therefore one cannot visualize too many values especially if they vary a lot, since in that case the produced diagram would waste a lot of space unless colors are used to signify size (but in that the correspondence between area and value is lost). These methods seem to be appropriate to identify periodic trends, not for visualizing values with big variations For clustered bi-partite graphs, [30] proposes a multi-circular style, e.g. see Figure 2(d), while for maps of data-intensive software ecosystems (by considering also clustering purposes) [34] proposes layout algorithms based on concentric circles, e.g. see Figure 2(e). Such layouts usually contain *only a few* circles and there are rather *big empty areas* (in comparison to the algorithms presented in this paper). There are *pixel-oriented* visualization techniques for exploring large data bases, e.g. the *snake-spiral* technique [33], however that technique does not respect the relative sizes (all values are visualized with the same size), e.g. see Figure 2(f). The same is true for the visualization method for serial periodic data [13]: it does not respect the sizes, it adopts the Archimedean spiral, e.g. see Figure 2(g). With respect to *tree-maps* [31], e.g. see Figure 2(h), and squarified tree maps [12], which are usually applied for the visualization of hierarchical information structures, they do respect the sizes, they fill entirely the available rectangular region, however it is *difficult to achieve a small aspect ratio* (the ideal is one since regions with a small aspect ratio, i.e. fat objects, are easier to perceive); and it also difficult to preserve some sense of the *ordering* of the input data. Even though, several improvements of tree-map representations have been proposed, like circle packing [50], Squarified Treemaps [12], Quantum Treemaps [8], treemaps with bounded aspect ratio [18], Stable Treemaps [41] Bubble Treemaps [27], Voronoi Treemaps [7] or GosperMap [5], see Figure 3, a few limitations among them are that (i) relative ordering may be lost [50, 12], (ii) changes in the data set can cause dramatic discontinuous changes in the layouts [12] (iii) efficiency may be decreased [41, 27] and (iv) in some cases no guarantees on the aspect ratio is given [7]. For instance, Bubble Treemaps [27] (Figure 3.o) achieve aspect ratio 1 (as it uses circles), however it does not respect the absolute ordering and its time complexity is quadratic. There are algorithms that produce *spiral treemap layouts* for visualizing changes of *hierarchical data*, e.g. [46], however these algorithms seem applicable only on *small number* of objects, e.g. see Figure 2(i), and we have not seen them being applied on real data. A spiral visualization for *time series* is proposed in [53], however it is not appropriate for values that variate a lot, e.g. see Figure 2(k).

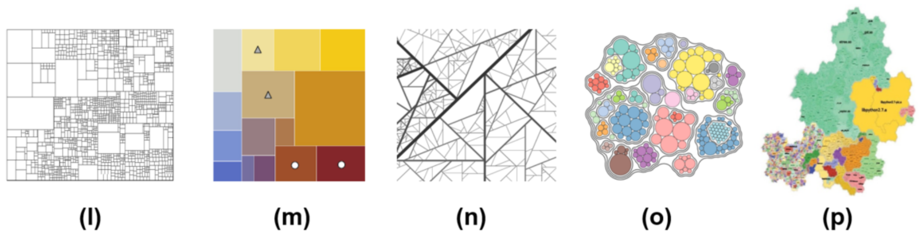


Fig. 3: Tree-map visualizations: (l) Squarified Treemap [12], (m) Stable Treemap [41], (n) Voronoi Treemap [7], (o) Bubble-Treemap [27], (p) Gospermap [5].

As regards, *circle packing*, the method described in [50] groups values in circles, while we use the sectors of a circle. That method seems to be suitable for cases of hierarchical data sets where users are more interested in the relationships between them. In contrary, our algorithms are also appropriate for sets of values that vary a lot (i.e. power-law) emphasizing the biggest/most important values of a data set. We also preserve the ordering, while that work does not. In general circle packing aims at minimizing the area, but the resulting diagrams do not respect the rank of the objects, i.e. the exact rank of an object is not at all obvious, nor which is its predecessor and successor in the rank. The corresponding decision problems (i.e. weather a given set of circles can be packed in a particular bin) are NP-hard, however various approximate algorithms exist. From that point of view, we could say that our algorithm could be considered as an approximate algorithm for packing squares in a circle that fully preserves the ranking of the objects and has linear complexity (if the values are sorted).

Overall, no radial, spiral or tree-map layout can tackle all requirements that we listed earlier (in Section 2.1). In brief, the radial layouts do not show/respect sizes and their ability to show many values is limited. The tree-map algorithms although they leverage all the available space, it is difficult to achieve a good aspect ratio and at the same time respect the ranking of the objects. The algorithm that we will present gives always aspect ratio 1 (square shapes), it fully respects the ranking of objects, and it tries not to waste space, while providing an aesthetically pleasing (rounded) layout. From a circle-packing point of view, we could say that our algorithm can be considered as an approximate algorithm for packing squares in a circle that fully preserves the ranking of the objects and has linear complexity (if the values are sorted).

4 The Algorithm Concentric Spiral

Let f be the function that we want to visualize, i.e. a set of $(x, f(x))$ pairs where x ranges a finite domain X, and let Y denote the range of the function, i.e. $Y = \{ f(x_i) \mid x_i \in X\}$ (we shall use Y_{desc} to denote Y sorted in descending order). Each $y \in Y$ will be visualized by a *square shape*, specifically with a square having side length $\sqrt{y}$, implying that its *area* equals y. We consider square shapes, since for a given perimeter, square is the rectangular shape with the maximum area[3].

Based on the requirements listed in §2.1, we present a new 2D placement algorithm that we call *Concentric Spiral.* The exact steps of the algorithm are shown in Alg. 1. The algorithm takes as input a series of numbers in descending order (i.e. Y_{desc}) and a number ($ringGap$) that specifies the desired minimum gap between the rings. The idea of the placement is the following. Each number of the input is represented as a square shape. The shapes are placed in *concentric rings.* The radius of the first, *smallest*, ring is the size of the *biggest number.* The placement of the subsequent shapes is done as follows. We compute a chord that ensures no collisions based on the sizes of the current (to be placed) and the previous shape. We set this to be equal to the side of the current shape plus the size of the previous shape. Based on that chord we compute the corresponding angle, and we place the new shape at the corresponding spot of the ring. This is illustrated in Figure 4. Suppose that A is the centre of the first shape, and B is the centre of the next shape that we want to place. The figure illustrates why the sought *angle* is: $\theta = 2\arcsin(\frac{chord}{2 \cdot radius})$ for a given chord denoted by χ in the figure, and radius ρ. This is what, in Alg. 1, $degreesOfChor(\chi, \rho)$ computes.

[3] Arbitrary shapes could be supported by considering their minimum bounding squares.

Algorithm 1 Concentric Spiral layout algorithm

```
Require: A list of $K$ shapes (each characterized by a number $len$ ) ordered in descending
    order wrt their size
Require: A constant $ringGap$ for setting the desired gap the between the rings
Ensure: A cyclic layout of the input shapes, with no collisions and no unnecessary empty
    spaces in the peripheral areas
 1: $i \leftarrow 0$; $\theta \leftarrow 0$                          ▷ counter and angle set to zero
 2: $\rho = slen_{cur} = slen_{max} = shapes[i].len$    ▷ initial radius equal to the side of the biggest
    shape
 3: while $i < K$ do                                     ▷ for each shape
 4:     $\chi \leftarrow shapes[i].len$     ▷ chord is set equal to the size of shape to be drawn (for placing
    the 1st shape above the x-axis)
 5:     if $i > 0$ then                                  ▷ if not the first shape
 6:         $\chi \leftarrow \chi + slen_{cur}$              ▷ adding to the chord the size of the prev. shape
 7:     $\theta_{incr} \leftarrow degreesOfChord(\chi, \rho)$     ▷ degrees corresponding to chord with size $\chi$ in a
    cycle with radius $\rho$
 8:     $\theta \leftarrow \theta + \theta_{incr}$                            ▷ increasing the current angle
 9:     if $\theta \geq 2 * \pi$ then                    ▷ if true then a new ring should be started
10:         $\rho \leftarrow \rho + slen_{max} + ringGap$  ▷ to avoid collisions with shapes of the previous ring
11:         $slen_{max} \leftarrow shapes[i].len$          ▷ this is the max shape size in this new ring
12:         $\theta \leftarrow \theta - 2 * \pi$                         ▷ for ranging $0..2\pi$
13:         $\chi \leftarrow shapes[i].len$                        ▷ the size of this shape
14:         $\theta_{incr} \leftarrow degreesOfChord(\chi, \rho)$
15:         $\theta \leftarrow \theta + (\theta_{incr}/2)$                    ▷ Since this is a new ring
16:     $x \leftarrow CanvasCenterX + \rho * \cos(\theta)$  ▷ the $x$ coord. of the point where the centre of the
    new shape should be placed
17:     $y \leftarrow CanvasCenterY + \rho * \sin(\theta)$  ▷ the $y$ coord. of the point where the centre of the
    new shape should be placed
18:     $shape_{cur} \leftarrow shapes[i++]$                        ▷ gets the next shape
19:     $drawShapeAtCenter(x, y, shape_{cur})$          ▷ draws the shape with center at $x, y$
20:     $slen_{cur} \leftarrow shape_{cur}.len$
```

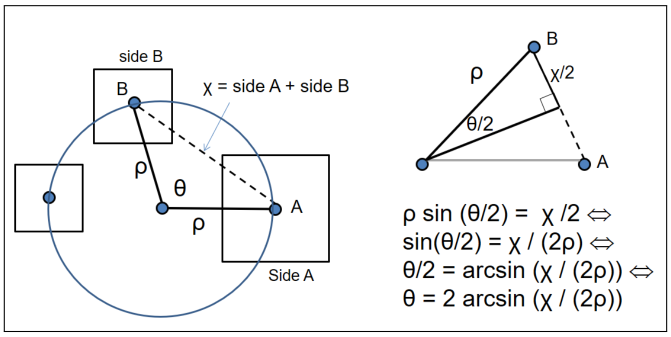


Fig. 4: The angle that determines the position of the next shape

The algorithm continues to place the rest of the shapes in that manner and just before we reach 2π, we start the *next bigger ring* whose radius is the radius of the previous ring increased by the size of the biggest shape in the previous ring, plus a number accounting for the extra empty space that we want to leave between the rings (this is the input *ringGap* of the algorithm). This is illustrated in Figure 5. This method ensures no collisions between the shapes (as we shall prove). It is not hard to see that as the *shapes get smaller*, the concentric rings

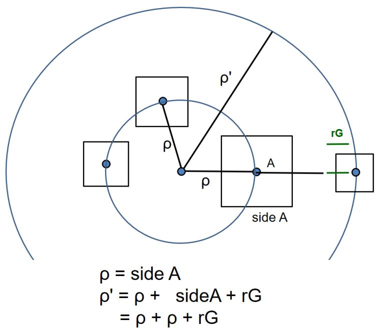


Fig. 5: From the first ring to the second ring

become *denser* avoiding in this way unnecessary empty spaces. In addition, as the rings become denser the empty spaces between the shapes decrease as well. Notice that Alg. 1 refers to the *len* (length) of shapes. This corresponds to the *side length* of the squares. Even if it corresponded to their *area*, this distinction would not be important for power-laws: if *area* follows *power-law* with $\beta = \beta_{area}$, then the *side* also follows *power-law* with $\beta = \beta_{area}/2$.

Time Complexity. It is straightforward that Alg. 1 has $O(n)$ time complexity (n is the number of shapes). Experimental measurements are given in subsequent sections (§8.1).

Prop. 1 Alg. 1 (a) always terminates, and (b) it yields a drawing with no collision.

Proof: (a) holds since the algorithm just consumes the input by scanning it once. (b) holds because of the way we compute chords (sum of the sides of the neighboring shapes) and due to the distance of the successive rings. Specifically, the maximum distance between the centres of two squares that intersect, is the half of the sum of their diameters (if bigger then it is impossible to intersect). Specifically, if the first square has side a and the second has side b, then this distance is $\frac{1}{2}(a\sqrt{2} + b\sqrt{2}) = \frac{a+b}{\sqrt{2}}$. Since the centres of two successively placed (intra-ring) squares by the algorithm is $a + b$ (since it is the sum of the length of the previous and to be placed shape), no collision is possible between squares of the same ring. Since the distance between two successive rings is the maximum size of the shapes in the previous ring, say a, plus the size of the first shape of the new ring, say b, it is guaranteed that no collisions are possible between shapes of neighboring rings (since $a + b > \frac{a+b}{\sqrt{2}}$). ⋄

4.1 Concentric Spiral in Small Sets of Values

To understand the behavior of the algorithm for small numbers of n, Figure 6 shows the visualization of only 100 synthetically generated values where the first (maximum) value is the number 100. In (a) the side of the squares is reduced by 7%, in (b) the side is reduced by 21%, in (c) the side follows a *power-law* with $\beta = 1$, in (d) the size is reduced by a random percentage in the range [2,30). The key points are: (a) if the values decrease smoothly, we get a circular drawing, and (b) the more skewed the distribution is, the more compact the drawing becomes for being able to host more values.

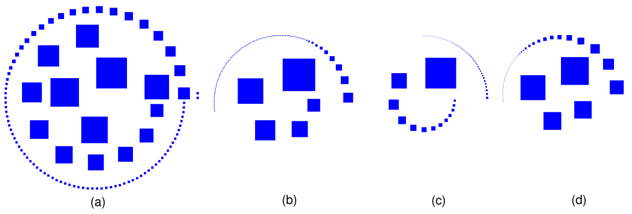


Fig. 6: (a): side reduction by 7%, (b): side reduction by 21%, (c): the side follows a *power-law* with $\beta = 1$, (d): side reduction by a random percentage in range [2,30)%

4.2 Configuration Parameters and their Impact

There are some parameters that affect the way the layout looks like, and can be exploited for getting a layout that facilitates others tasks that we may like to carry out.

The first is the *scaling*, i.e. the min and max size of the shapes in the visualization. This choice depends on the resolution of the canvas, but also on other aspects. For instance, if we want every shape to be clearly visible we may define as min size not only 1 pixel but more pixels. Although any scaling of the range $[Y_n, Y_1]$ to $[ShapeSize_{min}, ShapeSize_{max}]$ can be considered as a valid scaling, if $ShapeSize_{min}$ is more than 1 pixel, then the illustration of the relative sizes will not be very accurate. For instance, Figure 7 shows three different visualizations of the same dataset (the population of the 1000 biggest cities): the first uses for $[ShapeSize_{min}, ShapeSize_{max}]$ the interval $[1, 40]$, the second the interval $[5, 40]$, and the last the interval $[10, 40]$. We will revisit the notion of scaling, for the case of very big datasets, later in §7.1.

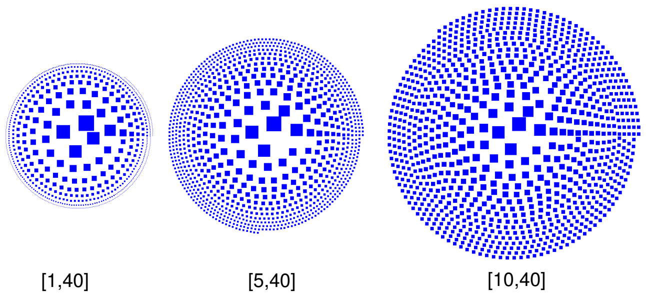


Fig. 7: The effect of using different min shape sizes in the visualization of the 1000 biggest cities

The second parameter is the parameter *ring gap* (*ringGap*). If we decide to reduce it as the shapes become smaller, then that would harm the readability of the diagram and the visibility of the outer rings. For this reason in the algorithms we keep it stable from the first up to the last ring.

Axes. Normal and log-log plots have an X-axis and a Y-axis: the values at the right side of X allow us to see at a glance *how many* points are visualized, while the values in the upper (resp. lower) parts of Y allow to see the *biggest* (resp. *smallest* value(s) of $f(x)$. The minimum enrichment of a Concentric Spiral-based drawing for conveying this information, is to enrich it with three values: the numbers of values, the maximum value and the minimum number. For example, the diagram in Figure 1 contains:

```
#Values=1,000
Max=26,495,000, Min=433,970
```

meaning that we see 1,000 values (city populations), where the biggest is 26,495,000, and the minimum is 433,970. For making evident at a glance how many values are visualized (and how many would fall in each order of magnitude of the X-axis of a traditional log-log plot), we enrich the diagram with *concentric axes* drawn in the following manner: after placing 10^i shapes in the drawing, we plot a cycle with thickness analogous to i. For example, Figure 8(a-b) shows the axis-enriched drawing of the 1000 biggest cities, as well as the drawing of 23,113 values (frequency of words from Shakespeare). In one glance we can understand that the first has 3 orders of magnitude, and the second 4. Analogously, we can use concentric axes for showing the information related to the Y axis, specifically by using concentric axes (with color different than the those for the X axis) at the radii where the values y_i change magnitude.

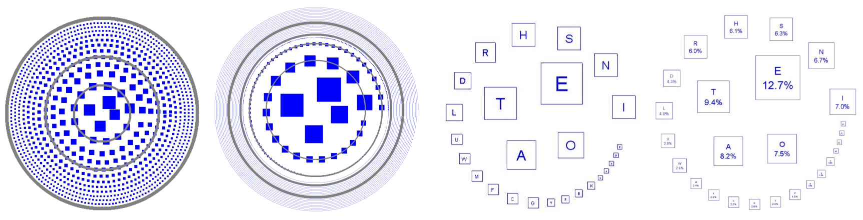


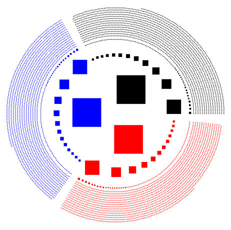
Fig. 8: (a): visible axes for the 1000 biggest cities, (b): a dataset comprising 23,113 values, (c and d): the frequencies of Latin characters

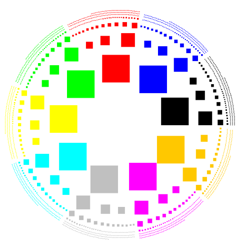
Labels. As regards labels, various options are supported for what to show inside the square: (i) the value y_i, (ii) the value/label x_i, (iii) the rank of y_i in Y_{desc}, i.e. i, (iv) any combination of them. The labels are enclosed in the squares, enabling in this way to inspect more clearly the top-K (say 5-10) elements without overloading the diagram. Note that the labels do not affect the size of the squares, since the font size of the labels adapts to the size of the squares. For example Figure 8(c-d) shows the frequency of the 26 letters in English, without labels and with labels (x_i and y_i)

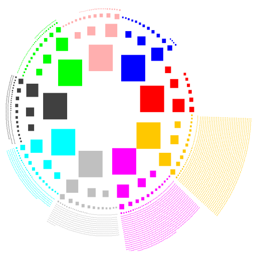
5 Extensions of Concentric Spiral

5.1 Pie chart-like Extension

Apart from visualizing one set of values Y_{desc} we may want to visualize m sets (for a small m value), i.e. a family of sets $Y_1, \ldots, Y_m$, each being a set of values ordered in descending order. This can be achieved by extending Alg. 1 so that instead of using all angles in the range $[0, 2\pi]$, to take as input the desired angle range. Then the range $[0, 2\pi]$ can be partitioned into m angle ranges, one for each of the m sets. We can use then Concentric Spiral for placing the shapes of each set at the corresponding slice defined by the angle interval $[\theta_{min}, \theta_{max}]$: we just have to use θ_{min} to determine where to place the first shape and θ_{max} to determine when to change ring (the exact steps of the extended algorithm are given in §6.5). For example, Figure 9(a) shows the visualization of 3 sets each having 2,000 values, (b) shows 9 sets each having 200 values, and (c) shows 9 sets each having unequal number of values (from 10 to 2,000). It is not hard to see that we can use this angle-restricted extension not with the entire range $[0, 2\pi]$ but with an angle range based on our preference. For example we can use the angle range $[0, \pi]$ for getting a visualization that resembles the seats of a parliament as illustrated in Figure 9(bottom) that shows 8 slices in $[0, \pi]$.


(a) 3 sets each having 2,000 values


(b) 9 sets each having 200 values


(c) 9 sets the smallest has 10 values and the biggest has 2,000 values

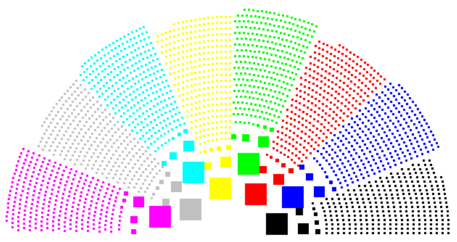


Fig. 9: Pie-chart like extension of Concentric Spiral

Overall, the pie chart-like extension of Concentric Spiral can be useful for (a) *visually inspecting/comparing more than one datasets*, and (b) for *clustering* the values of a dataset according to one criterion.

5.2 On Visualizing more than one Function

So far we have focused on how to visualize one function f, specifically its value set Y, ordered in descending order, i.e. Y_{desc}. Now suppose that we have another function g over X, i.e. $g : X \rightarrow Z$. Here we describe how we can visualize both $f(x)$ and $g(x)$. If the range of g is small (i.e. $|Z|$ is small), say $2 \leq |Z| \leq 5$, then apart from the pie char-like extension described in §5.1, we can extend the previous visualization and make clear the $g(x)$ by drawing the shape corresponding to $f(x)$ with a *different shape type*, e.g. square, triangle, oval, etc., or a *different color*. Both of them (i.e. shapes and colors) can be combined for visualizing *three* functions in total: f by the size of the shape, g by the shape type, and h by the color. As we shall see in Section 8, we can also employ *3D* and use the third dimension for visualizing an additional function, i.e. 4 functions in total. In that case, since the *volume* (and not the area) of the shape should correspond to $f(x)$, if we employ cubes then the edge length should be set to $\sqrt[3]{f(x)}$ so that the volume of the cube is equal to $f(x)$. Overall, by considering also the slice-based extensions (of §5.1), we can reach 5 functions in total: one "dominant" i.e. the one that follows the power-law distribution, and 4 whose range is small.

6 Variations of Concentric Spiral

Below we introduce three variations of the core algorithm, all having linear time complexity, that can reduce the internal free space at the center of the drawing, if that is required. Specifically Section 6.1 presents a *ring* variation, Section 6.2 presents a *theater-like* variation, and Section 6.3 presents a variation that *mixes* the above two. Then Section 6.4 *compares* these variations, and finally Section 6.5 presents the *general algorithm* that can produce *any* of these variations.

6.1 Concentric Spiral$_{Ring}$

The basic idea is the following: Instead of increasing the radius after the completion of the first circle, based on the largest shape placed in that circle (at line 10 of Alg. 1), we can increase the radius just by the *ringGap* for getting a denser layout. However we have to check whether that position is free for avoiding collisions with the shapes that have already been placed. Note that this is not required in Concentric Spiral because the radius is increased based on the largest shape placed in that circle. We can check whether a position is free before placing a shape (at line 28 of Alg. 1) by checking the colors of the canvas pixels. Since shapes are placed in descending order of size, it is enough to check the color only of the four corners of the candidate position for the placement. This algorithm produces layouts with less empty space between the core of big shapes and the periphery of small shapes. For the case of 5,000 shapes, Figure 10(a) shows the layout produced by the Concentric Spiral, while Figure 10(b) shows the layout produced by the Concentric Spiral$_{Ring}$.

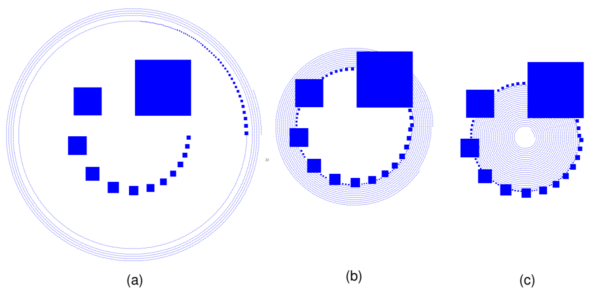


Fig. 10: (a): Concentric Spiral, (b): Concentric Spiral$_{Ring}$ (c): Concentric Spiral$_{Theater}$

6.2 Concentric Spiral$_{Theater}$

If we would like to leverage the *internal space* between the big shapes then we could use a variant of Concentric Spiral that we call Concentric Spiral$_{Theater}$: after the completion of the ring (that contains the biggest shapes), instead of increasing the radius (at line 10 of Alg. 1), as it is done in Concentric Spiral and Concentric Spiral$_{Ring}$, we *decrease* it by the width of the *ringGap*. In this way the biggest datasets form the big circle and the small ones then fill the internal space. This algorithm derives a more dense drawing resembling an ancient Greek theater, as shown in Figure 10(c). However it is not guaranteed that the internal space will be enough for hosting all small shapes, an issue that is tackled by the algorithm presented next, in §6.3.

6.3 Mixing Concentric Spiral$_{Ring}$ and Concentric Spiral$_{Theater}$

Concentric Spiral$_{Ring}$ and Concentric Spiral$_{Theater}$ can be combined: we can start as in Concentric Spiral$_{Theater}$ and if the internal space is completely filled before placing all shapes, we can continue by placing the remaining shapes in the peripheral area, as in the Concentric Spiral$_{Ring}$. This is shown in the diagrams of the third column of Figure 11 that visualizes 10,000 values. Alternatively one could use the Concentric Spiral$_{Theater}$ with a bigger initial radius but that would require estimating which radius is adequate. Instead, by mixing the two algorithms, there is no need for any estimation; a single pass is enough for drawing all shapes.

6.4 Concentric Spiral vs Concentric Spiral$_{Ring}$ vs Concentric Spiral$_{Theater}$

To better understand how the skewness of the distribution of data affects the produced layouts, Figure 11 shows Concentric Spiral, Concentric Spiral$_{Ring}$ and Concentric Spiral$_{Theater}$ over three versions of a dataset with 10,000 values. Each version has the same maximum value, however each version has different reduction rate, specifically in the 1st row $\beta = 1/2$, in the 2nd $\beta = 1$, and in the third row $\beta = 2$. The same scale has been used for all 12 diagrams. We observe that the diagrams of Concentric Spiral$_{Ring}$ and Concentric Spiral$_{Theater}$ occupy less

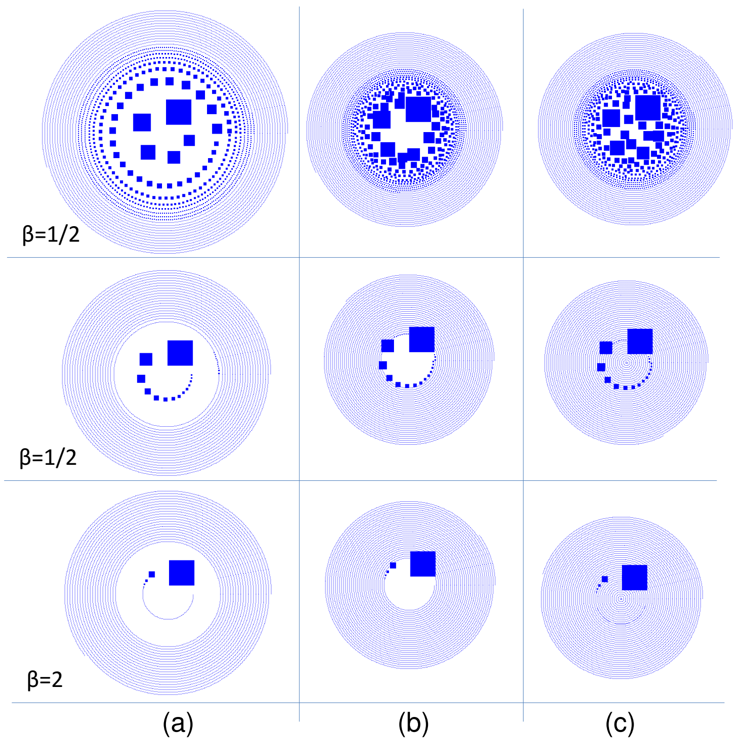


Fig. 11: (a): Concentric Spiral, (b): Concentric Spiral$_{Ring}$ (c): Concentric Spiral$_{Theater}$ over 10,000 values.

space than those of Concentric Spiral, however Concentric Spiral better reveals the relative sizes of the middle-sized elements.

6.5 The Generalized Algorithm

The generalized algorithm that supports *all* previously described extensions and variations is given in Alg. 2. Apart from the list of values in decreasing order, it takes as input a parameter *mode* (ranging *ExpandSpiral*, *ExpandRing*, and *Shrink*) that determines which layout (either Concentric Spiral, Concentric Spiral$_{Ring}$, or Concentric Spiral$_{Theater}$ respectively) should be produced. In addition it takes as input an angle range $[\theta_{min}, \theta_{max}]$ for producing the pie chart-based extension. Note however that the check at line 27 of the algorithm is redundant for the Concentric Spiral (since the condition will always be true). The small size of the algorithm facilitates its implementation in the existing visualization frameworks.

7 Analyzing the Occupied Space

Let's further analyze the space occupied by the diagrams produced by Concentric Spiral. The area of the ring defined by the radii R_2 and R_1 (where $R_2 > R_1$) is

Algorithm 2 Concentric Spiral layout algorithm with Variations

```
Require: A list of $K$ shapes (each characterized by a number $len$ ) ordered in descending
    order wrt their size
Require: An angle range defined by an interval $[\theta_{min}, \theta_{max}]$.
Require: A mode from the following list: $ExpandSpiral, ExpandRing, Shrink$.
Ensure: A cyclic layout, within a circular sector $[\theta_{min}, \theta_{max}]$, with no collisions and no
    unnecessary empty spaces in the peripheral areas
 1: $i \leftarrow 0$; $\theta \leftarrow \theta_{min}$                    ▷ Counter and angle set to zero
 2: $\rho = slen_{cur} = slen_{max} = shapes[i].len$    ▷ initial radius equal to the side of the biggest
    shape
 3: while $i < K$ do                                           ▷ for each shape
 4:     $\chi \leftarrow shapes[i].len$ ▷ chord set equal to the size of shape to be drawn (for being above
    the x-axis)
 5:     if $i > 0$ then                                        ▷ if not the first shape
 6:         $\chi \leftarrow \chi + slen_{cur}$          ▷ adding to the chord the size of the prev. shape
 7:     $\theta_{incr} \leftarrow degreesOfChord(\chi, \rho)$            ▷ degrees of chord with size $\chi$
 8:     $\theta \leftarrow \theta + \theta_{incr}$
 9:     if $\theta \geq \theta_{max}$ then                ▷ if true then a new ring should be started
10:         if $Mode = ExpandSpiral$ then
11:             $\rho \leftarrow \rho + (slen_{max} + shapes[i].len)/\sqrt{(2)} + ringGap$  ▷ to avoid collisions with
    shapes of the previous ring
12:         else if $Mode = ExpandRing$ then
13:             $\rho \leftarrow \rho + ringGap$   ▷ does not guarantee avoidance of collisions with shapes of
    the previous ring
14:         else if $Mode = Shrink$ then
15:             $\rho \leftarrow \rho - ringGap$         ▷ for filling the free space left by the first big shapes
16:         if $\rho < slen_{cur}$ then   ▷ We are in shrink mode and we have reached the center of
    all circles
17:             $Mode = ExpandSpiral$               ▷ We change mode from Shrink to Expand
18:             $\rho \leftarrow shapes[0].len$ ▷ the one at the beginning of the drawing, i.e. the size of the
    biggest shape
19:         $slen_{max} \leftarrow shapes[i].len$          ▷ this is the max shape size in this new ring
20:         $\theta \leftarrow \theta_{min}$                              ▷ for ranging the angle interval
21:         $\chi \leftarrow shapes[i].len$                                 ▷ the size of this shape
22:         $\theta_{incr} \leftarrow degreesOfChord(\chi, \rho)$
23:         $\theta \leftarrow \theta + (\theta_{incr}/2)$
24:     $x \leftarrow CanvasCenterX + \rho * \cos(\theta)$
25:     $y \leftarrow CanvasCenterY + \rho * \sin(\theta)$
26:     $shape_{cur} \leftarrow shapes[i++]$                                ▷ gets the next shape
27:     if $isEmpty(x, y, slen_{cur})$ then      ▷ The space is free (required only from RING and
    SHRINK)
28:         $drawShapeAtCenter(x, y, shape_{cur})$        ▷ draws the shape with center at $x, y$
29:         $slen_{cur} \leftarrow shape_{cur}.len$
30:     else                                                   ▷ The space is not free
31:         $i - -$                       ▷ For trying finding free space in the next iteration
```

$A_{R_1,R_2} = \pi(R_2^2 - R_1^2)$. Since the values are decreasing, also $R_2 - R_1$ and consequently A_{R_1,R_2}, are decreasing too. The more values we have to visualize, the bigger the radius becomes, and the more the occupation of space tends to be collinear, as shown in Figure 12, occupying the minimum space. Therefore the percentage of empty/filled area is reducing.

Let's now focus on data that follows power-law. Let *Filled Area*, for short FA be the area occupied by the shapes only. Let $\langle Y_1, \ldots, Y_n \rangle$ be the n values to be visualized. If this series follows *power-law* then we can write $Y_i = Y_1/i^{\beta}$ (using the notations of §2.2: $f(i) = \alpha/i^{\beta}$, $f(1) = \alpha = Y_1$). The total area of shaped (FA) is

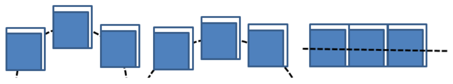


Fig. 12: Occupied space as the number of values increases

given by $FA = \sum_{i=1}^{n} \frac{Y_1}{i^{\beta}} = Y_1 \sum_{i=1}^{n} \frac{1}{i^{\beta}}$ Note that the p-series $\sum_{i=1}^{\infty} \frac{1}{i^p}$, converges for all $p > 1$ and diverges for all $p \leq 1$ (note that if $p > 1$ then the sum of the p-series is $\zeta(p)$, i.e., the Riemann zeta function evaluated at p). The key point is that for $\beta > 1$ the area of the visualized shapes is *bounded*, independently of how big n is (even if n was infinite).

Let's now estimate the empty area, denoted by EA, i.e. the area that falls within the outer circle of the diagram but is not occupied by any shape of the diagram.

Prop. 2 The area that falls within the outer circle of the diagram but is not occupied by any shape of the diagram, is less than 3 times the filled area by the shapes (FA).

Proof: Let Z be the number of the concentric rings of the diagram (obviously $Z \leq n$). Let EA_i be the empty space within ring i. Since certainly at least one shape is placed in each ring, e.g. in the first ring it will occupy one of the four quadrants, it holds that $EA_i \leq 3FA_i$, i.e. EA_i cannot be 3 times more than the filled part of ring i. Therefore, $EA \leq 3FA$. $\diamond$

Obviously this bound is very pessimistic. However the point is that if FA is bounded, then the same holds for EA. To summarize we have seen that if Y follows *power-law* and $\beta > 1$ then the entire diagram is finitely bounded in normal scale (not log-log scale) even if the number of n is infinite. In the proof of Prop. 1 (in Section 4) we have seen that $\frac{a+b}{\sqrt{2}}$ is the minimum chord that guarantees no collisions. Concentric Spiral uses $a + b$, i.e. a larger distance. The reason is that with $a + b$ the diagram looks better (to the authors), however the theoretically minimum chord can be used as well. As regards the three variations of the algorithm, Concentric Spiral$_{Theater}$ with the minimum chord is the variation of the algorithm that has the minimal empty space between the shapes.

7.1 Very Large Datasets

In the previous drawings, we have used minimum size 1 (i.e. 1 pixel) for making each point visible. However if the dataset is too big, then the canvas could become extremely big. Moreover, this causes problems with the bounding of the drawing: the min size 1 does not allow us to achieve a drawing of bounded size, since even if the values are extremely small they always occupy at least one pixel in the drawing space.

If we want to tackle this problem we can use 0 as min size, i.e. 0 pixels. However, whenever the algorithm reaches a point whose shape should have 0 as rounded integer value, it stops drawing points, and enters into a different mode. Specifically let $\langle Y_1, \ldots, Y_n \rangle$ be the values to be visualized and let Y_k $(1 < k \leq n)$ be the first value whose re-scaled rounding is 0. In that case the values $\langle Y_k, \ldots, Y_n \rangle$ are visualized differently: a *filled ring* is added with area equal to $S = \sum_{i=k}^{n} Y_i$ (after normalizing S in consistency with the normalization of the max value Y_1). This approach allows producing a more compact drawing, *compliant with the theoretical bounds*, that also preserves the relative sizes of all the visualized values.

The size of the ring is computed as follows: Let $R_{k'}$ be the radius when Y_k was about to be drawn. The algorithm in that case stops drawing shapes and instead fills a ring starting from the radius $R_k = R_{k'} + slen_{max} + ringGap$ up to the radius needed for having area equal to (the normalized) S. The outer radius R_x of that ring is computed as follows $\pi(R_x^2 - R_k^2) = S \Leftrightarrow R_x = \sqrt{\frac{S+\pi R_k^2}{\pi}} = \sqrt{R_k^2 + \frac{S}{\pi}}$. To achieve the above, Algorithm 2 just needs one line at the beginning (line 4) that checks if $\chi = 0$, i.e. if the integer rounded size is 0 pixels (that is less than 0.5 pixels), and if yes, it calls a method with the current index i and the radius from which the ring should start, i.e. the exact line is
"if ($\chi = 0$) then $fillRing(i, \rho + slen_{max} + ringGap)$; break". An excerpt from the visualization of *10 millions values* in this way is shown in Figure 13 (it is a synthetic dataset with max value 4000, min value 10, 20% percentage reduction between two consecutive values, and scaling interval $[0, 50]$). The axes allow to understand that we have 7 orders of magnitude (the axes in the filled ring are distributed uniformly in $[R_k, R_x]$ for aiding readability). Although our algorithm can show more small objects in comparison to tree-map and sunburst (as we have seen in Figure 1), if rings are adopted, then zoom-in on the rings cannot offer more information. To "analyze" a ring to individual shapes we can either (a) increase the size of the canvas, (b) click on the ring the visualize the subset of objects that belong to that ring (and that could be done recursively as many times as the user wishes to), or (c) enable the visualization of any subset of the values of the ring specified by their rank (e.g. all values with rank i up to rank j). As regards option (a), i.e. the usage of a very large canvas that does not require using any ring, the user can still locate the desired object(s) through *search*; this functionality (i.e. search and locate) is supported by our implementations that are described in Section 8.3, i.e. by the produced drawing in SVG, as well as by the interactive 3D system. All these are aligned with [16] that stresses that for visualizing massive data it is important to support selection, zooming, and filtering, and exploiting the rotation capabilities of 3D (examples of 3D are given in in Section 8.3). Finally, we have to note that if a very large canvas is selected to be used, even though our algorithm that produces the coordinates of the objects will run fast, the visualization of the produced raster image could require a lot of memory to load or transfer; such cases can be handled by loading the data/image to a GIS and we have already tested that scenario using GeoServer[4], moreover one can exploit methods for visualizing images of very big resolution like those derived by modern scanning approaches like the one described in [56].

8 Application

Concentric Spiral and its variations can be applied in a plethora of cases. We have focused mainly on the layout, not on other aspects, i.e. on colors, interactivity, etc, since the latter depend on the application context. Section 8.1 discusses *efficiency*, Section 8.2 provides various indicative cases, and Section 8.3 describes current

[4] `http://geoserver.org/`

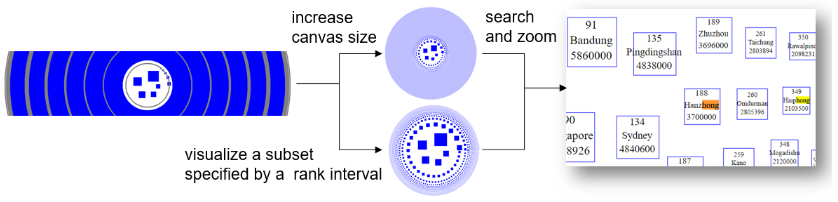


Fig. 13: 10^7 values with filled rings for the very small values, and interactive options

implementations. Finally, Section 8.4 presents some preliminary feedback from users.

8.1 Efficiency

We have implemented the algorithms using Java 1.8.0_51. In our experiments we measure the time required by the layout algorithm to fill a $N \times N$ java array of pixels each having a color (we did not count the time of the windowing system, e.g. for creating the JFrame). With a laptop with rather low computational power (Core i5-3320M CPU, 4 GB RAM, running Windows 10), Concentric Spiral needs only 30-40 milliseconds to compute all pixels of a 1000x1000 canvas for 100,000 (filled with blue) shapes. The other two variations are more expensive since they have to check for free spaces, therefore the elapsed time is more. For all the diagrams shown in this paper, these two variations were never more than 3 times slower than Concentric Spiral. In general, all visualizations shown in this paper (except the one with the millions of shapes) require *less than 120 milliseconds* to be computed. For datasets with *millions* of objects, roughly it takes 15 secs per million of objects (where each object is visualized). If however the scaling interval includes 0, then with the method presented in §7.1 the visualization of 10 million objects takes 17 secs.

8.2 Examples

Here we provide various examples.
Word Frequencies. Figure 14 shows the frequency of 23,113 words in Shakespeare[5] in the three plots (normal, log-log and Concentric Spiral).
Cities Population. Figure 15 shows the populations of the 1000 biggest cities in 3D, where each color represents a different continent, and the volume of each cube corresponds to the population of the corresponding city. The figure includes visualization with the Concentric Spiral layout, and with the pie chart-like extension for clustering (using the same angle range for each slice, although any range can be used.) Shots from different angles in 3D visualization system are also shown, as well as one case (at the bottom right of Figure 15) where instead of cubes we

[5] From `https://data.world/tronovan/shakespeare-word-frequencies` on May 27, 2019.

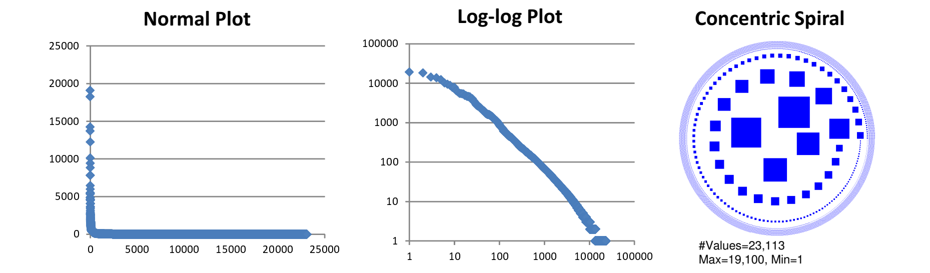


Fig. 14: The frequency of 23,113 words in Shakespeare: (a) normal, (b) log-log plot, (c) Concentric Spiral

use cuboids, where all cubes have the same width (in this case the width of the smallest cube) for comparing the values more easily based on only their heights.

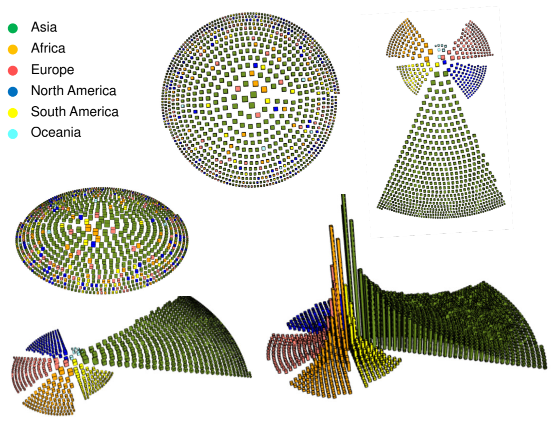


Fig. 15: The 1000 biggest cities clustered by continent

Coronavirus (COVID-19). Figure 16 shows in 3D all covid-19 cases, recovered cases and deaths on May 8, 2020: the bottom cubes represent the total cases detected with covid-19, the intermediate correspond to the recovered cases while the upmost to the total deaths. The above (and some more examples) are available at the webpage `https://rb.gy/d2impg`.

8.3 Implementations

We have already two implementations of the algorithms in two different settings. The first is stand-alone application (that we call "CoSpi") that allows the user to load a csv file and interactively select the desired version of the algorithms and parameter values enabling to derive the desired drawing(s) and save them as images or SVG files, as shown in Figure 17. The first version of this application has just

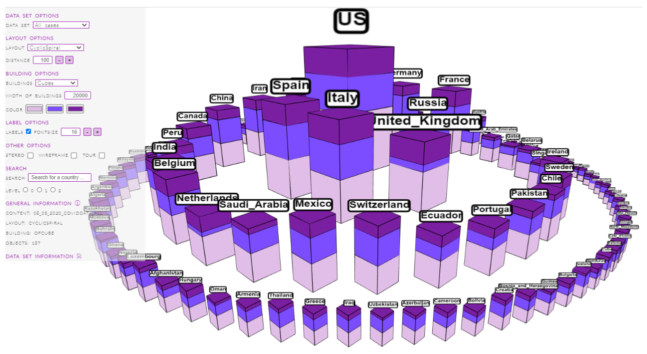


Fig. 16: COVID-19 cases, recovered cases and deaths per country, May 8, 2020

been released publicly (available at `https://rb.gy/oogsld`). The second implementation, is within a Web-based visualization system that exploits the algorithm for producing interactive 3D visualizations (that system was used for producing figures 15 and 16).

8.4 Feedback from Users

In order to understand whether users would like, or prefer, this kind of visualization, we conducted a small-scale evaluation mainly with students. Since not all users are familiar with systems that produce plots and visualizations, and for excluding the interactive functionalities (that depend on the particular system), we selected two easy to grasp datasets, specifically the populations of the 1000 largest cities, and the populations of all countries. For each dataset we produced one pdf that contained visualizations of: (1) normal plot, (2) log-plot, (3) tree-map, (4) sunburst, and (5) CoSpi. Among the various options that are possible with of these types of diagrams (relating to sizes, labels, axes, etc), we selected only two indicative ones for each kind. Then we asked the participants to inspect these visualizations and then to express their preference. No particular task was given to the users, because we did not want to include interactivity, since there is no single system that supports all these visualizations. We also have to mention, that the resolution of the pdf was low because it was produced by screenshots, so the users could not fully read all labels and zoom-in, however, they could get clearly the general idea. We invited by email various persons to participate in the evaluation voluntarily. No training material was given to them, and the participation to this evaluation was optional (invitation by email). Eventually, 28 persons participated (from Dec 10, 2020 to Dec 24, 2020).

In numbers, the participants were 32.1% female and 67.9% male, with ages ranging from 19 to 55 years; Figure 18 shows the histograms of the ages of the participants. As regards occupation and skills, 71.4% were computer science students (undergraduate and graduate) and 28.6% of them professionals (mainly engineers). The task description is shown next:
"At first download the zip file X that contains two excel sheets: one with the populations of the 235 countries and another one with the populations of the 1000 largest cities. Now suppose that you would like to prepare a presentation of these

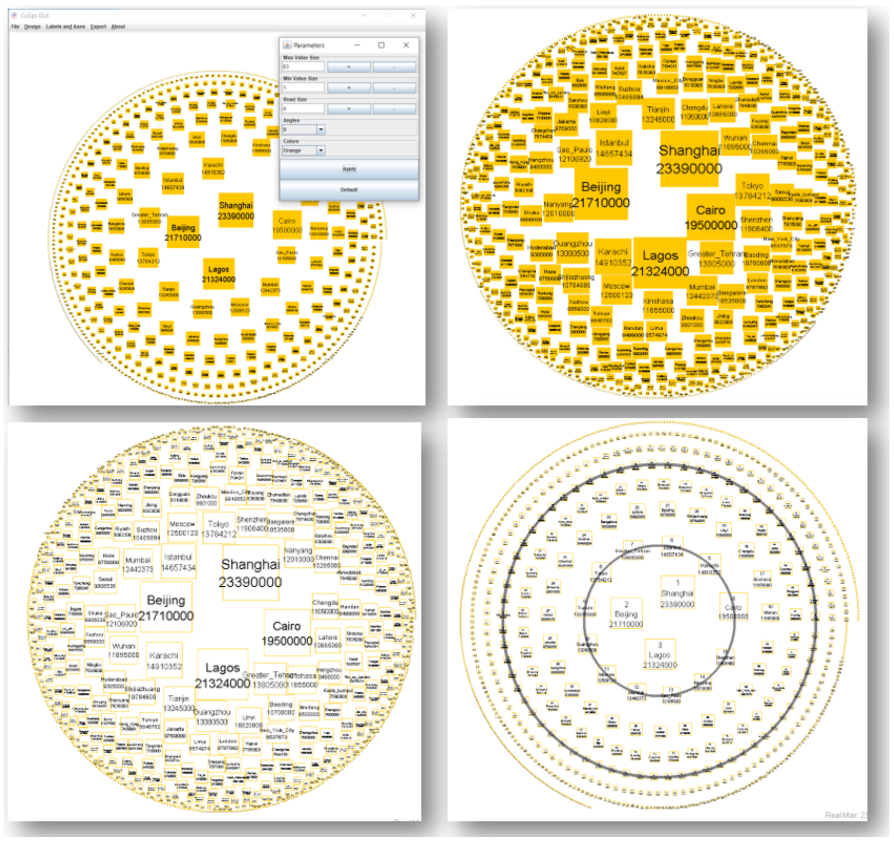


Fig. 17: Examples of visualizations using CoSpi (over a dataset with city populations)

two datasets. Open the two pdf files (that you can find in the zip file) that contain 5 different types of visualization of these two datasets. Please have a look and then fill the small questionnaire that you can find below. Please ignore the low resolution, i.e. do not judge the diagrams based on their resolution."
The questionnaire comprised questions of the form: "*How would you rate the X Plots*: Very Useful, Useful, Little Useful, Not Useful". The results are visualized in Figure 18. By inspecting the results it is clear that users did not like much the classical normal or log plots for the task at hand, instead they preferred tree-map, sunburst and CoSpi. Between the last three, no clear conclusion can be drawn, given the low resolution and the non-interactive nature of the task. They key-point is that users considered CoSpi as an acceptable method, since it achieved preferences quite close with the other well-established methods.

9 Concluding Remarks

The visualization of medium and large number of values that exhibit big variations is a challenging task. We presented a novel family of algorithms that places

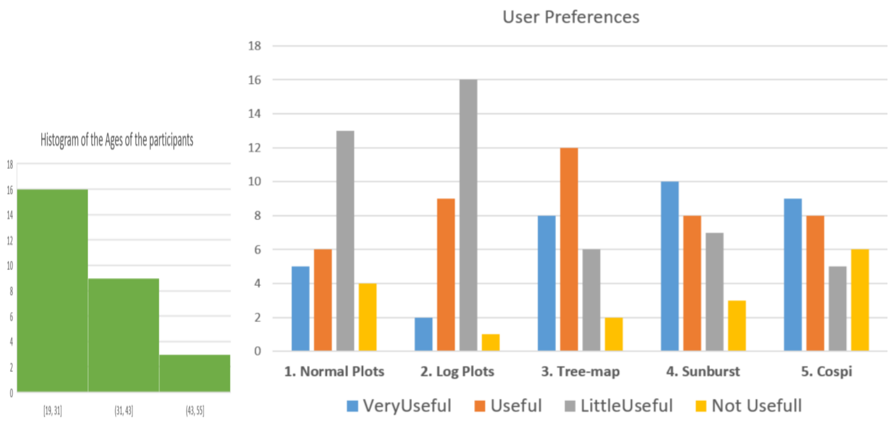


Fig. 18: Feedback by the users: Ages and Preferences of the Participants

objects in the 2D space (in a spiral-like layout), and satisfies a set of requirements that are not supported by the existing methods. One merit of the core algorithm (Concentric Spiral) is that it derives layouts that are suitable for collections of values whose sizes follow power-law because it makes evident the bigger values and it does not leave empty spaces in the peripheral area which is occupied by the majority of the values which are small, thus the produced drawings are both informative and compact, and the aspect ratio of all visualized shapes is 1. The algorithm has linear time complexity (if the values are sorted) and very limited main memory requirements, making it appropriate for very big collections of values (a few seconds are enough for producing the layout of millions of objects). For power laws with an exponent greater than one, it is proved that the occupied space is finite, even if the number of elements is infinite, and we have seen that the enrichment of Concentric Spiral with filled areas is compliant with the theoretical bounds and preserves the relative sizes of all the visualized values. Apart from the core algorithm, we investigated variations of the algorithm that can further reduce the empty space between the big values (if that is required), as well as extensions for showing and comparing more than one sets of values. We proved the feasibility and demonstrated the efficiency of the algorithm by providing two different implementations in different contexts. Finally we showcased applications of the algorithms in various datasets[6]. The proposed method can enrich existing visualization frameworks and *interactive visualizations* in general. In future it is worth investigating automatic methods for tuning the parameters of the algorithms, based on the characteristics of the dataset and the available size of the canvas, for suggesting configurations to the user and plugging the visualization to an engine for analytic queries for investigating various kinds of interactivity.

[6] More examples at: `https://rb.gy/oogsld`

Data Availability Statement

The datasets visualized during the current study are available in the webpage of the system `http://www.ics.forth.gr/~tzitzik/demos/cospi`, specifically the file with the windows executable client contains a folder with the dataset with the city populations, country populations, city populations and continents, and Shakespeare word frequencies.

The covid-19 related datasets that are visualized by the 3D system that is given in the webpage of the system `http://www.ics.forth.gr/~tzitzik/demos/cospi` are dynamically fetched from `https://pomber.github.io/covid19/timeseries.json`.

The various synthetically produced datasets generated and visualized during the current study are not publicly available since they do not have any distinctive characteristic; they were used for measuring efficiency (the latter can be measured by any dataset of that size), but are available from the corresponding author on reasonable request.

The source code of the client will be made available in github upon acceptance of this paper.

References

1. Lada A Adamic and Bernardo A Huberman. Power-law distribution of the world wide web. *science*, 287(5461):2115–2115, 2000.
2. Jeff Alstott, Ed Bullmore, and Dietmar Plenz. powerlaw: a python package for analysis of heavy-tailed distributions. *PloS one*, 9(1):e85777, 2014.
3. James D Anderson and Thomas Wischgoll. Visualization of search results of large document sets. *Electronic Imaging*, 2020(1):388–1, 2020.
4. Gennady Andrienko, Natalia Andrienko, Steven Drucker, Jean-Daniel Fekete, Danyel Fisher, Stratos Idreos, Tim Kraska, Guoliang Li, Kwan-Liu Ma, Jock D. Mackinlay, Antti Oulasvirta, Tobias Schreck, Heidrun Schmann, Michael Stonebraker, David Auber, Nikos Bikakis, Panos K. Chrysanthis, George Papastefanatos, and Mohamed Sharaf. Big data visualization and analytics: Future research challenges and emerging applications. In *Proceedings of BigVis 2020*, 2020.
5. David Auber, Charles Huet, Antoine Lambert, Benjamin Renoust, Arnaud Sallaberry, and Agnes Saulnier. Gospermap: Using a gosper curve for laying out hierarchical data. *IEEE transactions on visualization and computer graphics*, 19(11):1820–1832, 2013.
6. Ricardo Baeza-Yates. Visualization of large answers in text databases. In *Proceedings of the workshop on Advanced visual interfaces*, pages 101–107, 1996.
7. Michael Balzer, Oliver Deussen, and Claus Lewerentz. Voronoi treemaps for the visualization of software metrics. In *Proceedings of the 2005 ACM symposium on Software visualization*, pages 165–172, 2005.
8. Benjamin B Bederson, Ben Shneiderman, and Martin Wattenberg. Ordered and quantum treemaps: Making effective use of 2d space to display hierarchies. *AcM Transactions on Graphics (TOG)*, 21(4):833–854, 2002.
9. Nikos Bikakis, John Liagouris, Maria Kromida, George Papastefanatos, and Timos Sellis. Towards scalable visual exploration of very large rdf graphs. In *International Semantic Web Conference*, pages 9–13. Springer, 2015.
10. Nikos Bikakis, George Papastefanatos, and Olga Papaemmanouil. Big data exploration, visualization and analytics. Elsevier RADARWEG 29, 1043 NX AMSTERDAM, NETHERLANDS, 2019.
11. Nikos Bikakis and Timos Sellis. Exploration and visualization in the web of big linked data: A survey of the state of the art. *arXiv preprint arXiv:1601.08059*, 2016.
12. Mark Bruls, Kees Huizing, and Jarke J Van Wijk. Squarified treemaps. In *Data visualization 2000*, pages 33–42. Springer, 2000.

13. John V Carlis and Joseph A Konstan. Interactive visualization of serial periodic data. In *Procs of the 11th annual ACM symposium on User interface software and technology*, pages 29–38, 1998.
14. Adriane Chapman, Elena Simperl, Laura Koesten, George Konstantinidis, Luis-Daniel Ibáñez, Emilia Kacprzak, and Paul Groth. Dataset search: a survey. *The VLDB Journal*, 29(1):251–272, 2020.
15. Aaron Clauset, Cosma Rohilla Shalizi, and Mark EJ Newman. Power-law distributions in empirical data. *SIAM review*, 51(4):661–703, 2009.
16. Manuel Pérez Cota, María Díaz Rodríguez, Miguel Ramón González-Castro, and Ramiro Manuel Moreira Gonçalves. Massive data visualization analysis analysis of current visualization techniques and main challenges for the future. In *2017 12th Iberian Conference on Information Systems and Technologies (CISTI)*, pages 1–6. IEEE, 2017.
17. Aba-Sah Dadzie and Emmanuel Pietriga. Visualisation of linked data–reprise. *Semantic Web*, 8(1):1–21, 2017.
18. Mark de Berg, Bettina Speckmann, and Vincent van der Weele. Treemaps with bounded aspect ratio. *Computational Geometry*, 47(6):683–693, 2014.
19. Çağatay Demiralp, Peter J Haas, Srinivasan Parthasarathy, and Tejaswini Pedapati. Foresight: Recommending visual insights. *Proceedings of the VLDB Endowment*, 10(12):1937–1940, 2017.
20. Xiaoou Ding, Hongzhi Wang, Jiaxuan Su, Zijue Li, Jianzhong Li, and Hong Gao. Cleanits: a data cleaning system for industrial time series. *Proceedings of the VLDB Endowment*, 12(12):1786–1789, 2019.
21. Geoffrey M Draper, Yarden Livnat, and Richard F Riesenfeld. A survey of radial methods for information visualization. *IEEE transactions on visualization and computer graphics*, 15(5):759–776, 2009.
22. Michalis Faloutsos, Petros Faloutsos, and Christos Faloutsos. On power-law relationships of the internet topology. In *ACM SIGCOMM computer communication review*, volume 29, pages 251–262. ACM, 1999.
23. Jean-Daniel Fekete, Danyel Fisher, Arnab Nandi, and Michael Sedlmair. Progressive data analysis and visualization (dagstuhl seminar 18411). Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019.
24. Javier D Fernandez, Miguel A Martınez-Prieto, Pablo de la Fuente Redondo, and Claudio Gutierrez. Characterising rdf data sets. *Journal of Information Science*, 2017.
25. Martin Gerlach and Eduardo G Altmann. Testing statistical laws in complex systems. *Physical review letters*, 122(16):168301, 2019.
26. Parke Godfrey, Jarek Gryz, and Piotr Lasek. Interactive visualization of large data sets. *IEEE Transactions on Knowledge and Data Engineering*, 28(8):2142–2157, 2016.
27. Jochen Görtler, Christoph Schulz, Daniel Weiskopf, and Oliver Deussen. Bubble treemaps for uncertainty visualization. *IEEE transactions on visualization and computer graphics*, 24(1):719–728, 2017.
28. Ling Hu, Kenneth A Ross, Yuan-Chi Chang, Christian A Lang, and Donghui Zhang. Queryscope: visualizing queries for repeatable database tuning. *Proceedings of the VLDB Endowment*, 1(2):1488–1491, 2008.
29. Ajaz Hussain, Khalid Latif, Aimal Tariq Rextin, Amir Hayat, and Masoon Alam. Scalable visualization of semantic nets using power-law graphs. *Applied Mathematics & Information Sciences*, 8(1):355, 2014.
30. Takao Ito, Kazuo Misue, and Jiro Tanaka. Drawing clustered bipartite graphs in multi-circular style. In *Information Visualisation (IV), 2010 14th International Conference*, pages 23–28. IEEE, 2010.
31. Brian Johnson and Ben Shneiderman. Tree-maps: A space-filling approach to the visualization of hierarchical information structures. In *Procs of the 2nd conference on Visualization'91*, pages 284–291. IEEE Computer Society Press, 1991.
32. Uwe Jugel, Zbigniew Jerzak, Gregor Hackenbroich, and Volker Markl. M4: a visualization-oriented time series data aggregation. *Proceedings of the VLDB Endowment*, 7(10):797–808, 2014.
33. Daniel A Keim. Pixel-oriented visualization techniques for exploring very large data bases. *Journal of Computational and Graphical Statistics*, 5(1):58–77, 1996.
34. Efthymia Kontogiannopoulou, Petros Manousis, and Panos Vassiliadis. Visual maps for data-intensive ecosystems. In *International Conference on Conceptual Modeling*, pages 385–392. Springer, 2014.

35. Rosy Madaan and Komal Kumar Bhatia. Prevalence of visualization techniques in data mining. In *Data Visualization and Knowledge Engineering*, pages 273–298. Springer, 2020.
36. Svetlana Mansmann and Marc H Scholl. Extending visual olap for handling irregular dimensional hierarchies. In *International Conference on Data Warehousing and Knowledge Discovery*, pages 95–105. Springer, 2006.
37. Michalis Mountantonakis and Yannis Tzitzikas. On measuring the lattice of commonalities among several linked datasets. *Proceedings of the VLDB Endowment*, 9(12), 2016.
38. Michalis Mountantonakis and Yannis Tzitzikas. Large-scale semantic integration of linked data: A survey. *ACM Computing Surveys (CSUR)*, 52(5):103, 2019.
39. Maria-Evangelia Papadaki, Panagiotis Papadakos, Michalis Mountantonakis, and Yannis Tzitzikas. An interactive 3d visualization for the lod cloud. In *International Workshop on Big Data Visual Exploration and Analytics (BigVis'2018 at EDBT/ICDT 2018), Vienna, Austria*, March 2018.
40. Janet M Six and Ioannis G Tollis. A framework and algorithms for circular drawings of graphs. *Journal of Discrete Algorithms*, 4(1):25–50, 2006.
41. Max Sondag, Bettina Speckmann, and Kevin Verbeek. Stable treemaps via local moves. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):729–738, 2017.
42. Nicolas Spyratos, Ekaterina Simonenko, and Tsuyoshi Sugibuchi. A functional model for data analysis and result visualization. *ICEB 2009*, pages 57–6, 2009.
43. Chris Stolte, Diane Tang, and Pat Hanrahan. Polaris: A system for query, analysis, and visualization of multidimensional relational databases. *IEEE Transactions on Visualization and Computer Graphics*, 8(1):52–65, 2002.
44. Yannis Theoharis, Yannis Tzitzikas, Dimitris Kotzinos, and Vassilis Christophides. On graph features of semantic web schemas. *IEEE Transactions on Knowledge and Data Engineering*, 20(5):692–702, 2008.
45. Christian Tominski and Heidrun Schumann. Enhanced interactive spiral display. In *SIGRAD 2008. The Annual SIGRAD Conference Special Theme: Interaction; November 27-28; 2008 Stockholm; Sweden*, number 034, pages 53–56. Linköping University Electronic Press, 2008.
46. Ying Tu and Han-Wei Shen. Visualizing changes of hierarchical data using treemaps. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1286–1293, 2007.
47. Yannis Tzitzikas, Nikos Manolis, and Panagiotis Papadakos. Faceted exploration of rdf/s datasets: a survey. *Journal of Intelligent Information Systems*, 48(2):329–364, 2017.
48. Manasi Vartak, Sajjadur Rahman, Samuel Madden, Aditya Parameswaran, and Neoklis Polyzotis. Seedb: efficient data-driven visualization recommendations to support visual analytics. *Proceedings of the VLDB Endowment*, 8(13):2182–2193, 2015.
49. Lidong Wang, Guanghui Wang, and Cheryl Ann Alexander. Big data and visualization: methods, challenges and technology progress. *Digital Technologies*, 1(1):33–38, 2015.
50. Weixin Wang, Hui Wang, Guozhong Dai, and Hongan Wang. Visualization of large hierarchical data by circle packing. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 517–520, 2006.
51. Matthew O Ward. A taxonomy of glyph placement strategies for multidimensional data visualization. *Information Visualization*, 1(3-4):194–210, 2002.
52. Matthew O Ward and Benjamin N Lipchak. A visualization tool for exploratory analysis of cyclic multivariate data. *Metrika*, 51(1):27–37, 2000.
53. Marc Weber, Marc Alexa, and Wolfgang Müller. Visualizing time-series on spirals. In *Infovis*, volume 1, pages 7–14, 2001.
54. Vitalis Wiens, Markus Stocker, and Sören Auer. Towards customizable chart visualizations of tabular data using knowledge graphs. In *International Conference on Asian Digital Libraries*, pages 71–80. Springer, 2020.
55. Ka-Ping Yee, Danyel Fisher, Rachna Dhamija, and Marti Hearst. Animated exploration of dynamic graphs with radial layout. In *Information Visualization, 2001. INFOVIS 2001. IEEE Symposium on*, pages 43–50. IEEE, 2001.
56. Xenophon Zabulis, Panagiotis Koutlemanis, Nikolaos Stivaktakis, and Nikolaos Partarakis. A low-cost contactless overhead micrometer surface scanner. *Applied Sciences*, 11(14), 2021.