2-D Layout for Tree Visualization: a survey

Guanqun Wang[1] , Tsuneo Nakanishi[2] and Akira Fukuda[1]

[1]*Faculty of Information Science and Electrical Engineering, Kyushu University, Fukuoka, Japan 819-0395*
[2]*Department of Electronics Engineering and Computer Science, Faculty of Engineering, Fukuoka University, Fukuoka, Japan 814-0180*

Abstract. This is a survey of recent researches on 2D tree visualization approaches. The whole paper will focus on 2D layouts for general trees including different styles of node-link diagram, main variations of Treemap, some solutions designed especially for mobile devices, and several hybrid approaches. We are not trying to give an exhaustive list of tree layouts here. Instead, we will introduce new ideas in the main categories of general tree layouts, which may inspire more efficient and creative designs.

1 Introduction

Since a major part of real world information is hierarchically structured, and tree is a perfect data structure for the storage of hierarchy information, an effective method of tree visualization becomes necessary in a lot applications. This paper will introduce new tree visualization ideas proposed recently. Although there are already some researches on 2.5D[1, 2] and 3D[3, 4] visualization of trees, to narrow down the topic, this paper will only focus on 2D tree visualization methods. For the same reason, we mainly introduce layouts in this paper and very little information about navigation or interaction techniques will be included.

The researches on tree visualization started about half a century ago [5], and there have been a lot of developed approaches to visualize trees in different situations. Based on how the parent-child relationship is represented, all tree visualization methods can be roughly divided into two large groups, connection methods and enclosure methods [6, 7]. In the following part of this paper, Section 2 will introduce some widely used connection methods; Section 3 will introduce some famous enclosure methods; Section 4 will discuss about other tree visualization methods designed especially for mobile devices; Section 5 will talk about some hybrid methods.

2 Connection Methods

Connection methods uses node-link diagram to explicitly show the relationships of nodes. They make it very easy for the user to get an immediate perception of the structure. However, usually they are not efficient in display space utilization [6]. The following subsections will introduce some main categories of connection methods.

2.1 Level-Based Drawing

In a level-based drawing, all nodes are placed on a set of layers which are several parallel lines in the display area, and typically all nodes from the same level are placed on one unique layer. The structure is pretty explicit in such drawings. However, it usually takes a lot of efforts to improve the space utilization.

Buchheim, Jünger and Leipert summarized 5 aesthetic properties which are commonly required when drawing a rooted tree [8]:

- The y-coordinate of each node corresponds its level.
- The edges do not cross.
- The drawing of a subtree is independent of its position in the tree.
- If the tree is ordered, the ordering information is shown in the drawing.
- The drawing of the reflection of a tree is the reflected drawing of the original tree.

In 1981, Reingold and Tilford proposed the first linear time layout algorithm satisfying all 5 aesthetics, thus it is one of the most famous tree layout algorithms and usually called Reingold-Tilford algorithm now [9]. This algorithm, in a bottom-up direction, recursively draws subtrees independently and puts subtrees with the same parent as close as possible. After all nodes have fixed coordinates, the edges are then inserted accordingly. The Reingold-Tilford algorithm was originally proposed to draw ordered binary trees, but Walker modified it later and successfully made a variation which works for general trees without violating any of the aesthetics [10]. This variation is known as Walker's algorithm in a lot of

researches. In 2002, Buchheim, Jünger and Leipert proved that Walker's algorithm does not run in linear time like Walker claimed and presented a modification of Walker's algorithm with linear runtime [8]. So far all variations of Reingold-Tilford algorithm place a parent at the centre of its children, which in fact, according to the research of Marriott and Sbarski, can lead to an unnecessarily large width. By relaxing the requirement of placing a parent exactly in the centre of its children and taking edge lengths into consideration during drawing, Marriott and Sbarski presented a compact version of Walker's algorithm [11].

When the size varies from node to node, layered drawing may use more space than necessary. In such situations, a non-layered drawing method which places children at a fixed distance from the parent may be more practical. A lot of work on non-layered drawing can be found in the existing literature [12–16], however, none of them gave a linear algorithm for the general case. In 2014, Ploeg proposed a non-layered variation of Reingold-Tilford algorithm and proved it can run in linear time [17].

When we allow nodes from different levels to be placed on the same layer, we can obtain a shorter drawing and the minimum-layer drawing becomes an interesting problem. Some researches have been done on drawing graphs on two or three layers [18]. Suderman have proved optimal upper and lower bounds for h-layer drawing of trees and proposed a linear layout algorithm matching the upper bound [19]. Alam, Samee, Rabbi, and Rah- man proposed a layout algorithm for minimum-layer up- ward drawing which can run in linear time [20]. One year later, Mondal, Alam, and Rahman demonstrated a linear algorithm for general minimum-layer drawing [21]. In- stead of the pathwidth [22] used in a lot of layered drawings, these two algorithms use a new parameter called line- labelling to divide the tree, and the results are satisfying.

2.2 Non-Level-Based Drawing

Non-level-based drawings are not as good at showing structure information as level-based drawing. However, given more freedom of nodes placements, they can achieve improvements in some metrics such as angular resolution, space utilization, aspect ratio, algorithm runtime and so on.

The H-V (Horizontal-Vertical) approach is one of the most well-known none-level-based tree drawing methods. The basic idea of this approach is very simple. It recursively draws the root in the top-left corner and subtrees in horizontal composition (one next to the other) or vertical composition (one below the other), which is shown in Fig.1 [23]. One algorithm based on this approach has been developed for binary trees [24], and it can be easily extended to general trees [23].

Hyperbolic tree is a famous "Focus+Context" visualization method for large hierarchies. It was first proposed by Lamping and Rao [25, 26]. This technique lays out the tree uniformly on a hyperbolic plane which is mapped onto a circular display area. As shown in Fig.2 [25], this kind of lay- out assigns more space to a specific portion of the tree (focus) without sacrifice of context presentation so that it can effectively visualize a much larger tree than the conventional methods.

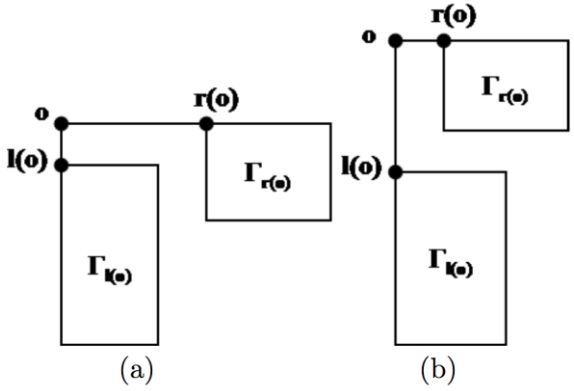

Figure 1. (a) Horizontal composition (b) Vertical composition

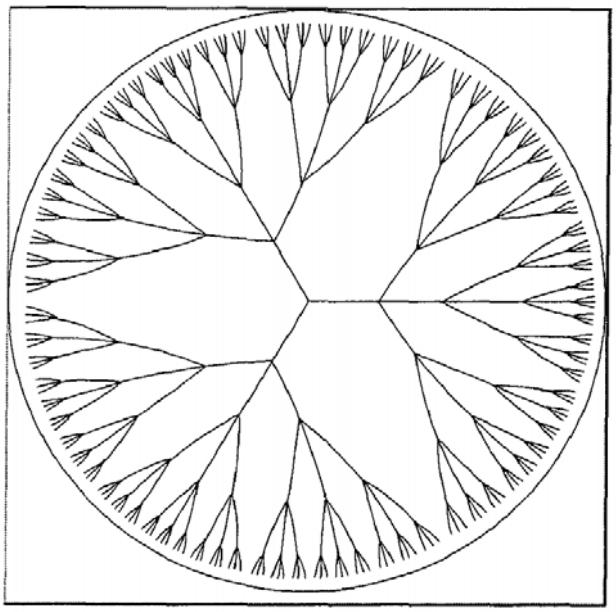

Figure 2. A uniform tree of depth 5 and branching factor 3 (364 nodes)

Radial drawing is another popular category of graph visualization. Fig.3 [24] shows a transformation from layered drawing to radial drawing. Radial drawing can be considered as a special category of level-based drawing which uses polar coordinates instead of Cartesian coordinates. All nodes are placed on several concentric circles while the root is placed at the centre, and nodes from the same level are places on the same circle. Bachmaier presented a radial version of the Sugiyama Framework [27] by placing nodes on concentric circles instead of horizontal lines and drawing edges as outward monotone segments of spirals instead of straight lines. The experiments showed that this drawing style allows more flexible edge and the number of crossings is reduced by about 30% [28].

Duncan, Eppstein, Goodrich, Kobourov, and Nollen- burg have demonstrated how to draw trees with perfect angular resolution and polynomial area [29]. As shown in

Fig.4 [29], the result is a little similar to radial drawing. They also found that it's not always possible to achieve perfect angular resolution and polynomial area if the tree is ordered. However, if we use Lombardi drawing style [30], which means we draw edges with arcs instead of straight lines, both perfect angular resolution and polynomial area can be guaranteed as shown in Fig.5 [29].

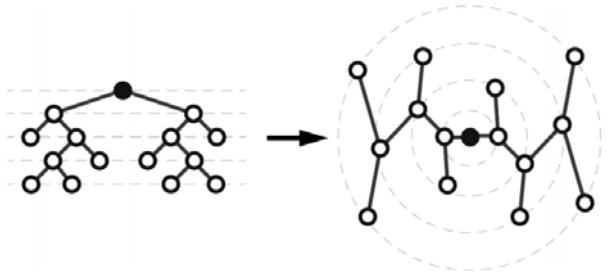

Figure 3. A transformation from layered drawing to radial drawing

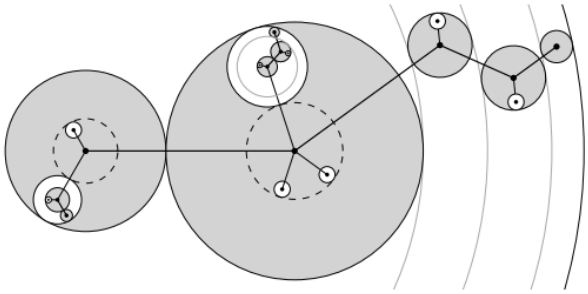

Figure 4. Straight-line drawing for an unordered tree

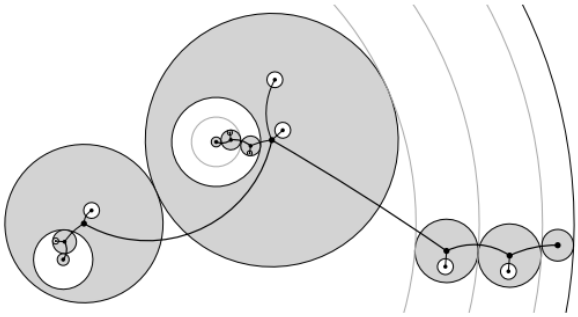

Figure 5. Lombardi drawing for an ordered tree

Ring-based drawing is a kind of drawing style usually used for trees with high degrees. In a ring-based drawing, children are typically placed on the circumference of a circle centred at their parents as shown in Fig.6 [24]. Rusu, crowell, Petzinger, and Fabian also presented an edgeless ring-based tree drawing method called PieVis, one example of which can be found in Fig.7 [31]. All nodes are rep- resented by a circle in this kind of drawing, and children are placed inside of the circle of the corresponding parent. It's important to notice that this PieVis method can only partially visualize the tree by defining a parameter called "degree of interest".

Force-directed layout is another very interesting visualization method. As early as in 1984, Eades modelled a graph as a physical system with rings and springs. With decent parameters, the optimal drawing should occur when the system stays in a stable state [32]. Although there is springs in the system, Eades didn't use Hooke's law to model the force exerted by springs. Instead, he chose his own formula. In fact, it was found that it's very difficult to overcome local minima when using Hooke's law [33]. Fruchterman and Reingold presented a modification of the approach of Eades in 1991. They used a simpler but effective equation to calculate the virtual force and create a faster algorithm. Recently, efforts on graph drawing by physical system simulation and quadratic programming can also be found [34–36]. Even for the Lombardi drawing [30], a relatively new style, there are already some work done on applying force-directed algorithm to generate the optimal drawing [37].

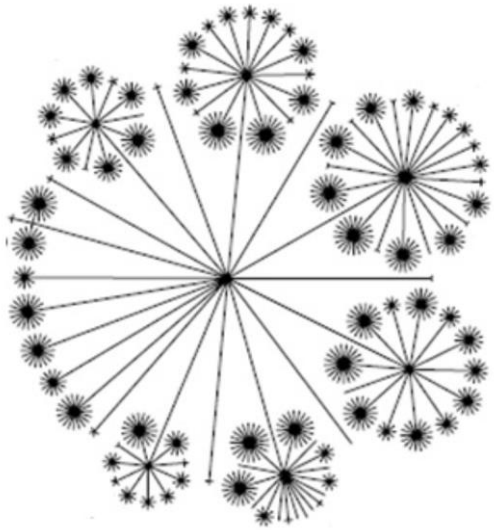

Figure 6. An example of general ring-based drawing

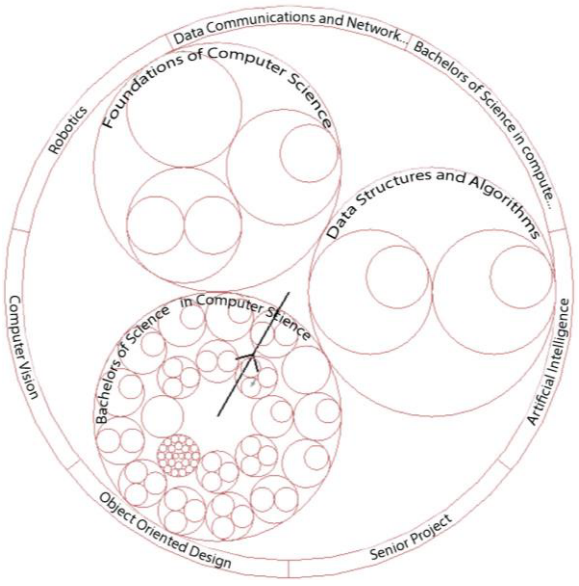

Figure 7. An example of PieVis

3 Enclosure Methods

Enclosure methods use the space-filling concept to map the hierarchical information to a display area and usually can achieve 100% space utilization [6]. Treemap is the most famous and widely used category of enclosure methods and has a lot of variations, so the rest part of this section will focus on Treemap.

Treemap is a widely used tree visualization tool. Application based on it can be found in a lot of fields including financial solution [38], sport information analysis [39, 40], browsing software [41], and even in usenet activity research [42, 43]. The following subsections will introduce the original treemap at first and then some variations based on it.

3.1. Original Idea

In 1990, Shneiderman introduced the concept of Treemap [44, 45]. It is a layout method that represents hierarchical structures with enclosing rectangles while fully utilizing space. In Treemap, each node is represented by a rectangle. The sizes of the rectangles are proportional to the number of children of the node to be partitioned. The general algorithm is usually called "slice and dice". It recursively uses parallel lines to divide a rectangle representing a node into smaller rectangles representing its children nodes. At each level of hierarchy, the orientation of the lines, which is either vertical or horizontal, is switched. The whole process stops when all leaf nodes are reached. Treemap of a tree shown in Fig.8 [44] is shown in Fig.9 [44].

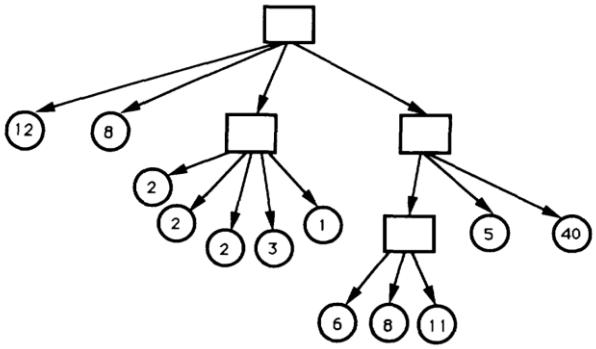

Figure 8. A 3-level tree with a number in each leaf indicating the weight

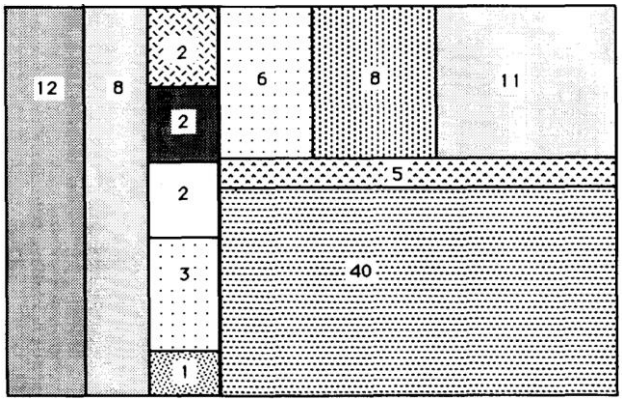

Figure 9. Treemap of the tree shown in Fig.8

3.2 Clustered Treemap

Wattenberg pointed out that there are two problems in the original treemap visualization approach proposed by Shneiderman. The first problem is that this approach may lead to a layout containing rectangles with extremely high aspect ratio, which is very hard to read or label. The second problem is that the original approach can only visualize hierarchical information, which means it cannot show relationships between some special vertices. To solve these two problems, Wattenberg proposed Cluster Treemap [46]. In Cluster Treemap, both vertical and horizontal partitions are employed at each level of the tree to avoid high-aspect-ratio rectangles. Although this new partition method doesn't eliminate extreme aspect ratios theoretically, very obvious improvement in readability was observed in the experiment with real-world data. In all possible layouts, one is chosen to make sure neighbouring rectangles have as much similarity as possible. The more similarity there are between two rectangles, the more similarly how their values have changed in the past. A Cluster Treemap of the original treemap shown in Fig.10 [46] can be found in Fig.11 [46].

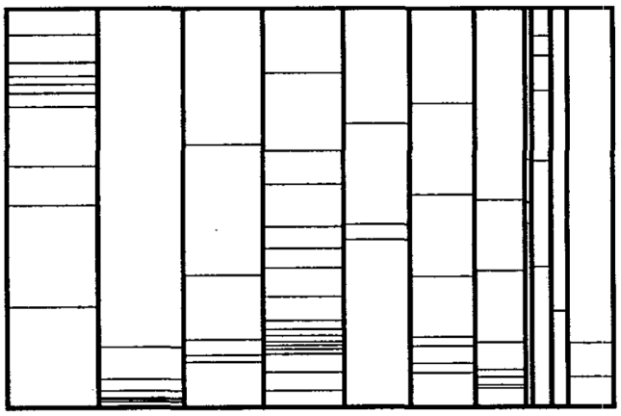

Figure 10. A treemap generated by slice and dice algorithm

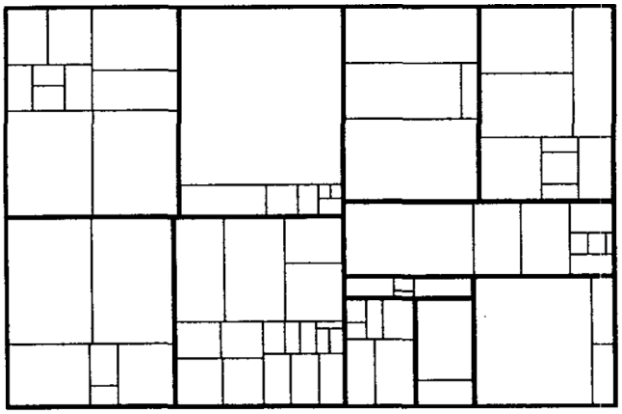

Figure 11. Cluster Treemap of the tree shown in Fig.10

3.3 Cushion Treemap

Wetering and Wijk argued that the original Treemap is not good at visualizing the structure of the tree. Especially when the tree is a balanced tree, the corresponding Treemap turns to a regular grid. To solve this problem, they presented a new visualization method called Cushion Treemap [47]. Cushion Treemaps use intuitive shading to provide insight in the hierarchical structure. A ridge rendered with a simple shading model is added when drawing the rectangle for each node. "Valleys" between adjacent rectangles can be observed after shading, and the depth of the "valleys" is proportional to the distance between the corresponding nodes in the tree. Fig.12 [47] shows one example of Cushion Treemap.

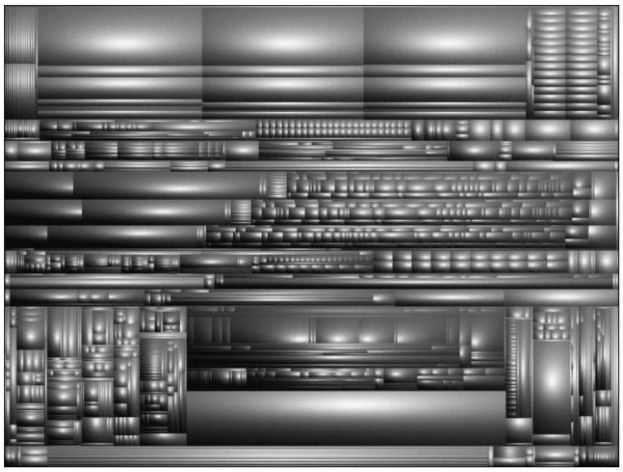

Figure 12. Cushion Treemap example

3.4 Squarified Treemap

After the Cushion Treemap, Bruils, Huizing, and Wijk also proposed another visualization method, Squarified Treemap [48], to solve the large aspect ratio problem of the original Treemap, which was pointed out by Wattenberg before, too. The general algorithm is a little similar to Cluster Treemap. Squarified Treemap also lay out rectangles both horizontally and vertically for nodes in the same level. When drawing rectangles of a set of siblings, the algorithm lays out the new rectangle in either the same direction as the previous one or a new direction. The algorithm will choose whichever direction that has a better worst case of aspect ratio. Although there is no theoretical evidence to show that this algorithm is the best solution to improve the aspect ratios, Bruls, Huizing, and Wijk claimed that this method empirically turned out to give the best results. The Squarified Treemap of the Treemap shown in Fig.13 [48] can be found in Fig.14 [48].

3.5 Ordered and Strip Treemap

In 2001, Shneiderman, and Wattenberg pointed out that Cluster Treemap and Squarified Treemap share several drawbacks [49]. At first, changes in data set can lead to dramatic discontinuous changes in the layouts, which may be very hard for users to follow. In fact, Heer and Roberston have proved by experiments that smooth transitions can help users track changes in data [50]. The other problem is that no ordering information is included, which is sometimes very useful for readers to see patterns or find particular items.

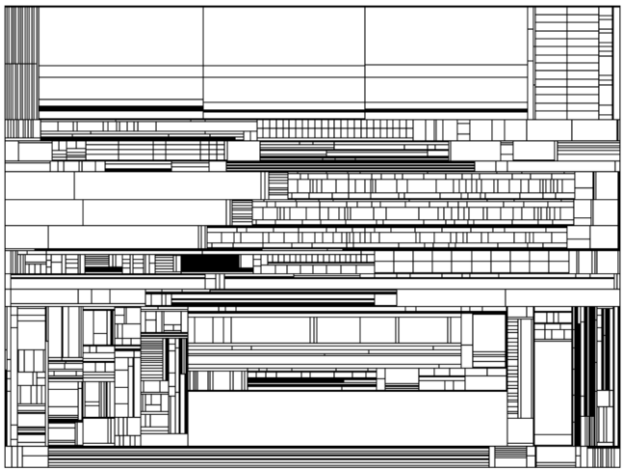

Figure 13. A treemap generated by slice and dice algorithm

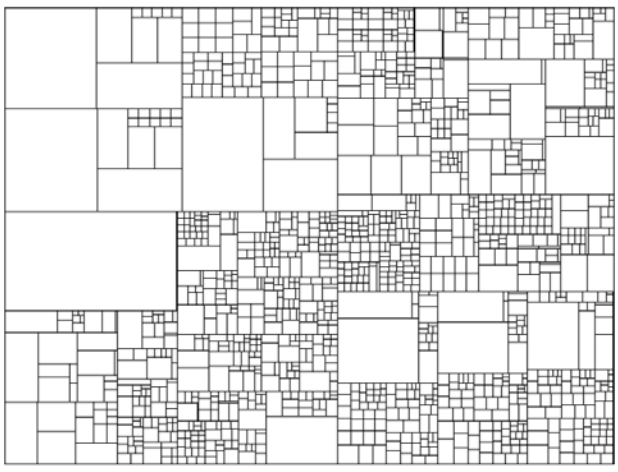

Figure 14. Squarified Treemap of the tree shown in Fig.13

In order to solve these two problems, they presented Ordered Treemap [49]. In Ordered Treemaps, items that are next to each other in the given order are also adjacent. The basic idea of Ordered Treemap algorithm is similar to the conventional Quicksort algorithm. It recursively divides the display area (a rectangle) into 4 sub-areas in a fixed way, divide the items to display into 3 groups and one single item while all items in each group are continuous in the given order, and then assign the 3 groups and one item to 4 sub-areas until every single item has its own rectangle. The single item apart from 3 groups is selected based on some assumptions and there are several decent options to make the decision.

An alternative solution for Ordered Treemap, Strip Treemap, was also introduced by Bederson, Shneiderman, and Wattenberg [51]. Strip Treemap is a modification of Squarified Treemap [48]. It divides the whole display area into several strips with different widths. Each strip is divided again into several rectangles, each of which represents a node in the tree. The general algorithm recursively decide to either draw the new rectangle in the current strip or a new strip. The decision is made based on the how the average aspect ratio changes respectively. Fig.15, Fig.16, and Fig.17 [51] shows visualization results of a tree with original treemap algorithm, Ordered Treemap algorithm, and Strip Treemap respectively.

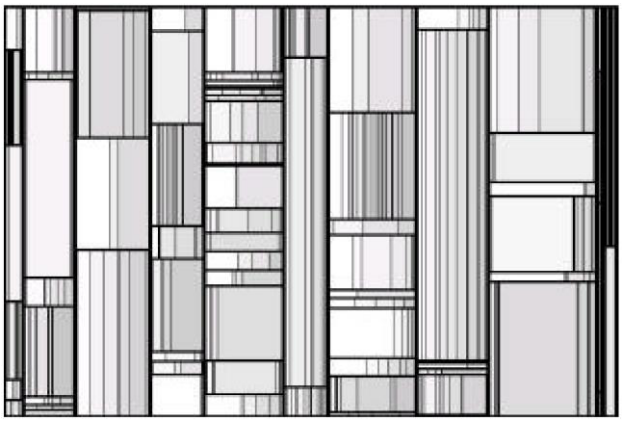

Figure 15. A treemap generated by slice and dice algorithm

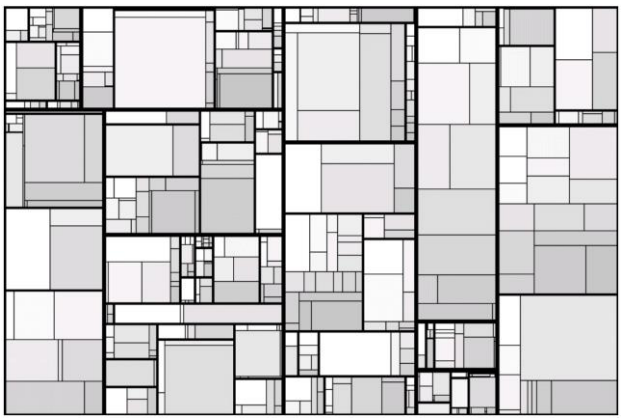

Figure 16. Ordered Treemap of the tree shown in Fig.15

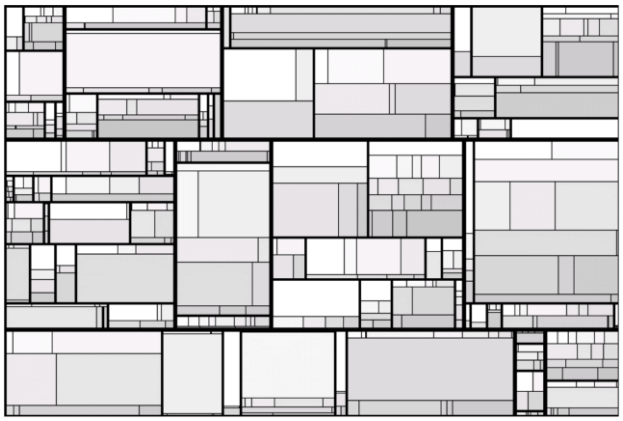

Figure 17. Strip Treemap of the tree shown in Fig.15

3.6 Quantum Treemap

Bederson, Shneiderman, and Wattenberg also proposed Quantum Treemap in the paper introducing Strip Treemap [51]. When images with fixed aspect ratios need to be dis- played in each rectangles of treemaps, some images may have to be scaled down to fit their own rectangles, which may finally make the images very hard to read. Quantum Treemap was designed to solve this problem exactly. All widths and heights of rectangles in Quantum Treemaps are multiples of a step size decided by the user. In this way, all images displayed in the treemap can have exactly the same size and perfectly aligned. Any treemap can be modified to a Quantum Treemap easily by adding one quantization step at the end of regular algorithm.

3.7 Voronoi Treemap

According to [52, 53], various shapes in treemap may help the user with pattern recognition and change tracking. However, all conventional treemaps can only rep- resent each node with a rectangle and the entire display area has to be a rectangle, too. Several researches were done to break through this limitation. In 2005, Balzer and Deussen proposed Voronoi Treemap [54, 55]. In Voronoi Treemap, one display area is divided based on centroidal Voronoi tessellations [56] which allows subareas with different shapes. Fig.18 [54] shows an example of Voronoi Treemap where PW is a distance function chosen in Voronoi tessellation. Finding an optimal solution is a NP-complete problem, so the layout algorithm actually tries different parameters to look for an approximation good enough for treemaps [54], which means a lot of computation power is required. In 2010, Sud, Fisher, and Lee proposed a GPGPU-based technique to speed up the layout algorithm [57].

Figure 18. An example of Voronoi Treemap

3.8 Spiral Treemap

In 2007, Tu and Shen presented Spiral Treemap to reduce abrupt layout changes and produce smooth transitions in treemaps [58]. Ordering information is also stored in Spiral Treemap, so it can be considered as an alternative of Ordered Treemap. The main idea of Spiral Treemap is to use 'spirals' for area partition. Items with the same parent are organized as a spiral in the treemap, and the order is reserved. Since usually there are many possibilities for the layout, the layout algorithm use a similar technique to Strip Treemap. The algorithm only starts drawing a rectangle on the next side if the average aspect ratio decreases after adding this rectangle on the current side. One example of Spiral Treemap is shown in Fig.19 [58].

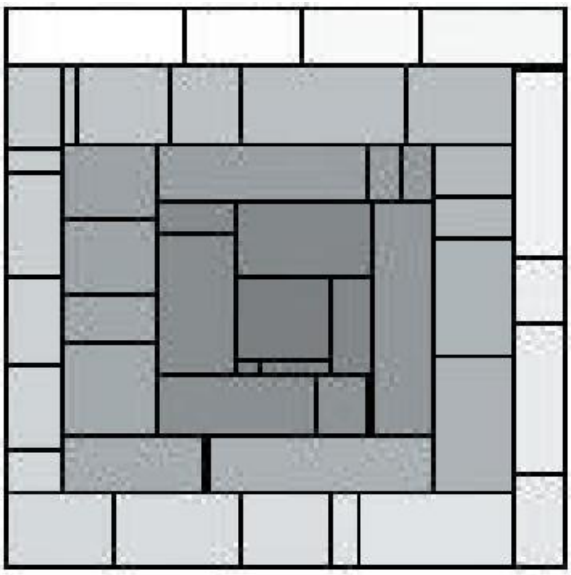

Figure 19. An example of Spiral Treemap

3.9 D&C Treemap

In 2013, Liang and Nguyen proposed D&C Treemap as an alternative solution to break through the "rectangle limitation", and this kind of treemap was claimed to be better than Voronoi Treemap because of lower computation power requirement [59, 60]. D&C Treemap visualizes trees just like other treemaps except that it can partition a polygonal display space into smaller polygons. The algorithm recursively divides the nodes at the same level into two groups with similar total weights and divide the whole display area into two polygons, each of which is as- signed to one group. This process stops when every node has its own polygon. When partitioning the display area, three kinds of algorithms (polygonal/angular/rectangular) are available. Each algorithm can be applied alone, and multiple ones can also be applied together. One example of D&C Treemap is shown in Fig.20 [60].

3.10 Hilbert and Moore Treemap

To achieve smooth transition and store ordering information during data change like Ordered Treemap and Spiral Treemap did, Tak and Cockburn proposed Hilbert and Moore Treemaps, which are two kinds of treemaps based on Hilbert and Moore space-filling curves respectively [61]. The layout algorithm recursively divides the data into four quadrants until each quadrant has four or fewer items. All quadrants have similar weights, and the order or items is reserved. Each level of four quadrants are then laid out by Hilbert or Moore curve. In quadrants at the lowest level, items are laid out by a brute-search method to achieve the lowest aspect ratio. Fig.21 [61] gives an example of Hilbert and Moore Treemap.

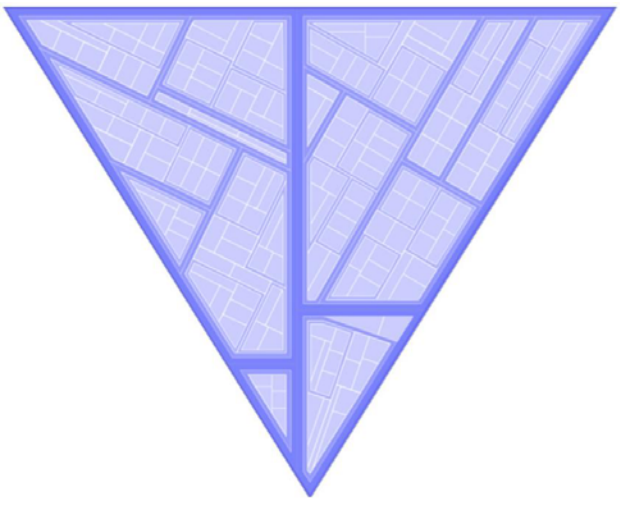

Figure 20. An example of D&C Treemap

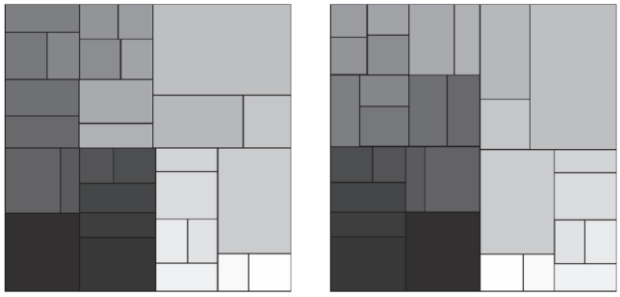

Figure 21. An example of Hilbert (left) and Moore (right) Treemap

4 Methods for Mobile Devices

When mobile devices are used, the display area is usually extremely small and some approaches other than conventional connection and enclosure methods have been pro- posed especially for this situation.

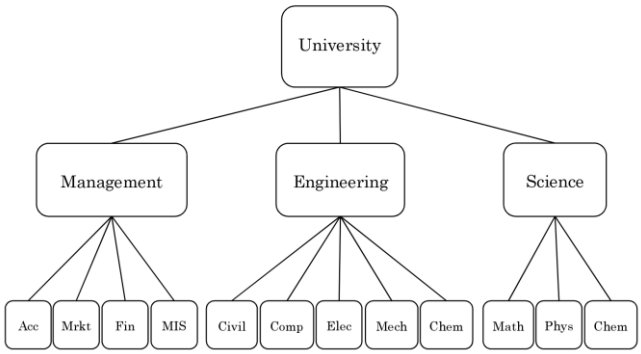

Figure 22. A university structure

Hao, Zhang, and Huang demonstrated "Radial Edge- less Tree" (RELT), a tree visualization method based on space-fill idea for extremely small screen [62, 63]. In RELT, the whole display area is partitioned into several polygons defined by three or four edges. Each polygon represents one node in the tree, and boundary sharing is used to represent the relationship between nodes. Children of one node share the same boundary of this node, and adjacent siblings share one entire boundary. The root is typically placed on the top-left corner of the screen, and other nodes are hierarchically drawn from the top-left to bottom-right, which finally leads to a radial shape. One example of visualization of a university structure by RELT is shown in Fig.22 and Fig.23 [63]. By using boundary sharing instead of links between nodes, a high space utilization is achieved so that a small screen can display more. A proto- type was implemented in Android and compared with the traditional user interface for showing hierarchy information. The results showed that RELT has a relatively long learning curve, which means the compact layout of RELT may hurt its readability.

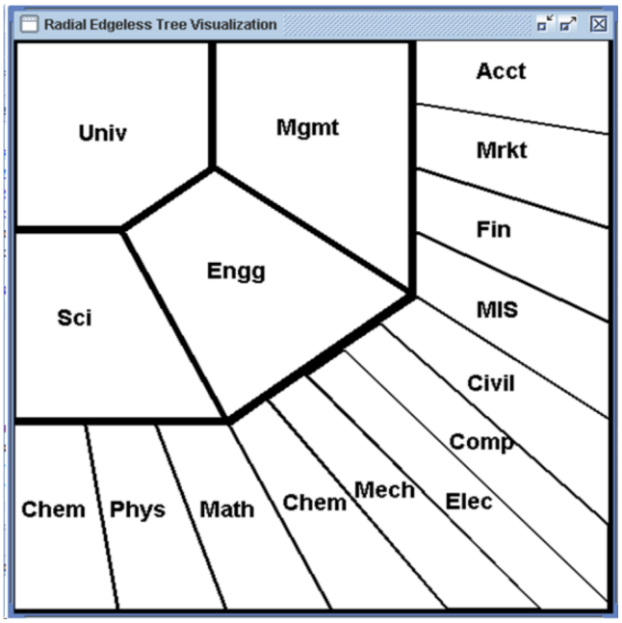

Figure 23. RELT of the university structure shown in Fig.22

Tablorer is another tree visualization system designed especially for mobile devices, which is proposed by Shin, Park, and Han in 2011[64]. In Tablorer, all children of a node are drawn in one rectangle called sibling block. A sibling block is placed either horizontally or vertically, and the orientation alternates according to the level of sib- ling block. In the tree represented by Tablorer, one node is chosen as focus, and all its descendants are defined as the "focus area" while all nodes with higher levels as well as siblings of the focus are defined as the "context area". Three examples of Tablorer can be found in Fig.24 [64]. As Fig.24 (b) and (c) show, all nodes that doesn't belong to focus area or context area are not displayed in Tablorer so that a compact layout can be obtained. Some experiments were also performed to compare Tablorer and other conventional visualization methods and the results showed that it takes 22% less time for users to search information in Tablorer [64].

5 Hybrid Methods

In the existing literature, there are also some examples of tree visualization using hybrid method which is a combination of conventional enclosure approach and connection methods.

In 2005, Zhao, McGuffin, and Chignell proposed Elastic Hierarchies, which is a visualization method combining treemap and node-link diagram [65]. They numerated every possible combination in Fig.25 [65], where **A** is the pure node-link diagram, and **F** is the pure treemap. As shown in Fig.25, **B** is using treemap as a part of the node-link diagram to show higher-level nodes in the tree; **C** and **D** use treemap for the overall structure, and represent some children outside of their parents; **E** uses node-link diagram as a part of treemap to show some higher-level nodes. Zhao, McGuffin, and Chignell implemented a prototype which supports every kind of Elastic Hierarchy shown in Fig.25 except **E**.

Another hybrid approach, EncCon, was also demonstrated in 2005 by Nguyen and Huang [6]. Instead of directly combining treemap and node-link diagram, EncCon uses the basic idea of treemap, space-filling technique, to layout a node-link diagram for maximization of space utilization. EncCon recursively divide the display area into several rectangles and assign each rectangle to a node. Each node is drawn at the centre of its rectangle and edges connecting nodes are added later accordingly. The rectangular division naturally guarantees very few node over- lapping or edge crossing and high space utilization. Be- cause children will finally be around their parent like a circle, a rectangular division algorithm similar to Squarified Treemap [48] is applied so that each node's rectangle can have an aspect ratio close to 1, which will lead to better readability. One example of EncCon is shown in Fig.26 [6], where the numbers are only names of nodes.

In 2011, Marriott, Sbarski, Gelder, Prager, and Bulka proposed a new tree visualization method, Hi-Tree [66]. Nodes in Hi-Tree are categorized into two groups, simple and compound. Children of a simple node are connected to the node by edges, which is similar to node-link diagrams. Children of a compound node are contained within the node, which is similar to treemaps. A compound node is required to have at least one child and its children must be simple nodes. One algorithm for compact layout was also demonstrated in [66] so that Hi-Tree can be applied in small display area. One example of Hi-Tree is shown in Fig.27 [66].

Acknowledgement

This work was partially supported by JSPS KAKENHI Grant Number 15H05708.

References

1. D. Turo, B. Johnson, "Improving the visualization of hierarchies with treemaps: design issues and experimentation." *Proceedings of the 3rd conference on Visualization '92*, 124-131 (1992)
2. H. Lü, J. Fogarty, "Cascaded treemaps: examining the visibility and stability of structure in treemaps." *Proceedings of graphics interface 2008.*, (2008)
3. D. Urribarri, S. M. Castro, S. Martig. "Gyrolayout: A Hyperbolic Level-of-Detail Tree Layout." Journal of universal computer science, **19.1**, 132-156, (2013)
4. G. G. Robertson, J. D. Mackinlay, S. K. Card, "Cone trees: animated 3D visualizations of hierarchical information." *Proceedings of the SIGCHI conference on Hu- man factors in computing systems.*, 189-194 (1991)
5. D. E. Knuth, *The Art of Computer Programming* (Addison-Wesley, Reading, MA, 1968) Chapter1
6. Q. V. Nguyen, M. L. Huang, "EncCon: an approach to constructing interactive visualization of large hierarchical data." Information Visualization, **4.1**, 1-21(2005)
7. M. J. McGuffin, R. Balakrishnan, Interactive "Visualization of Genealogical Graphs." *Proc. IEEE Symp. Information Visualization*, 16-23 (2005)
8. C. Buchheim, M. Jünger, S. Leipert, "Improving Walker's algorithm to run in linear time." *Graph Drawing*, 344-353 (2002)
9. E. M. Reingold, J. S. Tilford, "Tidier drawings of trees." IEEE Transactions on Software Engineering, **2**, 223-228 (1981)
10. J. Q. Walker, "A node-positioning algorithm for general trees." Software: Practice and Experience, **20**, 685–705 (1990)
11. K. Marriott, P. Sbarski, "Compact layout of layered trees." *Proceedings of the thirtieth Australasian conference on Computer science*, **62**, 7-14 (2007)
12. A. Bloesch, "Aesthetic layout of generalized trees." Software: Practice and Experience, **23**, 817–827 (1993)
13. M.Hasan, Md. S. Rahman, T.Nishizeki, "A linear algorithm for compact box-drawings of trees." Networks, **42.3**, 160–164 (2003)
14. Y. Miyadera, K. Anzai, H. Unno, T. Yaku, "Depth-first layout algorithm for trees." Information Processing Letters, **66**, 187–194 (1998)
15. B. Stein, F. Benteler, "On the generalized box-drawing of trees: Survey and new technology." *Proceeding of I-KNOW '07*, (2007)
16. L. Xiaohong, H. Jingwei, "An improved generalized tree layout algorithm." *Proceedings of the 2nd International Asia Conference on Informatics in Control, Automation and Robotics*, **2**, 163–166 (2010)
17. A. Ploeg, "Drawing non-layered tidy trees in linear time." Software: Practice and Experience, **44.12**, 1467- 1484 (2014)
18. S. Cornelsen, T. Schank, D. Wagner, "Drawing graphs on two and three lines." *Graph Drawing*, 31-41 (2002)
19. M. Suderman, "Pathwidth and layered drawings of trees." International Journal of Computational Geometry & Applications, **14.03**, 203-225 (2004)
20. M. J. Alam, Md. A. H. Samee, M. Rabbi, Md. S. Rahman, "Minimum-Layer Upward Drawings of Trees." Journal of Graph Algorithms and Applications, **14.2**, 245-267 (2010)
21. D. Mondal, M. J. Alam, Md. S. Rahman, "Minimum-layer drawings of trees." *WALCOM: Algorithms and Computation*, 221-232 (2011)
22. P. Scheffler, "A linear algorithm for the pathwidth of trees." Topics in combinatorics and graph theory, (Physica-Verlag HD, 1990) 613-620
23. R. Tamassia(editor), "Handbook of Graph Drawing and Visualization." (CRC Press, 2013) 160
24. P. Crescenzi, G. Di Battista, A. Piperno, "A note on optimal area algorithms for upward drawings of binary trees." Comput. Geom. Theory Appl., **2**, 187-200 (1992)
25. J. Lamping, R. Rao, "Laying out and visualizing large trees using a hyperbolic space." *Proceedings of the 7th annual ACM symposium on User interface software and technology*, 13-14 (1994)
26. J. Lamping, R. Rao, P. Pirolli, "A focus+context technique based on hyperbolic geometry for

visualizing large hierarchies." *Proceedings of the SIGCHI conference on Human factors in computing systems*, 401-408 (1995)
27. K. Sugiyama, S. Tagawa, M. Toda, "Methods for Visual Understanding of Hierarchical System Structures." IEEE Trans. Systems, Man, and Cybernetics, **11. 2**, 109-125 (1981)
28. C. Bachmaier, "A radial adaptation of the Sugiyama framework for visualizing hierarchical information." IEEE Transactions on Visualization and Computer Graphics, **13.3**, 583–594 (2007)
29. C. A. Duncan, D. Eppstein, M. T. Goodrick, S. G. Kobourov, M. Nollenburg, "Drawing trees with perfect angular resolution and polynomial area.", *Graph Drawing*, 183-194 (2010)
30. C. A. Duncan, D. Eppstein, M. T. Goodrick, S. G. Kobourov, M. Nollenburg, "Lombardi drawings of graphs." *Graph Drawing*, 195–207 (2010)
31. A. Rusu, A. Crowell, B. Petzinger, A. Fabian, "PieVis: Interactive Graph Visualization Using a Rings-Based Tree Drawing Algorithm for Children and Crust Display for Parents." *Proc. IEEE Info Vis '11*, 465-470 (2011)
32. P. Eades, "A heuristic for graph drawing", Congressus Nutnerantiunt, **42**, 149–160 (1984)
33. T. M. J. Fruchterman, E. M. Reingold. "Graph drawing by force-directed placement." Softw., Pract. Exper., **21.11**, 1129-1164 (1991)
34. T. Dwyer, Y. Koren, K. Marriott, "Drawing directed graphs using quadratic programming." IEEE Transactions on Visualization and Computer Graphics, **12.4**, 536-548 (2006)
35. M. Bohanec, "DEXiTree: A program for pretty drawing of trees." *Proc. Information Society IS 2007*, 8-11(2007)
36. T. Dwyer, K. Marriott, P. Sbarski. "Hi-tree layout using quadratic programming." Diagrammatic Representation and Inference (Springer Berlin Heidelberg, 2010) 212-219
37. R. Chernobelskiy, K. Cunningham, M. T. Goodrich, S. G. Kobourov, L. Trott, "Force-directed lombardi-style graph drawing." *Graph Drawing*, 78–90 (2011)
38. M. L. Huang, J. Liang, Q. V. Nguyen, "A visualization approach for frauds detection in financial market." *Proc. IEEE Info Vis '09*, 197-202 (2009)
39. L. Jin, D. C. Banks, "Tennisviewer: A browser for competition trees." IEEE Computer Graphics and Applications, **17.4**, 63-65 (1997)
40. D. Turo, "Hierarchical visualization with treemaps: making sense of pro basketball data." *CHI '94* Conference Companion on Human Factors in Computing Systems, 441-442 (1994)
41. B. B. Bederson, "Quantum treemaps and bubblemaps for a zoomable image browser." *Proc. User Interface Systems and Technology*, 71-80 (2001)
42. A. Fiore, M. Smith, "Treemap Visualizations of Newsgroups." Technical Report, Microsoft Research, Microsoft Corporation, Redmond, WA (2001)
43. M. Smith, "Tools for Navigating Large Social Cyberspaces." Communications of the ACM, **45.4**, 51-55 (2002)
44. B. Shneiderman, "Tree visualization with tree-maps: 2-d space-filling approach." ACM Transactions on graphics, **11.1**, 92-99 (1992)
45. B. Johnson, B. Shneiderman. "Tree-maps: A space-filling approach to the visualization of hierarchical information structures." *Proc. IEEE Visualization '91*, 284-291 (1991)
46. M. Wattenberg, "Visualizing the stock market." *CHI'99 extended abstracts on Human factors in computing systems* (1999)
47. J. J. Van Wijk, H. Van de Wetering, "Cushion treemaps: Visualization of hierarchical information." *Proc. IEEE Info Vis '99*, 73-78 (1999)
48. M. Bruls, K. Huizing, J. J. Van Wijk. "Squarified treemaps." *Data Visualization 2000*, 33-42 (2000)
49. B. Shneiderman, M. Wattenberg, "Ordered treemap layouts." *Proc. IEEE Info Vis '01*, 73-78 (2001)
50. J. Heer, G. G. Robertson, "Animated transitions in statistical data graphics." IEEE Transactions on Visualization and Computer Graphics, **13.6**, 1240-1247 (2007)
51. B. B. Bederson, B. Shneiderman, M. Wattenberg, "Ordered and quantum treemaps: Making effective use of 2D space to display hierarchies." ACM Transactions on Graphics, **21.4**, 833-854 (2002)
52. J. Liang, M. L. Huang, "Highlighting in information visualization: A survey." *Proc. IEEE Info Vis '10*, 79- 85 (2010)
53. 15. J. Liang, Q. V. Nguyen, S. Simoff, M. L. Huang, "Angular Treemaps-A New Technique for Visualizing and Emphasizing Hierarchical Structures." *Proc. IEEE Info Vis '12*, 74-80 (2012)
54. M. Balzer, O. Deussen, "Voronoi Treemaps." *Proc. IEEE Info Vis '05*, 49-56 (2005)
55. M. Balzer, O. Deussen, C. Lewerentz, "Voronoi treemaps for the visualization of software metrics." *Proc. ACM SoftVis '05*, 165-172 (2005)
56. Q. Du, V. Faber, M. Gunzburger, "Centroidal Voronoi tessellations: applications and algorithms." SIAM review, **41.4**, 637-676 (1999)
57. A. Sud, D. Fisher, H. Lee, "Fast Dynamic Voronoi Treemaps." *Proc. IEEE International Symposium on Science and Engineering '10*, 85-94 (2010)
58. Y. Tu, H. Shen, "Visualizing changes of hierarchical data using treemaps." IEEE Transactions on Visualization and Computer Graphics, **13.6**, 1286-1293 (2007)
59. J. Liang, S. Simeon, Q. V. Nguyen, M. L. Huang, "Visualizing large trees with divide & conquer partition." *Proceedings of the 6th International Symposium on Visual Information Communication and Interaction*, 79-87 (2013)
60. J. Liang, Q. V. Nguyen, S. Simoff, M. L. Huang, "Divide and Conquer treemaps: Visualizing large trees with various shapes." Journal of Visual Languages & Computing, **31**, 104-127 (2015)
61. S. Tak, A. Cockburn, "Enhanced spatial stability with Hilbert and Moore treemaps." IEEE Transactions on Visualization and Computer Graphics, **19.1**, 141-148 (2013)
62. J. Hao, K. Zhang, M. L. Huang, "RELT–visualizing trees on mobile devices." Advances in Visual

63. Information Systems (Springer Berlin Heidelberg, 2007) 344- 357
64. J. Hao, C. A. Gabrysch, C. Zhao, Q. Zhu, K. Zhang, "Visualizing and Navigating Hierarchical Information on Mobile User Interfaces." International Journal of Advanced Intelligence, **2.1**, 81-103 (2010)
65. H. Shin, G. Park, J. Han, "Tablorer–An Interactive Tree Visualization System for Tablet PCs." *Proceedings of the 13th Eurographics*, **30.3** 1131-1140 (2011)
66. S. Zhao, M. J. McGuffin, M. H. Chignell, "Elastic hierarchies: Combining treemaps and node-link diagrams." *Proc. IEEE Info Vis '05*, 57-64 (2005)
67. K. Marriott, P. Sbarski, T. V. Gelder, D. Prager, A. Bulka, "Hi-trees and their layout." IEEE Transactions on Visualization and Computer Graphics, **17.3** 290-304 (2011)

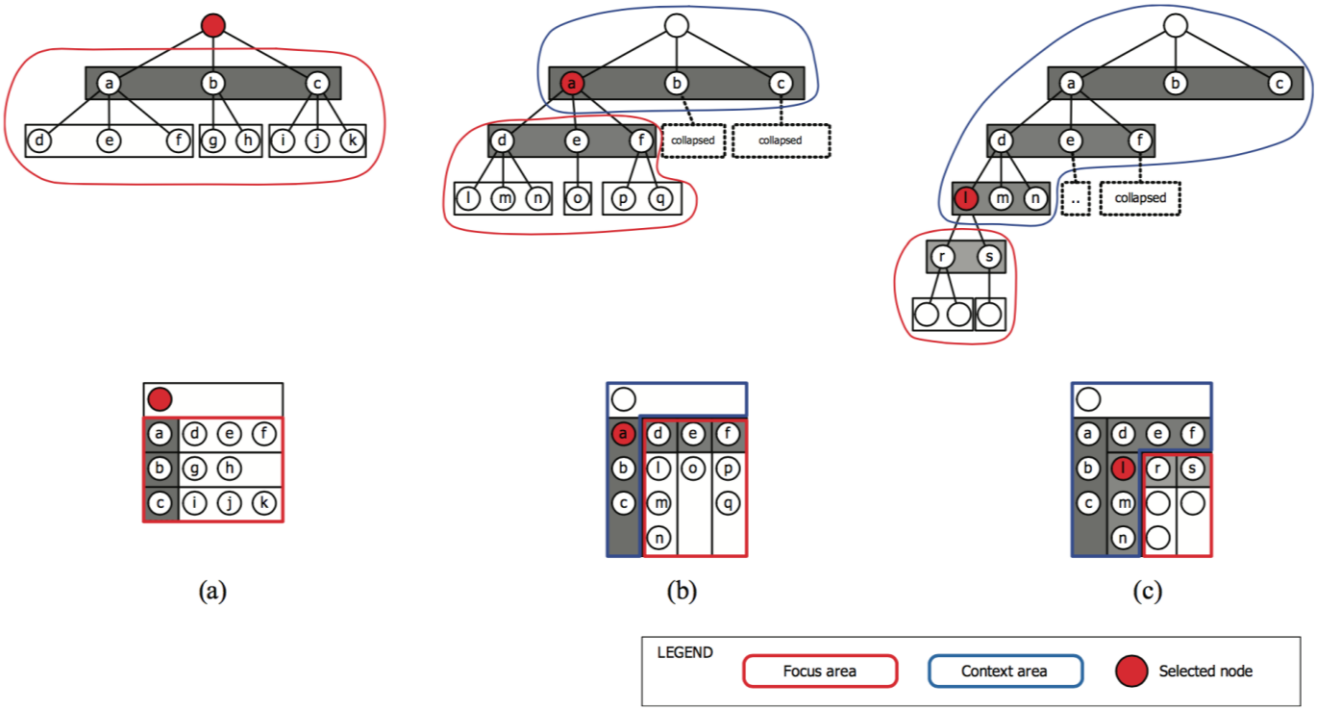

Figure 24. Three different examples of Tablorer

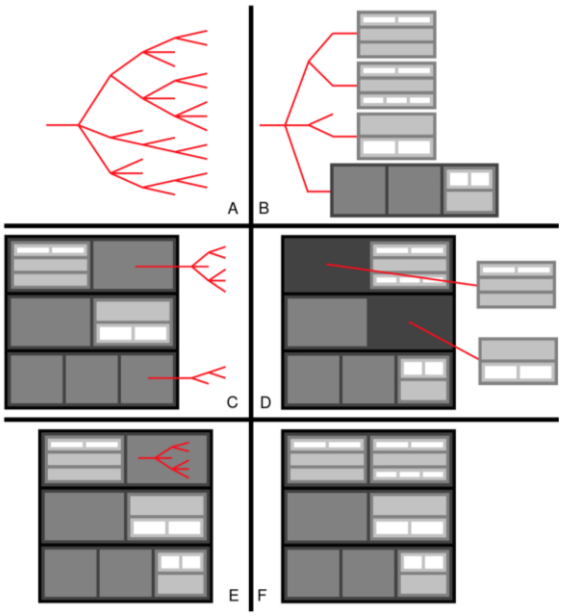

Figure 25. All possible combinations of treemap and node-link diagram

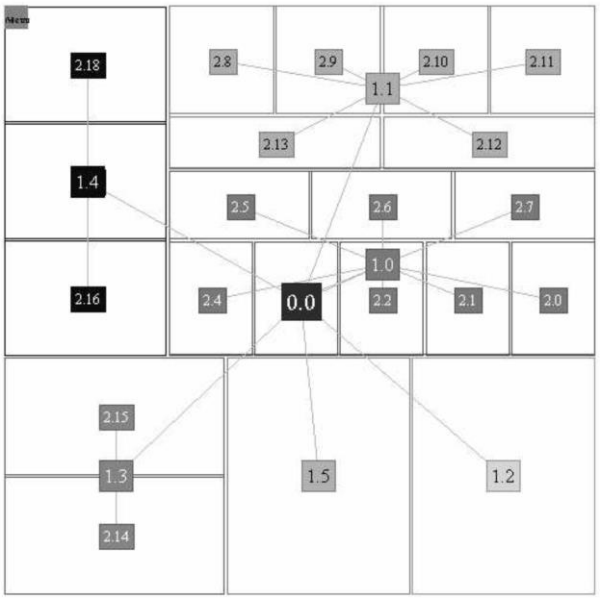

Figure 26. One example of EncCon approach

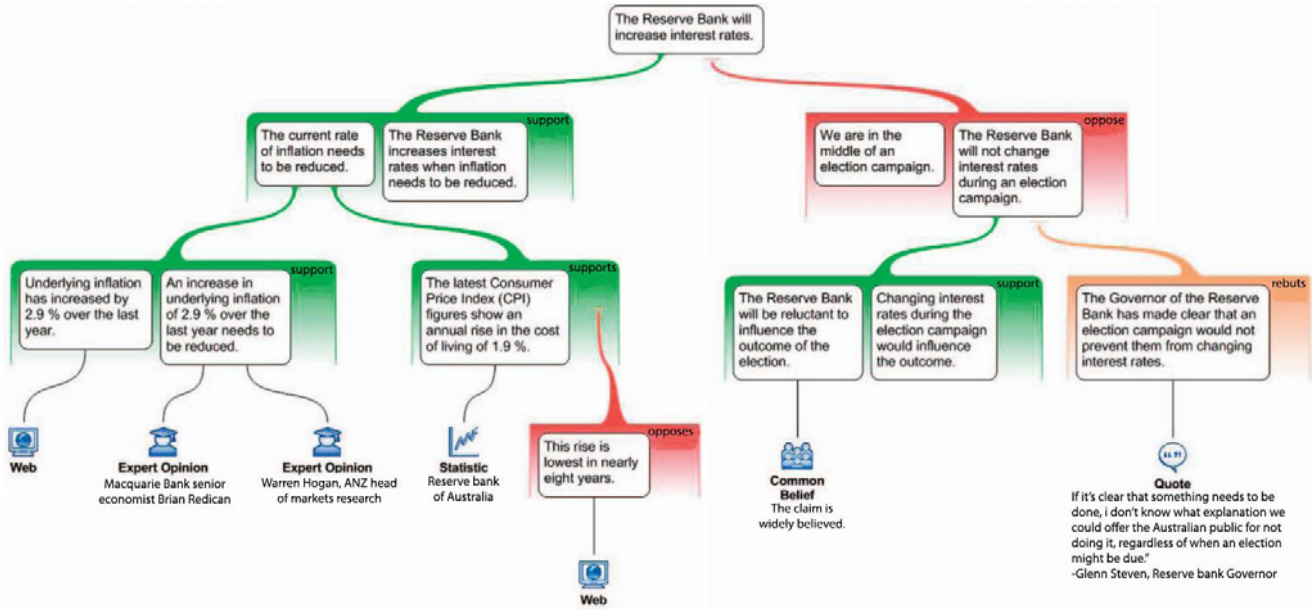

Figure 27. One example of Hi-Tree, an argument map