HierarchyMap: A Novel Approach to Treemap Visualization of Hierarchical Data

Aborisade D. O. And Oyelade O. J

GJCST Classifications:
D.2.12, I.2.8, E.1, G.4, G.2.2

***Abstract*- The HierarchyMap describes a novel approach for Treemap Visualization method for representing large volume of hierarchical information on a 2-dimensional space. HierarchyMap algorithm is a new ordered treemap algorithm. Results of the implementation of HierarchyMap treemap algorithm show that it is capable of representing several thousands of hierarchical data on 2-dimensional space on a computer and Portable Device Application (PDA) screens while still maintaining the qualities found in existing treemap algorithms such as readability, low aspect ratio, reduced run time, and reduced number of thin rectangles. The HierarchyMap treemap algorithm is implemented in Java programming language and tested with dataset of Departmental and Faculty systems of Universities, Family trees, Plant and Animal taxonomy structures.**

Keywords-Treemaps, Aspect ratio, HierarchyMap, Hierarchical data, Tree-like structure, Node.

I. INTRODUCTION

Large volume of data we use today are represented in hierarchical structures, such structures in their natural forms includes information about Corporate Organizations, University/Departmental Structures, Family trees, Manuals Directory, Internet Addressing, Library Cataloging, Computer Programs, Animal and Plant Taxonomy, e.t.c. The contents and organization of these structures are easily understood if they are small, but very difficult to understand when the structures become large (Mark Bruls, et al.,2000). These problems lead to the concept of Treemaps (Shneiderman and Johnson, 1991). Treemap describes the notion of turning a tree into a planar space-filling map. It is described as space-filling visualization method capable of representing large hierarchical collections of quantitative data. A treemap works by dividing the display area into a nested sequence of rectangles whose areas correspond to an attribute of the dataset, effectively combining aspects of a Venn diagram and a pie chart (Shneiderman et al., 2002).
With Treemaps, large hierarchical structures can be viewed without any difficulty because the Treemap visualization method maps hierarchical information into a rectangular 2-dimensional display in a space-filling manner such that 100% of the designated display space is utilized. Interactive control allows users to specify the presentation of both structural (depth bounds, etc.) and content (display properties) information (Shneiderman, 1992). This is in contrast to traditional static methods of displaying hierarchically structured information, which generally makes either poor use of display space or hide vast quantities of information from users. With the Treemap method, sections of the hierarchy containing more important Information can be allocated more display space while ortions of the hierarchy, which are less important to the specific task, can be allocated more space. Although treemaps are originally designed to visualize files on a hard drive (Shneiderman, 1992), it has been applied to a wide variety of areas ranging from financial analysis, business intelligence, money market, stock portfolio to sports reporting (Wattenberg, 1999). A key ingredient of a treemap is the algorithm used to create the nested rectangles that make up the map. These set of rectangles are referred to as the layout of the treemap.
In this work, we developed and implemented a novel HierarchyMap Algorithm. The idea behind this algorithm is to layout information from an hierarchy structures on nested rectangles which we called HierarchyMap Treemap. With this algorithm, every attribute in a hierarchical structure is represented by a rectangular node on the treemap. Each rectangle on the treemap corresponds to an attribute of the dataset. Each of these nodes representing the main attributes of tree-like structures is made to generate the information of sub-nodes of a lower level of the hierarchical structures. This process would continue until all the information in the different levels of the tree hierarchy are displayed one after the other on the same 2-dimensional screen.

II. RELATED WORKS

There are various methods that have been applied to display structure of information, and one of these techniques is the traditional tree diagram where elements are shown as nodes and relations are shown as links from parent to child nodes. More improved techniques have been presented to enhance the efficiency and qualities of such diagram both in 2-dimensional and 3-dimensional space (Furnas , 1986), Knuth, 1973), (Bruggemenn, 1989), and (Card et al.,1991). These techniques have been found to be effective for small trees, but generally ineffective when more than hundreds elements have to be visualized simultaneously. The major reason for this limitation is that node and link diagrams use

Manuscript received "19th December 2009"
1st author: Department of Computer and Mathematical Sciences, College of Natural and Applied Sciences, Crawford University, Faith City, Igbesa, Nigeria.
(Telephone: +234-8056535109
Email: adoj_olan@hotmail.com)
2nd author: Department of Computer and Information Sciences, School of Natural and Applied Sciences, Covenant University, Ota, Nigeria.
(Telephone: +234-8035755778
Email: ola2000faith@yahoo.co.uk)

the display space inefficiently as depicted in the Figure 1 below:

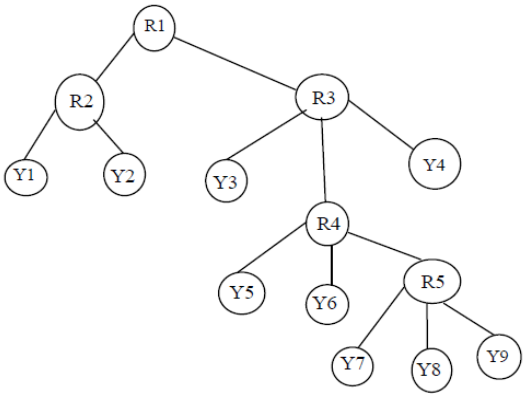

Fig. 1: Tree diagram for representing Hierarchical Data Structure (Mark Bruls et al., 2000)

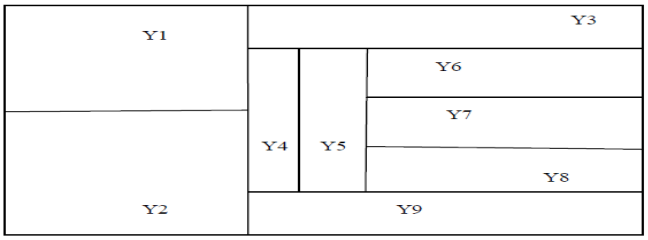

Fig. 2: TreeMap representing the Hierarchical Data Structure in fig. 1 (Mark Bruls et al., 2000)

A treemap as shown in Figure 2 above was developed and introduced to solve the problem of this space usage by using the full display space to visualize the contents of the tree (Johnson and Shneidermann, 1991), (B. Shnerdermann, 1992). As illustrated in Figure 2 above, Slice and Dice treemap algorithm splits the display rectangles along horizontal and vertical lines while recursively traversing a hierarchically structured dataset in top–down direction (Shneiderman,1992). Slice- and –Dice treemap are very effective when size is the most important feature to be displayed. However, this method also has the problem of creating layouts that contain many rectangles with a high aspect ratio. Therefore, many other treemap layout algorithms have been proposed. In order to overcome this limitations. These include Cluster and Squarified treemap algorithms,

Cluster treemap uses a simple recursive algorithm that reduces overall aspect ratios (Wattenberg, 1999), while Squarified treemap algorithm presented the layout of the children in one rectangle as a recursive procedure squarify (Bruls et al., 2000). This procedure lays-out the rectangles in horizontal and vertical rows. When a rectangle is processed, a decision is made between two alternatives, either the rectangle is added to the current row, or the current row is fixed and a new row is started in the remaining sub-rectangle. This decision depends only on whether adding a rectangle to the row will improve the layout of the current row or not.

These methods also have their drawbacks; changes in the data set can cause dramatic discontinuous changes in the layout produced by both cluster treemaps and squarified treemaps. This rapid layout changes also cause an unattractive flickering that draws attention away from other aspects of the visualization and makes it hard to find items on the treemap. Another problem with Cluster and Squarified treemap is that, its layouts fail to preserve order of information as it is done with slice and dice treemap. Many ordered treemap algorithms were introduced to address the limitations in slice-and-dice, Cluster, and Squarified treemap algorithms. The motivating factor here is to seek for the creation of layout in which items that are next to each other in a given order are adjacent in the treemap. Ordered treemaps include Pivot by Split Size, Pivot by Middle, Split and Strip treemap algorithm. These ordered treemaps generally change relatively smoothly under dynamic updates and roughly preserve order, produce rectangles with low aspect ratios compared to that of cluster and squarified treemap (Shneiderman et al. 2002).

Pivot- by- middle algorithm selects the pivot to the middle item of the list so as to create a balanced layout. With this idea, this algorithm is not sensitive to changes as Pivot –by-Split Size. The pivot is taken to be the item (rectangle) with the largest area. Pivot –by-Split-size selects the pivot that will split the list into approximately equal total areas. These two algorithms create layouts that roughly preserve order and are relatively efficient, but fail to produce layouts with relatively low aspect ratio.

Strip algorithm is a modification of the Squarified treemap algorithm. It works by processing input rectangles in order, laying them out in horizontal or vertical strips of varying thickness. It is efficient in that it produces a layout with better readability than the basic ordered treemap algorithm, and reasonable aspect ratios and stability (Shneiderman et al. 2002).

III. Methods

A. Development of HierarchyMap Algorithm

The algorithm for the HierarchyMap treemap is as follows: Infotree(treedata nodes) T={t1,t2 ,t3, ……..., tn} and a 2-D space divided into four equal rectangles.

i. If the number of hierarchical items to be displayed is zero (i.e. T=0) , then no display.
ii. If the number of hierarchical items to be displayed is 1 (i.e T=1), then Set 2-D space to the item.
iii. If the number of items is greater than 1, split the rectangular 2-D space into four equal sizes and recursively divides each of the resultant item into fours until all items in the list are exhausted such that $\forall$ ti $\in$ T1, $\forall$ tj $\in$ T2, $\forall$ tK $\in$T3,……$\forall$ tn $\in$ Tn : ti $\leq$ ti+1 $\leq$ tj $\leq$ tj+1 $\leq$ tk $\leq$ tk+1 $\leq$ . . . tn $\leq$ tn+1.
iv. An attribute of each hierarchical item corresponds to an area of each of the nested rectangles is defined as area(R) in such a manner that their areas correspond to the size of the elements of T1, T2 T3, and T4 where area (R1) $\approx$ area (R2) $\approx$ area (R3) $\approx$ . . . . . area (Rn).

The algorithm accepts inputs data in hierarchical form. These input items in their hierarchical order are stored, read

and lay-out on nested rectangles which make up a treemap on the computer screen. The entire 2-dimensional computer screen is divided first into four equal parts, each of the successive parts is then repeatedly divided into four parts in such a way that the resultant rectangles are grouped according to the nodes level to be represented in the entire hierarchical data. This is to ensure that the order of the items to be displayed is maintained. These items are then linked to each of the resultant rectangles that make up the treemap. Each rectangle that represents the node level of tree data can then clicked repeatedly to display the sub-node elements. Every other nodal rectangle on the treemap could be clicked to display their own sub-node elements in a similar manner. In this process, several thousands of items of information could be displayed and viewed in a single space of 2-dimensional treemap.

IV. Results and Discussion

HierarchyMap algorithm is tested with a several number of sample data of the information structures such as University system, Family system, and Animal Taxonomy. The results of this implementation are represented in Figures 3,4 and 5 respectively. Figure 3 shows the treemap appearance with no information, Figure 4 shows the treemap representation of ten different families Structure and the adjustment of each of the rectangles to reduce their aspect ratio, improve their readability, reduction of thin rectangles . Finally, Figure 5 shows the HierarchyMap for the combination of several tree structures capable of displaying thousands of information. It also shows the adjustment change of the rectangles to demonstrate its optimum measures of the three treemap metrics (i.e. aspect ratio, readability, ordering and capability for change) as data is updated.

The results of this implementation also shows that this HierarchyMap algorithm is similar to other existing treemaps in that, it lays out hierarchical information on nested rectangles, and added further advantage by making it possible to display very large volume of hierarchical information by continuous clicking of node level rectangle, which we have demonstrated in the implementation.

Figure 3: HierarchyMap showing nested rectangles without information

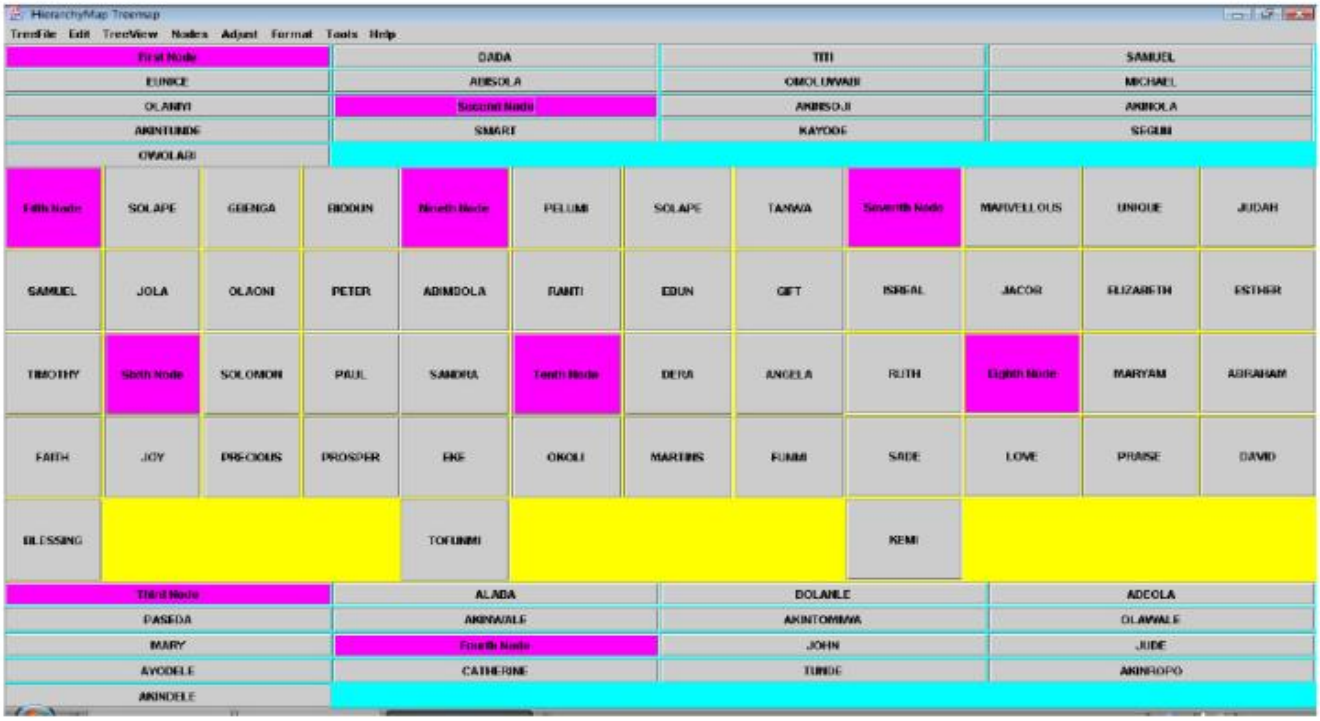

Figure 4: HierarchyMap representing ten different family Structures

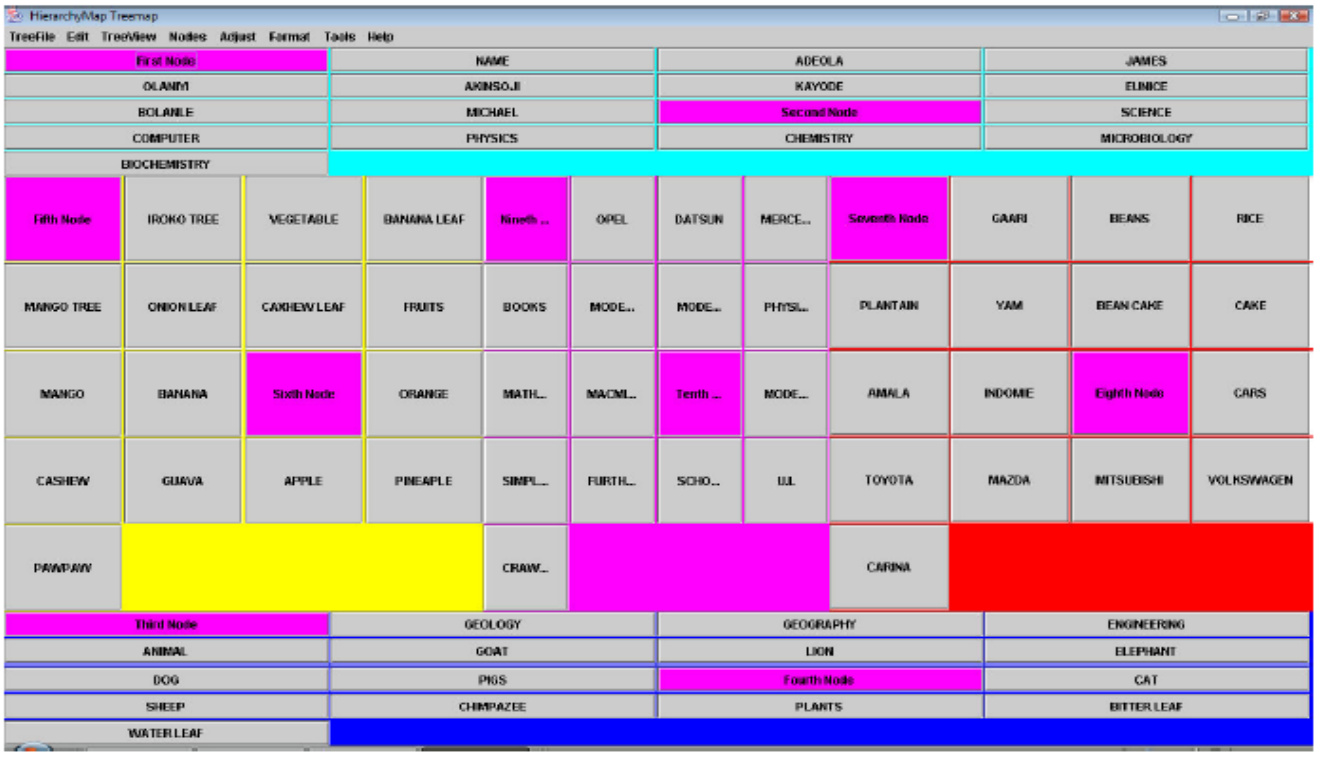

Figure 5: HierarchyMap representing a combination of several hierarchical Structures.

V. Conclusions

In this work, we developed and implemented a novel treemap called HierarchyMap algorithm, which improved on the limitations of the existing treemap algorithms such as Slice-and-dice, Cluster, Squarified, Strip, etc. and added a new feature, which enable viewing of several thousands of hierarchical information by clicking on any of the nodal rectangles. The result showed that the HierarchyMap treemap algorithm has the capability for adjustment change whenever data are updated; it also improved on readability, preservation of order, low aspect ratio, and reduced number of thin rectangles. The combination of these treemap metrics makes HierarchyMap a promising treemap algorithm for the future.

VI. References

1) Bruggemann-Klein and D. Wood. Drawing trees nicely with tex. Electronic Publishing, 2(2):101–115, 1989.
2) Johnson and B. Shneiderman. Treemaps: A space-filling approach to the Visualization of Hierarchical Information Structures. In Proc. of the 2nd International IEEE Visualization Conference, pages 284–291, October 1991.
3) Shneiderman. Tree visualization with treemaps:A 2-D space-filling approach. ACM Transactions on Graphics, 11(1):92–99, September 1992.
4) Bruls S., M., Huizing, K., and Van Wijk, J., 2000. Squarified treemaps. In Proceedings of the Joint Eurographics and IEEE TCVG Symposium on Visualization
(VisSym), 33–42.
5) Bederson, B., Shneiderman, B., and Wattenberg, M. 2002. Ordered and quantum treemaps: Making effective use of 2D space to display hierarchies. ACM Transactions on Graphics 21, 4, 833–854.
6) D.E. Knuth. Fundamental algorithms. Art of computer programming. Volume 1. Addison-Wesley, Reading, MA, 1973.
7) G.W. Furnas. Generalized fisheye views. In Proc. of ACM CHI'86, Conference on Human Factors in computing systems, pages 16–23, 1986.
8) Herman H, Maurer. Data Structures and Programming Techniques. Prentice- All Incorporation. 1977.
9) J. Bingham and S.Sudarsanum. Visualising large hierarchical clusters in Hyperbolic space. Bioinfomatics Chapter 16:pg. 660-661, 2000. Malin Koksal, Visualization of threaded discussions forums on hand-held devices, Masters Thesis at NADA, 2005.
10) Russel Winder and Graham Roberts, Developing Java Software, John Wiley & Sons. 1998.
11) S.K. Card, G.G. Robertson, and J.D. Mackinlay. The information visualizer, an information workspace. In Proc. of ACM CHI'91, Conference on Human Factors in Computing Systems, pages 181–188, 1991.
12) Wattenberg, M. 1999. Visualizing the stock market. In Extended Abstracts onBHuman Factors in Computing Systems (CHI), ACM Press, 188–189.