# Cascading CurtainMap: An Interactive Visualization for Depicting Large and Flexible Hierarchies

by

Bi Wu

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved September 2014 by the
Graduate Supervisory Committee:

Ross Maciejewski, Chair
George Runger
Hasan Davulcu

ARIZONA STATE UNIVERSITY

December 2014

ABSTRACT

In visualizing information hierarchies, icicle plots are efficient diagrams in that they provide the user a straightforward layout for different levels of data in a hierarchy and enable the user to compare items based on the item width. However, as the size of the hierarchy grows large, the items in an icicle plot end up being small and indistinguishable. In this thesis, by maintaining the positive characteristics of traditional icicle plots and incorporating new features such as dynamic diagram and active layer, we developed an interactive visualization that allows the user to selectively drill down or roll up to review different levels of data in a large hierarchy, to change the hierarchical structure to detect potential patterns, and to maintain an overall understanding of the current hierarchical structure.

Keywords: hierarchical structure, large hierarchy, flexible hierarchy, icicle plot, dynamic visualization.

TABLE OF CONTENTS

Page

LIST OF TABLES ..... iv

LIST OF FIGURES ..... v

CHAPTER

1 INTRODUCTION ..... 1

2 RELATED WORK ..... 7

Rearrangement of Nodes and Links ..... 8

Introduction of Extra Dimension ..... 9

Clevenland's Hierarchy ..... 11

Use of Dynamic Diagrams ..... 12

Limitations and Shortcomings of Node-Link Diagrams ..... 13

Space-Filling Diagrams ..... 13

Visualizations for Flexible Hierarchy ..... 22

3 CASCADING CURTAINMAP ..... 27

Modification of Traditional Icicle Plot ..... 27

Color Configuration for the Items ..... 31

Dynamically Fold/Unfold Hierarchical Levels ..... 34

Node-Link Structure ..... 36

Flexible Change of Hierarchy ..... 38

Display Large Hierarchies using Interactive Technique ..... 40

CHAPTER Page

4 CASE STUDY ..... ..... 46

Large Hierarchy .....46

Flexible Hierarchy.....51

Big Data.....54

5 CONCLUSION AND FUTURE WORK ..... 56

REFERENCES..... ..... 59

LIST OF TABLES

Table Page

1. The Usage of Item Height and Item Width ..... 31

LIST OF FIGURES

Figure Page

1. This Icicle Plot First Applies a "brand" Filter and Then Applies a "price" Filter...…….2
2. This Icicle Plot First Applies a "price" Filter and Then Applies a "brand" Filter……....2
3. An Icicle Plot of which the Leaf Nodes are Indistinguishable when Used to Display a Hierarchy with Five Levels…….4
4. A Traditional Node-Link Diagram with Minimum Wasted Space Generated by the Reignold-Tilford Algorithm…….7
5. A Node-Link Diagram and its Corresponding Indented Pixel Tree Plot...….8
6. In a Hyperbolic Tree, all Leaf Nodes are located on the Circumference and their Parent Nodes are inside the Circle……..…9
7. In a Cone Tree, a Third Dimension is Introduced to Convey More Information while Providing Interactions such as Rotation, Selection and Highlighting……...……10
8. Example of a Space Tree……...12
9. Example of a Typical Treemap…….15
10. Treemaps are Confusing when Used to Specify a Node “Mobile” and its Child Node “Mobile Priced Between $1000 and $2000”…….16
11. An Icicle Plot can be Used to Effectively Specify a Parent Node and its Child Node in the Same Time Without Confusion…….……..16
12. An Example of Radial Space-Filling RSF Diagram……..18

Figure Page

13. Hyponymy of the Word Vertebrate……..…….19

14. Expanding the swing/com/sun/java/swing Subdirectory into the Right-Hand Disc……...…….20

15. Visualization of Projects, Grouped First by Region, and Then by Department……22

16. Control Panel with Hierarchy Tab that Enables Users to Specify the Hierarchy……...23

17. An Instance of CatTrees Visualization……..24

18. The Width of the Left Rectangle is 5% Shorter than the Width of the Right Rectangle……...28

19. By Using Vertical Length to Represent Item Values and Aligning Items on Y Axis, it is Easy to See the Difference and Make Comparison Between Items…….....28

20. In the Left Diagram, the Items with Different Values Seem to Have Equal Width; in the Right Diagram, the Same Items are Represented in a Vertical Way that it is Easy to See Their Difference……....…….29

21. Width of Items on the Inactive Layers Represents the Number of Sub Items, whereas Height of Items on the Active Layer Represent the Item value. Items on the Inactive Layers Have the Same Height and Items on the Active Layer Has the Same Width…….……...31

22. When Using the Sequential Color Scheme, the Inactive Layers (Colored Black) Serve as the X Axis for the Items of the Active Layer. A Series of

Figure Page

Green Colors are Used for the Items on the Active Layer. Red is Used to Indicate the Problematic Item……33

23. When Using the Qualitative Color Scheme, the Inactive Layers (Colored Black) Serve as the X Axis for the Items of the Active Layer. Another Set of Colors are Used for the Items on the Active Layer. A Legend is Provided at the Bottom…….....33
24. This is a Complete Unfolded Layout of the Hierarchical Structure with Level 3 (Colored Green) Being the Current Active Level. Level 0, Level 1 and Level 2 (Colored Black) Serve as the X Axis……...……34
25. This is the Layout after Folding One Level from the Hierarchy in (a) with Level 2 Being the Current Active Level. Level 0 and Level 1 Serve as the X Axis…….....35
26. This is the Layout after Folding Two Level from the Hierarchy in (a) with Level 1 Being the Current Active Level. Only Level 0 Serves as the X Axis……35
27. All the Items on the Active Layer are Rearranged in Descending Order by Clicking the Items Right Above the Active Layer. By Hovering an Item on the Active Layer, its Attribute Name and Value are Shown in a Message Box……...….36
28. The Node-Link Structure Displays the Order of Filters Applied to the Corresponding Hierarchy: First "Price", Then "Brand", and Finally "Region"……..…37

Figure Page

29. The Node-Link Structure Displays the Order of Filters Applied to the Corresponding Hierarchy: First "Brand", Then "Price", and Finally "Region"……37
30. The Node-Link Structure Displays the Order of Filters Applied to the Corresponding Hierarchy: First "Price", Then "Brand"…….37
31. A Hierarchy with Two Levels with the Categorical Attribute - “price” Being its Current Bottom Level……...39
32. A Hierarchy with Two Levels with the Categorical Attribute - “phone type” Being its Current Bottom Level……39
33. A Hierarchy with Two Levels with the Categorical Attribute - “years” Being its Current Bottom Level……...40
34. By Clicking an Item on the Active Layer, the Selected Item and Each Item in the Path from the Selected Item up to the Top (Red), that is Cell Phone, $800 - $1500, iPhone, and China, will be Drawn with the Full Width of the Icicle Plot…….41
35. The Black Items that Occupy the Full Width of the Icicle Plot are the Selected Item and All its Ancestors in Figure 21. The Items on the Active Layer Belong to the Selected Item……42
36. When “$800 - $1500” is Selected, the Selected Item and all its Sub Items Occupy the Full Width of the Icicle Plot, as Indicated by the Red Arrows..43
37. After the Selected Item and its Sub Items Occupy the Full Width of the Icicle Plot, All their Sibling Items are Hidden……...……...43

Figure Page

38. In this Status, the Items “$800-$1500”, “iPhone” and “China” All Occupy the Full Width of the Icicle Plot. It will Generate Different Layouts as the User Clicks Any One of Them……...……...44
39. When the User Clicks “China” in (a), since “China” is Already in Focus in (a), it is Defocused and All its Siblings are Shown, and the Width of the Sub Items Change Accordingly……44
40. When the User Clicks “iPhone” in Figure 38, since “iPhone” is Already in Focus in Figure 38, it is Defocused and All its Siblings are Shown, and the Width of the Sub Items Change Accordingly. We don’t Display the “years”, Because the Width of the Item under the Active Layer is Below the Threshold……...……45
41. When the User Clicks “$800-$1500” in Figure 38, since ““$800-$1500” is Already in Focus in Figure 38, it is Defocused and All its Siblings are Shown, and the Width of the Sub Items Change Accordingly. We don’t Display the “years”, Because the Width of the Item under the Active Layer is Below the Threshold……..……45
42. World Population Classified by Continents…..……46
43. By Clicking the “North America” in Figure 42, the North America Population Classified by Countries can be Displayed. Users can Hover on the Items to See the Country Name and its Population……...……..47

Figure Page

44. By Clicking the “North America” in Figure 32, the United States Population Classified by States can be Displayed. Users can Hover On the Items to See the State Name and its Population…….……48

45. By Clicking the “North America” in Figure 33, the Arizona State Population Classified by Cities can be Displayed. Users can Hover On the Items to See the City Name and its Population……..49

46. When “Cities in Arizona” is the Active Layer, it Takes One Single Click to go to the View in which “Countries in North America” is the Active Layer, and it Further Takes Two Steps to go to the View in which “Cities in Sonora” is the Active Layer. The Number of Steps is Equal to the Distance Between “North America”, which is the First Common Ancestor of “Cities in Arizona” and “Cities in Sonora”, and “Cities in Sonora”, which are the Items the User Wants to See……..……..……..50

47. The Flow Chart of Specifying the Order of Filters Applied on the Hierarchy using Cascading CurtainMap……...…….53

48. The Size of the Supply Chain Data Grows as the Number of Hierarchy Levels Increases or as the Time Passes By……..…….54

49. The Processing and Visualization of the Supply Chain Data…….……...55

# CHAPTER 1

## INTRODUCTION

As long as there has been big data, there have been large hierarchies and a significant demand for visualizing them. Different levels in a hierarchy usually have different sets of values. A good visualization should not only represent these values efficiently, but also help the user maintain an overall understanding of the hierarchical structure. As the hierarchy goes deeper, a good visualization should also be able to display the items without causing any difficulty in distinguishing and understanding the values of them.

There are two types of hierarchies - fixed hierarchy and flexible hierarchy. A fixed hierarchy is formed by logically subdividing data from large sets to smaller sets in a fixed order (e.g. country → state → city → district). A flexible hierarchy uses attributes of the data, which can also be called filter, to do the classifications in an order that could be changed according to users' demand. For example, when analyzing information about the popularity of laptops, Apple fans might first apply a "brand" filter to narrow the scope only to MacBook (not Dell, HP or Lenovo) and then apply other filters such as "price" and "color" that are their secondary interests. In Figure 1 we use an icicle plot to represent this case. Note that the width of the icicle plot indicates the popularity of the item. Users with limited amount of money might want to first apply a "price" filter to list all the laptops they can afford, then apply other filters, which is shown in Figure 2.

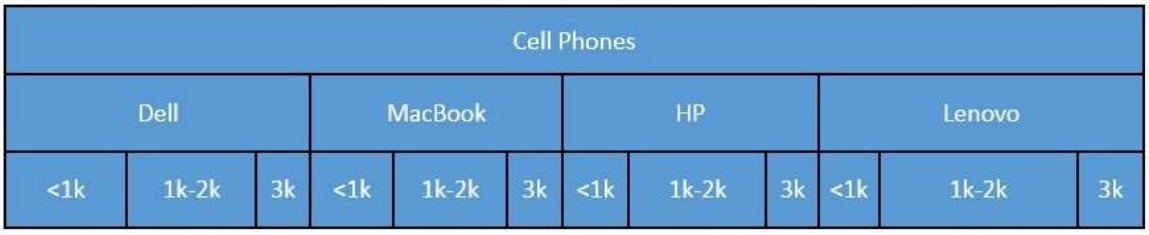


*Figure 1.* This Icicle Plot First Applies a "brand" Filter and Then Applies a "price" Filter.


*Figure 2.* This Icicle Plot First Applies a "price" Filter and Then Applies a "brand" Filter.

These requirements necessitate an effective solution for the following main issues:

1. How to best enable the user to change the order of classifications applied on a hierarchy?
2. How to make a diagram capable of displaying large amounts hierarchical levels?

While most recent studies focus on issues and techniques for visualizing fixed hierarchies, research on flexible hierarchies does not draw much attention. Combined with the fact that big data keeps growing into large hierarchies every day, an effective visualization technique for depicting large and flexible hierarchical structures is sorely needed for data analysis and decision making. For depicting hierarchical structures, the representation schemes traditionally fall into two categories, node-link scheme and space-filling scheme. Although node-link diagrams are traditional and straightforward

representations of hierarchies, the space and aesthetic issues, such as imbalance of the tree (Reingold & Tilford, 1981) and redundant pixels used as background (Van Wijk & Van de Wetering, 1999), keep being the major reasons that prevent them being widely used. Additionally, since node-link diagrams usually use separated circles or rectangles to display data items and no visual variables such as size and color are used to represent their values, it is difficult for people to make comparisons between them. These limitations drive people to focus on space-filling diagrams that utilize visual variables such as size (Treemaps), width (Icicle Plots), and color (Cushion Treemaps) to represent item values and hierarchical relationship.

Some commonly used space-filling diagrams include treemaps, icicle plots, and sunburst. Treemaps (Johunson & Shneiderman, 1991) recursively subdivides a large rectangle into smaller rectangles horizontally and vertically. Icicle plots (Kruskal & Landwehr, 1983) use a top-down technique that keeps subdividing items vertically and all the child nodes are drawn underneath the parent node. Sunburst is a Radical Space Filling (RSF) diagram that is similar to icicle plot and it uses a radical layout for the hierarchical structure. Based on these space-filling visualizations, previous researches and experiments were done to suggest that people preferred sunburst to treemaps (Stasko, Catrambone, Guzdial & McDonald, 2000) and that icicle plot is more favorable than both sunburst and treemaps for its left-to-right and top-to-bottom orientation layout (Barlow & Neville, 2001). In an icicle plot, it is also easier to identify an object compared with other visualizations. According to Kruskal and Landwehr (1983), “the object labels are repeated over and over again vertically means that the eye never needs to travel very far up or down to identify an object” (p. 162), whereas in other diagrams such as mosaic

plots and treemaps, the eye needs to travel both vertically and horizontally to establish such understanding, which may be cognitively burdensome.

However, icicle plots are not without their own shortcomings. First, traditional icicle plots are static diagrams. They cannot be used to display flexible hierarchies. Second, since the algorithm of icicle plot divides the width of root node at each lower hierarchical level, the leaf nodes become more and more crowded and illegible as the number of hierarchy levels increases. In such a case, it is not only difficult to label objects on the bottom level, which makes it impossible to read which objects belong to which cluster, but also difficult to learn the contribution each item makes among its siblings since they are all too narrow. As big data becomes more and more common in information processing and analysis, it is not rare that hierarchical data has multiple levels to the extent that prevents traditional icicle plot from displaying all the information efficiently. As we can see the icicle plot in Figure 3, due to the depth of the hierarchy, some leaf nodes are too narrow to be labelled and it is difficult to make comparison between them by the width.

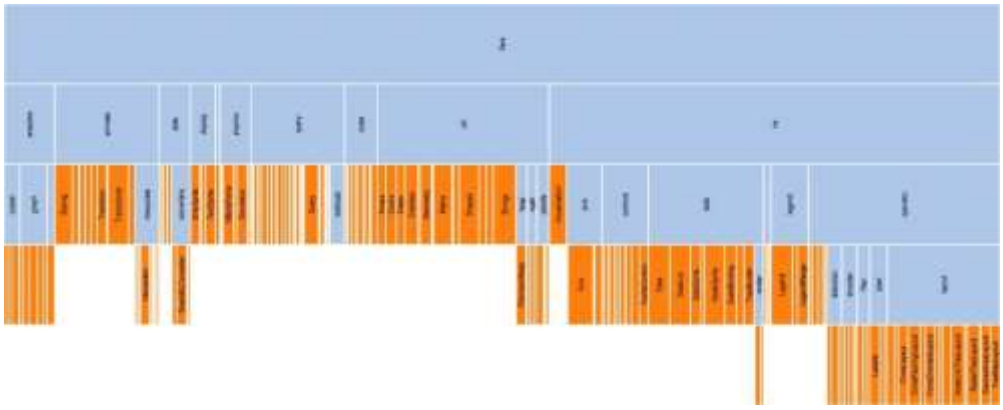

*Figure 3.* An Icicle Plot of which the Leaf Nodes are Indistinguishable when Used to Display a Hierarchy with Five Levels.

The main reason that traditional icicle plots are incapable of displaying large hierarchies is that they use limited number of pixels to visualize potentially unlimited size of hierarchies at once, which is the drawback of most static diagrams. For addressing this issue, Plaisant, Grosjean and Bederson (2002) introduced the Spacetree, which enables the user to perform the "fold/unfold" operation over a non-leaf node (see Figure 8). With this flexibility, a space tree doesn't need to display the entire hierarchy at once, thus providing more space for the nodes that are being examined by the user.

In our cascading CurtainMap, an interactive visualization system for representing large and flexible hierarchical structures, we incorporate the idea of the Spacetree (Plaisant et al., 2002), that is, to dynamically display the items inside a hierarchy according to users' interest, with "the top-to-down and left-to-right" advantage of traditional icicle plots. We first enable the user to fold/unfold the hierarchy based on their interaction with the visualization (see Figure 24), then we introduce the "active layer" for the user to analyze the data based on items' height (see Figure 21). The user can also change the current active layer so that they can deal with flexible hierarchies (see Figure 31-33). For representing large hierarchies, we introduce the "cascading interaction" and enable the user to focus on one item of interest and unfold the underlying hierarchy accordingly (see Figure 34-35), which resembles the mechanism used in a Spacetree.

Our system takes advantage of three Bertin's visual variables – Position, Size and Color. We use position to indicate the parent-child relationship between items on different hierarchical levels; we use size, including both width and length, to represent the number of sub items belonging to a parent node and its value; and we use color to represent the value of an item or distinguish items on the active layer. Combined with

techniques such as redundant representation, auxiliary node-link structure and animations, the user can easily manipulate the hierarchy while maintaining a good understanding of the values it represents. Our system is connected to a MS SQL Server Database, and we use dynamic queries (Ahlberg, Williamson & Shneiderman, 1992), which are connoted by a number of graphical items in our visualization, to retrieve the data from the database and update the front-end view based on users' interactions.

In the remainder of this paper, we begin with the related work in Chapter 2, discussing and analyzing the current popular methods for depicting hierarchical data. Chapter 3 discusses the characteristics of traditional icicle plot and describe how we modify it and develop a dynamic visualization, Cascading CurtainMap, which can be used to represent large and flexible hierarchies. In Chapter 4, we discuss the application of our system in real cases, and Chapter 5 concludes our work and discusses the possibilities for future work.

# CHAPTER 2

## RELATED WORK

In representing hierarchies, node-link diagrams resemble the structure of a tree. Lines are used to connect nodes with parent-child relationships. Nodes with sibling relationships are located on the same level. Figure 4 displays a typical tree that uses node-link scheme:

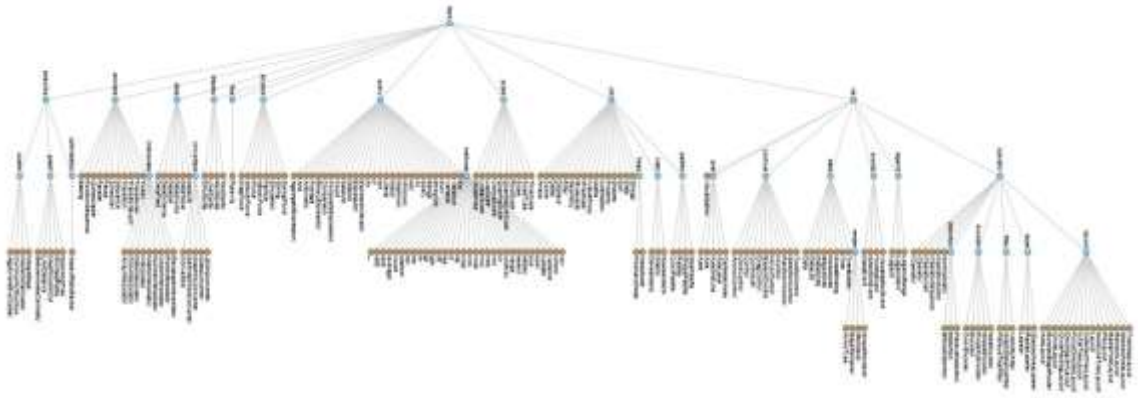

*Figure 4.* A Traditional Node-Link Diagram with Minimum Wasted Space Generated by the Reignold-Tilford Algorithm (1981).

The layout of the tree in Figure 4 conforms to the esthetics introduced by Wetherell and Shannon (1979) that are used to define a “tidy” drawing of a binary tree, which makes it capable of utilizing space efficiently without losing aesthetical properties. However, there still exist aesthetic issues if the nodes of this tree are not well distributed (Reingold & Tilford, 1981). Spacing issues will also emerge as the tree grows big in size due to redundant unused pixels (Van Wijk & Van de Wetering, 1999), which prevents it from being used to display large hierarchies within available screen space and causes difficulty in understanding the hierarchical structure. In reality, it is common that the hierarchical structure of the information is composed of a large number of levels and that

each node has its own sub tree, in which case the tree will grow really big and people can easily get lost when reviewing such a huge structure. Basically, most solutions to the spacing issues of traditional node-link tree fall into three categories: reconfiguration of nodes and links, introduction of extra dimension, and dynamic diagrams.

**Rearrangement of Nodes and Links**

By reducing the space between non-leaf nodes, Burch, Raschke, and Weiskopf (2010) proposed indented pixel tree plots, which uses vertical lines to represent non-leaf nodes and dots to represent leaf nodes. As we can see in Figure 5, the span of the leaf nodes is reduced by putting them together in the form of a series of dots, thus leaving extra space for the parent nodes. It is a space-condensed version of the traditional tree.

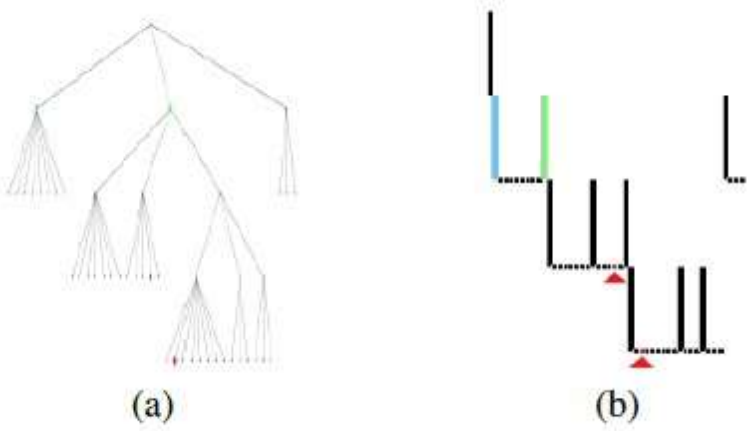


*Figure 5.* A Node-Link Diagram and its Corresponding Indented Pixel Tree Plot (Burch et al., 2010).

In a similar way, a hyperbolic tree (Lamping, Rao & Pirolli, 1995) rearranges its nodes and links to solve the spacing problem. Figure 6 shows a hyperbolic tree in which all the leaf nodes (colored orange) are located on the circumference and their parent nodes are inside the circle. As the number of hierarchical levels increases, the circle will

grow bigger accordingly and hence provide more space for the leaf nodes. Additionally, instead of using Cartesian coordinates, hyperbolic tree uses polar coordinates, which leads to an aesthetically pleasing layout, while using space efficiently.

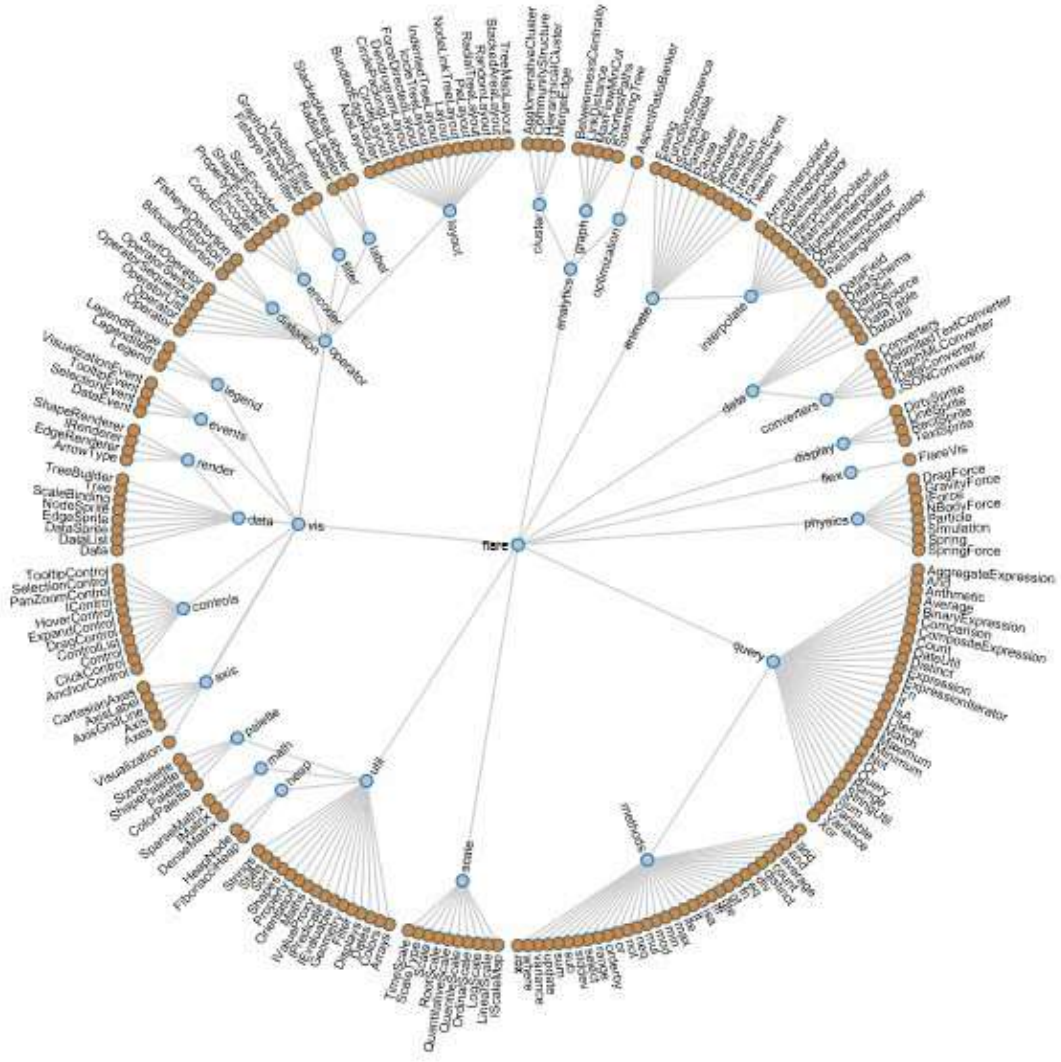

*Figure 6.* In a Hyperbolic Tree, all Leaf Nodes are located on the Circumference and their Parent Nodes are inside the Circle.

## Introduction of Extra Dimension

In order to deal with the spacing issue in representing large hierarchical structures, Robertson, Mackinlay, and Card (1991) introduced the cone tree that utilizes a

third dimension, the depth, to fill the screen with more information. Figure 7 shows a dynamic 3D cone tree that provides interactions such as rotation, selection and highlighting.

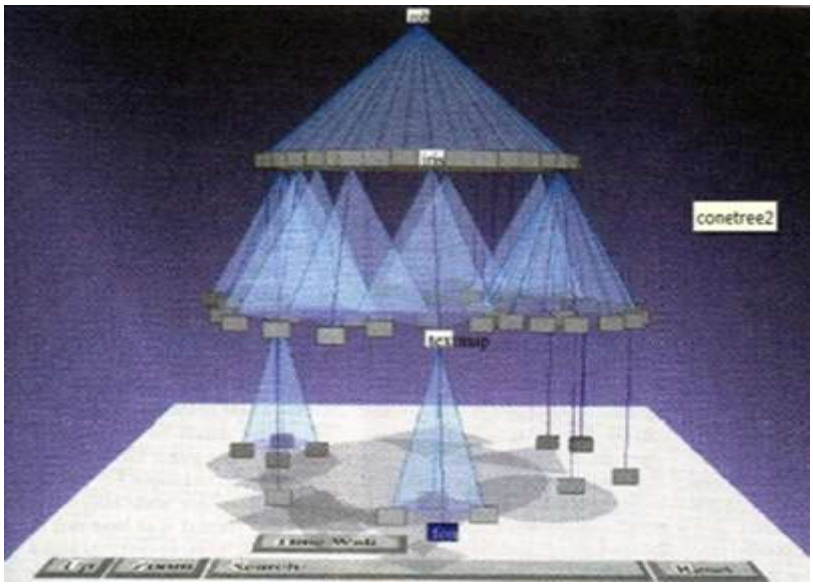


*Figure 7.* In a Cone Tree (Robertson et al., 1991), a Third Dimension is Introduced to Convey More Information while Providing Interactions such as Rotation, Selection and Highlighting.

In the 3D space of a cone tree, all the child nodes are laid out in a cylinder below their parent node. In such a cylinder, since only the nodes that are brought to the front are visible to the users and other nodes are hidden in the back, it provides extra space for the other sibling nodes. When the user selects a node, in order to minimize the perceptual complexity caused by a third dimension, the cone tree rotates so that the selected node and each node in the path between the selected node and the top (root) node are brought to the front and highlighted. In such a dynamic way, a tree conveys more information.

A third dimension enables a cone tree to display more information, but according to Cockburn and McKenzie (2001), "When using Cone-Trees the subjects took

significantly longer to locate files, and their efficiency deteriorated rapidly as the branching factor of the hierarchy increased.", which might be due to the fact that it usually takes longer for human being's brain to assimilate a structure in 3D than in 2D and they are more difficult to understand. A graph, being different from text, should require as minimum explanation as possible. If users take too long to read it, it will gradually increase the cognitive load and be difficult to draw meaningful conclusions. Therefore, the introduction of 3D diagram might artificially complicate the scene without adding any useful value.

## Clevenland's Hierarchy

In our CurtainMap, we use length to encode item values (See Figure 21). It is different than the encoding used in a treemap or in a sunburst diagram, which use area to encode item values. According to "Cleveland's Hierarchy" (Cleveland, 1985), attributes used for graphing data are ranked from worst to best. They are "Color", "Volume", "Area", "Angle/Slope", "Length", "Position along nonaligned scales" and "Position along a common scale", in which "Length", which is an one-dimensional attribute, is better than "Area", which is a two-dimensional attribute. That's why some people prefer to assign only one dimension of an object instead of two. A typical example is the histogram, in which the bars have same width but have different height.

Treemaps are typical diagrams that use area to encode item values, in which the children items are drawn inside their parent item with different size (area). In a treemap, people need to look at both width and height to determine the area, whereas in our CurtainMap, the items on the active layer (See Figure 21) have the same width, so people

only need to make comparison between items based on only one dimension, the height, therefore our CurtainMap is better than the treemap according to Cleveland's theory.

## Use of Dynamic Diagrams

Instead of statically displaying all hierarchical information in the same time, Plaisant et al. (2002) Introduced a dynamic way to represent the hierarchical structure. Figure 8 shows a Spacetree (Plaisant et al., 2002) that provides users the freedom to fold / unfold a non-leaf node.

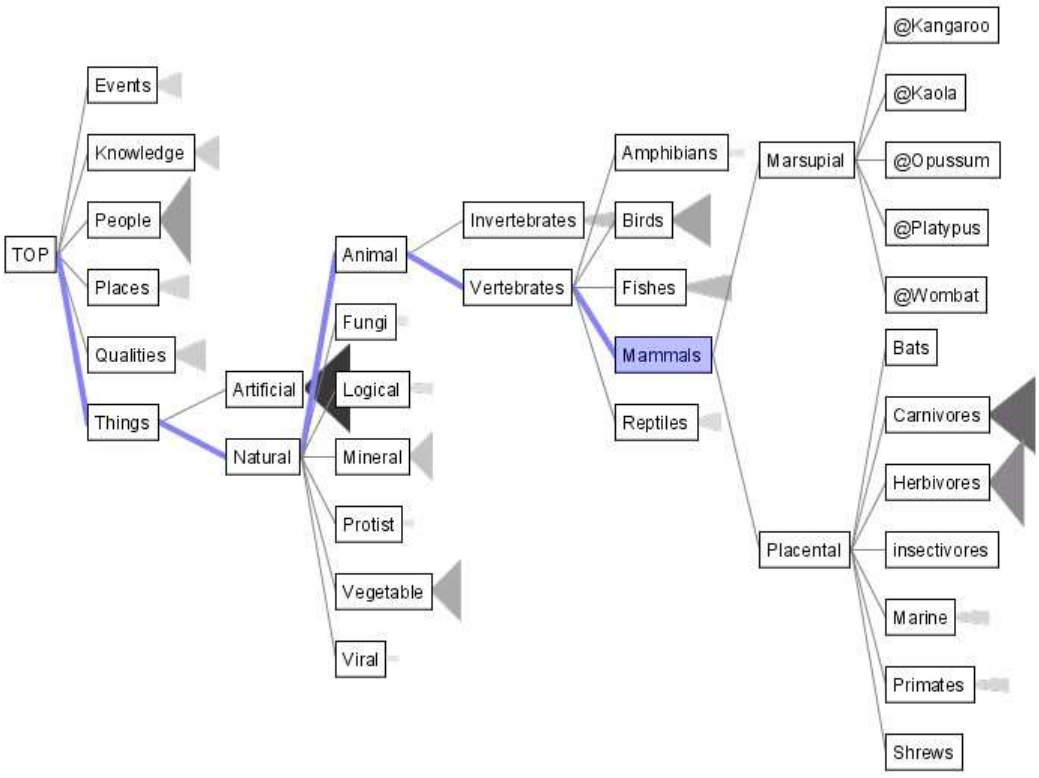


*Figure 8.* Example of a Space Tree (Plaisant et al., 2002).

As we can see in Figure 8, the triangular icon indicates that the node beside it has a sub tree and users can choose whether or not to review its child nodes according to their demand. With this flexibility, a space tree doesn't need to display the entire hierarchical structure all at once, thus providing more space for the currently visible nodes without

introducing a third dimension. In such a way, people could concentrate on the data items they are interested without getting distracted by the huge hierarchical structure.

Other than space tree, the Hyperbolic Browser (Lamping & Rao, 1996) is another 2D dynamic diagram. The Hyperbolic Browser uses hyperbolic geometry for the layout and enables the user to bring any node into focus at the center region where the items can be magnified.

### Limitations and Shortcomings of Node-Link Diagrams

Most of the time, the structure of the hierarchical data is not only what people want to know. More importantly, they also care about the value of the data item and the contribution each data item makes among the total. In node-link diagrams, since data are represented in a tree structure in which the nodes are separated with each other, it is difficult to make comparisons between them. Moreover, links are the major causes of the redundant blank space. When it comes to data composed of a large number of levels, the redundant space will increase exponentially.

### Space-Filling Diagrams

Based on the issues discussed above, we have to consider another scheme, space-filling diagrams, as an alternative. Space-filling diagrams are better than node-link diagrams in dealing with the following issues.

Representation of item values: In space-filling diagrams, since data items are generally represented in terms of rectangles that are located closely with each other, with the width, length or area indicating the values they represent, it is easy to learn about the difference of item values and to make comparison between them, whereas in node-link

diagrams, nodes are separated by links and most of node-link diagrams are not capable of representing item values.

Spacing: In node-link diagrams, a link indicates a parent-child relationship, whereas in space-filling diagrams such as treemaps, such relationship is indicated by dividing the rectangle of the parent node, which doesn't require any redundant space. However, the price it has to pay is that the rectangles of the child nodes will become smaller and smaller as the number of times of division increases.

While existing lots of variations, some commonly used space-filling diagrams for representing hierarchical data include Treemaps, Icicle Plot, and Sunburst. All these diagrams have their own advantages and shortcomings.

**Treemaps**

In early 1990s', Johnson and Shneiderman (1991) presented a visualization technique that utilizes 100% of the available display space, called Treemaps, representing the full hierarchy on a rectangular region in a space-filling manner. The treemap algorithm recursively subdivides area into rectangles, which is a powerful space-filling diagram in representing hierarchical data. Figure 9 shows a typical treemaps representing soft drink preference in a small group of people, where color and gradients are used to group items.


*Figure 9.* Example of a Typical Treemap.

Below is some typical characteristics of treemaps:

1. Children are drawn inside their parent;
2. Alternate horizontal and vertical slicing at each successive level;
3. Use area to encode other variable of data items.

Using treemaps, it is convenient for viewers to understand the hierarchical structure of the data and to make comparison between the data items on the same level. One problem of treemaps is that child nodes are drawn inside parent nodes, it keeps users from specifying a parent node and its child nodes in the same time. In the treemaps shown in Figure 10, we are trying to specify two nodes, “mobile phone” and “mobile phone priced between $1000 and $2000”, which have parent-child relationship, to indicate that they are both problematic, but it turns out to be the entire “mobile phone”

that are in red. Such two problematic data items are not clearly expressed by the treemaps.

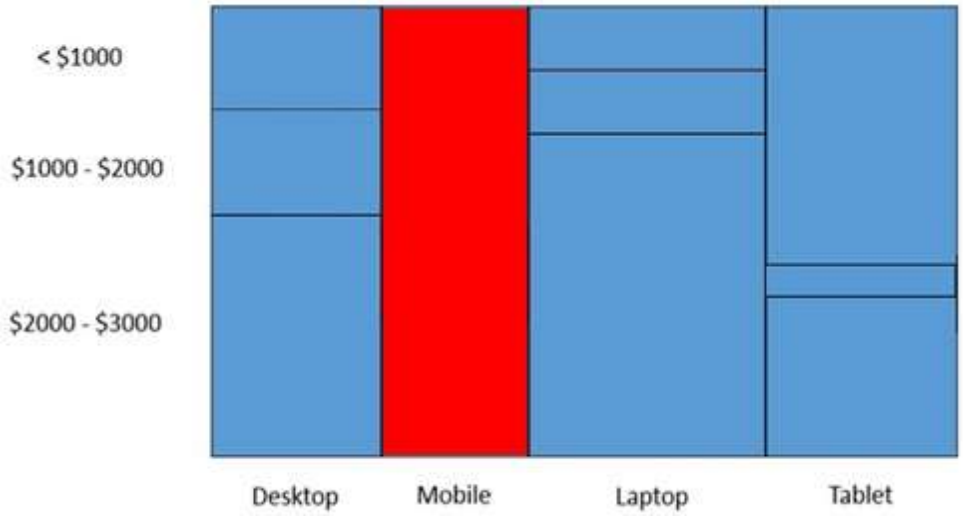


*Figure 10.* Treemaps are Confusing when Used to Specify a Node "Mobile" and its Child Node "Mobile Priced Between $1000 and $2000".

**Icicle Plots**

This issue can be solved by using icicle plot. In an icicle plot (Kruskal & Landwehr, 1983), since objects belonging to a cluster are drawn under parent cluster, we can easily specify both parent node and child nodes in the same time. The icicle plot in Figure 11 highlights the same two problematic data items with parent-child relationship that we failed to specify in treemaps in Figure 10.


*Figure 11.* An Icicle Plot (Kruskal & Landwehr, 1983) can be Used to Effectively Specify a Parent Node and its Child Node in the Same Time Without Confusion.

However, as we discussed in the introduction, traditional icicle plot is not capable of displaying large hierarchies since they are static diagrams. As the number of levels in a hierarchy increases, the leaf nodes will end up being indistinguishable. The main cause for this problem is that the width of nodes at each lower hierarchical level keeps being subdivided without a limit, which leads to the indistinguishability of leaf nodes. The width of a node on a specific level i can be calculated using the formula below: NodeWidth (i) = RootWidth * $\prod_{k=0}^{i} C(k)$ (where C(k) is the contribution the item makes over its siblings).

We can see in this formula above that the RootWidth is multiplied by a series of values C(k) that are less than 1. Such continuous multiplication makes it smaller and smaller as i increases.

**RSF Visualizations**

Instead of subdividing the width of the root node repeatedly, various Radial, Space-Filling (RSF) visualizations were born and lots of their variations were developed to display large hierarchies for their own purpose. Figure 12 shows a typical RSF diagram.

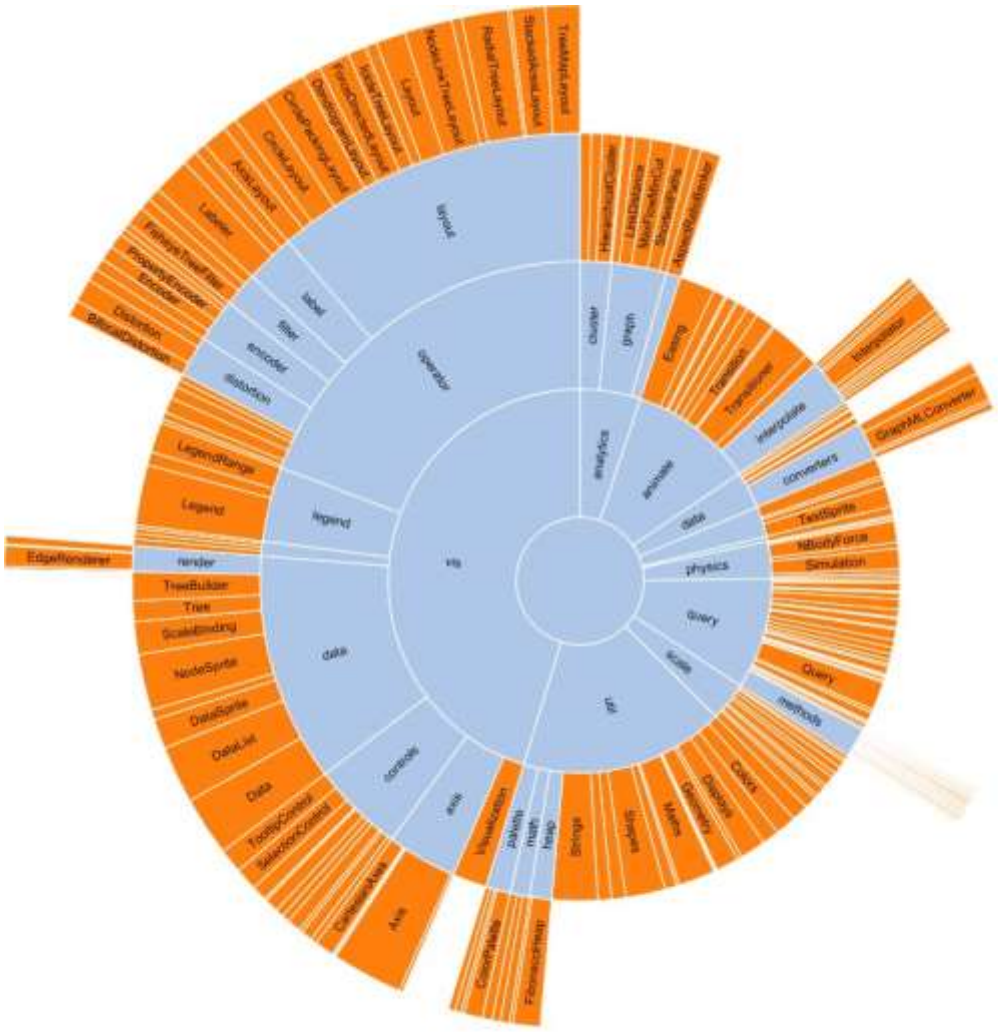


*Figure 12.* An Example of Radial Space-Filling (RSF) diagram.

**DocuBurst.** DocuBurst (Collins, 2007) is a RSF diagram for visualizing document content. It represents in the form of a radial space-filling diagram the hyponymy of a given word in the document with sub-trees indicating words that are the subcategories of their parent-tree. It enables interactive techniques such as filter, zoom and details-on-demand. As shown in Figure 13.

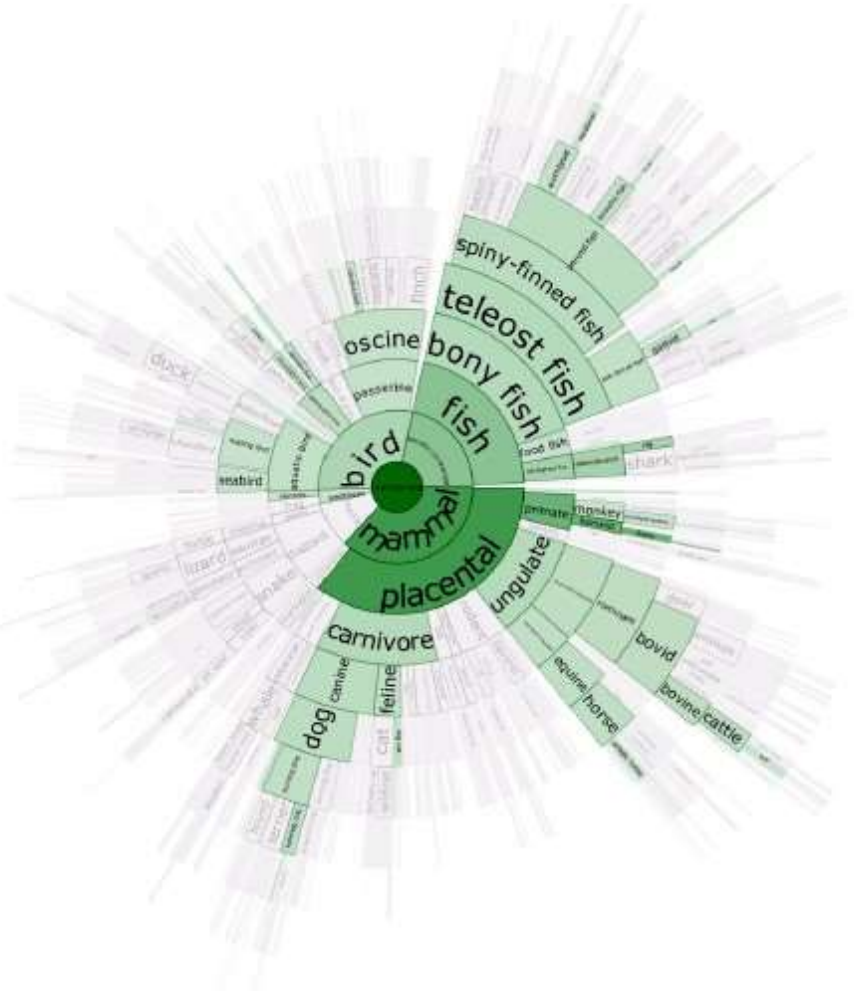


*Figure 13.* Hyponymy of the Word "vertebrate". Sub trees with words occurring in the reference document are green with opacity directly related to strength of occurrence.

**Information slices.** Information Slices (Andrews & Heidegger, 1998) is a dynamic RSF visualization of hierarchies for representing the OS directory structures and is used for disk files management. It uses multiple semicircular discs to visualize large hierarchies of directories and files. Figure 14 shows a case in which a single disc (left-hand disc) is not enough to represent the hierarchical structure and hence the visualization expands into another disc (right-hand disc) for deeper hierarchies. In such a dynamic way, it is capable of visualizing a large number of hierarchical levels.

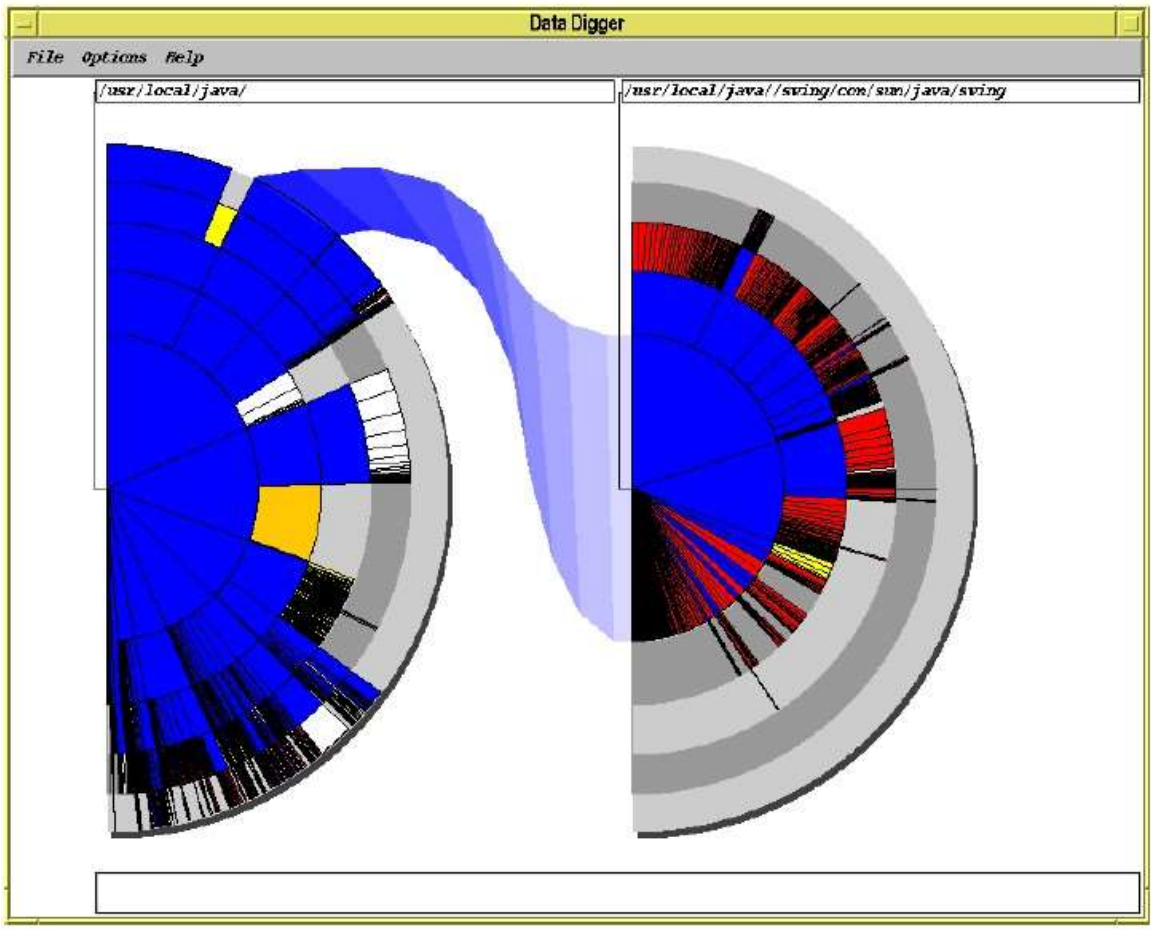


*Figure 14.* Expanding the "swing/com/sun/java/swing" subdirectory into the Right-Hand Disc.

**Sunburst.** Similar to what we discussed in the icicle plot before, as the hierarchy levels increase, these RSF diagrams still suffer from the same kind of problem - it will be difficult to distinguish the peripheral nodes. For solving this problem, Sunburst (Stasko & Zhang, 2000) enables users to interact with the items (including both peripheral nodes and inner nodes) and guide them to a more focused display for detailed examination. It can enlarge items on the periphery in different ways while maintaining an overall view of the entire hierarchical structure.

**InterRing.** With regard to the issues such as improper animations, space usage and multiple foci existing in Information Slice approach (Andrews & Heidegger, 1998) and Sunburst (Stasko & Zhang, 2000) visualization, InterRing (Yang, Ward, Rundensteiner, & Patro, 2003) was developed to address these problems. Instead of using extra space for detailed examination of focused items, it enlarges the selected items and performs proper distortions on the original diagram itself to minimize the loss of understanding of the hierarchical structure during the animation and to enable multiple foci. Moreover, InterRing provides interactive techniques such as Drill-Down/Roll-Up, Zooming and Panning, Rotation, and Modification of the hierarchy, which makes such system more flexible and robust in dealing with hierarchical structures.

However, as we can see in these diagrams, they are all used to represent hierarchies in which parent-child relationships are all strictly fixed, which are fixed hierarchies. While these diagrams are efficient in visualizing such type of hierarchies composed of large amounts of levels, we also want them to keep their original benefits when used to represent the flexible hierarchical data. Since there is no parent-child relationship between any levels (attributes) for such flexible hierarchical data (e.g. considering “price”, “brand”, and “popularity” of laptops), we can switch any two levels in such a hierarchical structure while maintaining the logical correctness (See Figure 1 and Figure 2), which necessitates another requirement – we want to visually and effectively modify the order of classifications applied on the data.

Although Information Slices and Sunburst can solve the issues in displaying large hierarchies in certain way, they don’t provide any solutions for modifying the hierarchy structure. The drag and drop modification operation provided in InterRing solves the

problem in which unrelated items share the same cluster, but it cannot modify the hierarchy based on the entire level.

## Visualizations for Flexible Hierarchy

By taking the order of classifications into consideration, Chintalapani, Plaisant, and Shneiderman (2004) extended the utility of traditional treemaps and make them capable of visualizing flexible hierarchy.

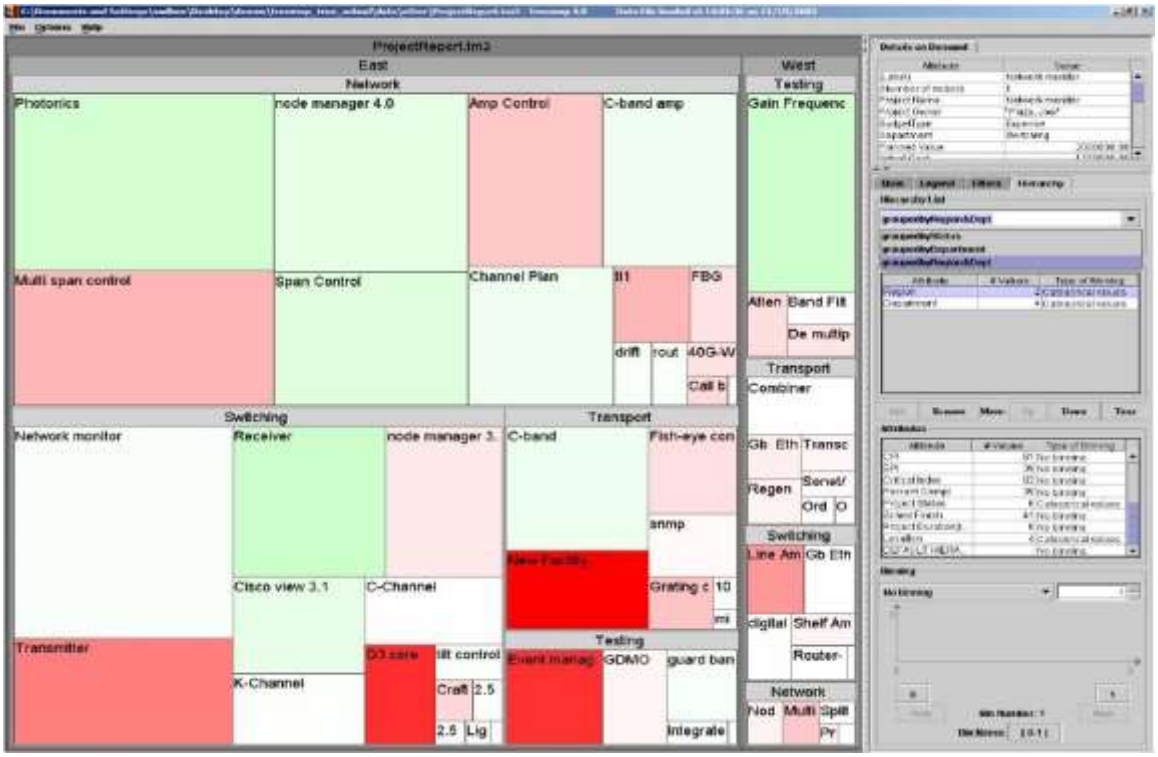


*Figure 15.* Visualization of Projects, Grouped First by Region, and Then by Department.

Figure 15 shows the system they developed. On the left-hand section is a traditional treemap that applies the categorical attributes in a specific order. On the right-hand section is a control panel with a "Hierarchy" Tab that enables users to specify the hierarchy. Users can select attributes from the "Attribute" table and add them to the

“Hierarchy “table to produce a specific hierarchical order. They can also save such order in the “Hierarchy List” above for future use. See Figure 16.

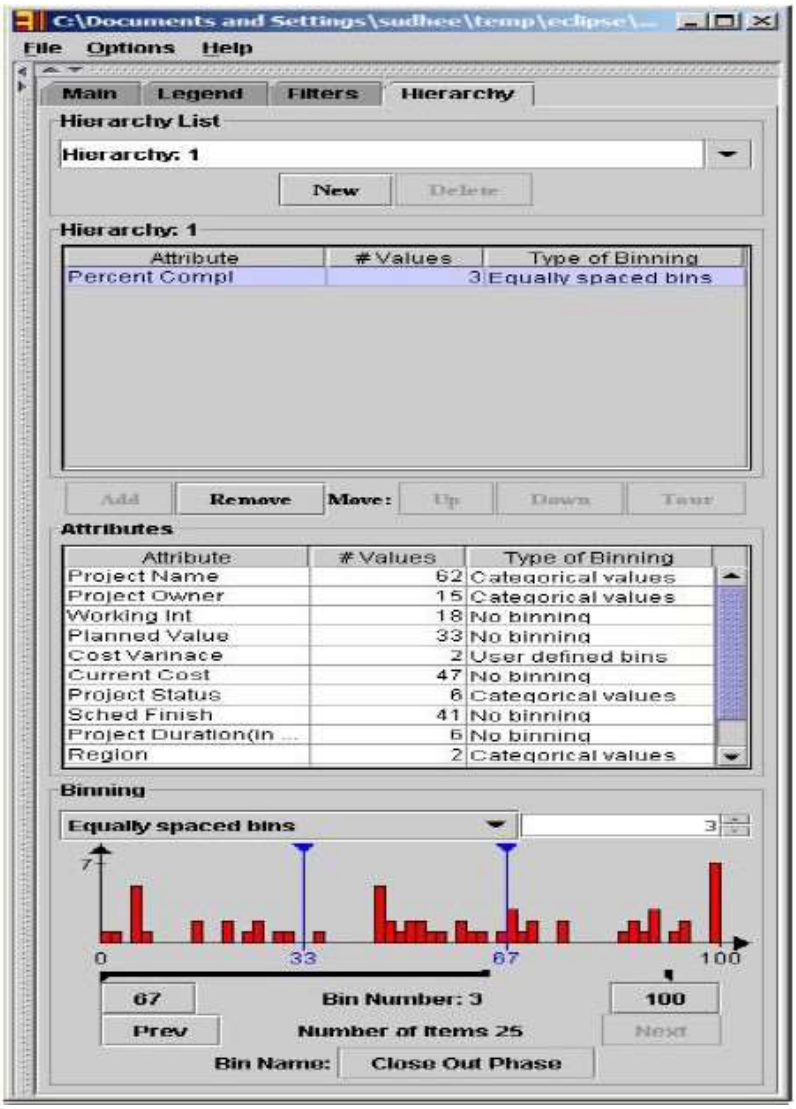


*Figure 16.* Control Panel with Hierarchy Tab that Enables Users to Specify the Hierarchy.

Other functionalities include enabling the user to control the hierarchy depth, to select layout algorithms, and to modify the color, size and label for the items in the treemaps.

CatTrees (Kolatch & Weinstein, 2001) is another visualization based on treemaps that takes categorical data and enables the user to modify the order of attributes added to the hierarchy. Figure 17 shows the CatTrees system.

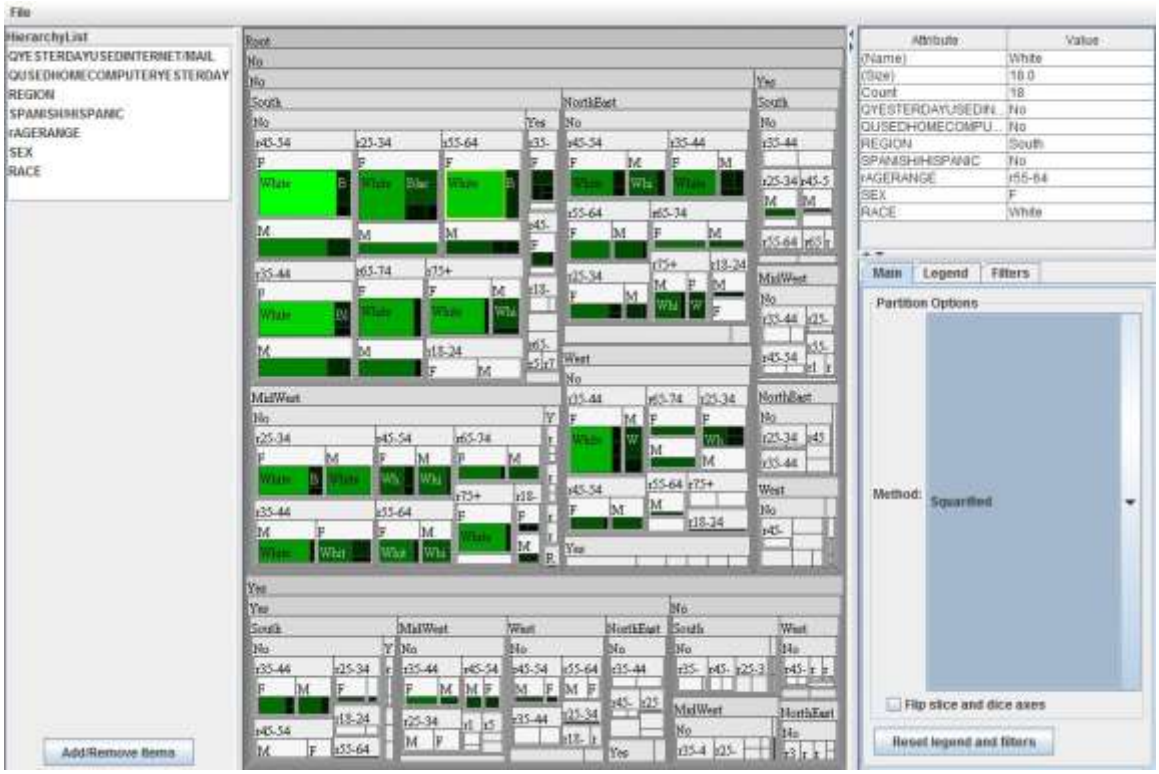


*Figure 17.* An Instance of CatTrees Visualization.

In the "HierarchyList" on the left-hand section, users can change the hierarchical order by dragging items to a different position. When such change happens, the treemaps in the middle part will be restructured accordingly. It enables the user to add or remove items in the "HierarchyList" to simplify or enrich the hierarchy structure. In the treemaps in the middle, user can select a leaf node or an aggregated node, meanwhile the values of the attributes from root to the selected node will be shown in the table on the upper-right section. In the lower-right tabs, CatTrees provides partition options" Squarified" and

"Slice and Dice", functionalities for changing label, color and size of the items, and enables users to define the maximum depth of the hierarchy.

We felt that both the extended Treemaps and CatTrees (Chintalapani et al., 2004; Kolatch & Weinstein, 2001) could be used to effectively display flexible hierarchies, but they are not without their drawbacks.

1. As we discussed before, since the child nodes in treemaps are drawn inside their parent node, it is difficult to specify both parent node and its child nodes simultaneously;
2. While efficient in dealing with flexible hierarchies, the extended Treemaps and Cattrees (Chintalapani et al., 2004; Kolatch & Weinstein, 2001) might lose efficiency in visualizing large hierarchies. Different with the Focus+Context technique introduced in Sunburst (Stasko & Zhang, 2000) and multi-focus distortions introduced in InterRing (Yang et al., 2003) in visualizing details of the items that are deep within the hierarchy, the Zooming In/Out technique in extended Treemaps and Cattrees simply enlarges the selected item and cause its context to be lost, which is difficult for users to make comparison between selected items and their context.
3. In order to specify a different hierarchy, users need to drag items in a text list when using the Cattrees (Kolatch & Weinstein, 2001) or to add attribute one by one into the hierarchy table when using the extended Treemaps (Chintalapani et al., 2004), which is time-consuming, and not user-friendly.

4. When transiting between different hierarchies, they both lack proper animations, which is important in order to maintain an overall understanding of the hierarchical structure.

# CHAPTER 3

# CASCADING CURTAINMAP

Based on these issues above, and combined with techniques in space tree and traditional icicle plot, our Cascading CurtainMap for visualizing flexible hierarchies has the following advantages:

1. Capable of effectively displaying large hierarchies using "cascading interaction" technique;
2. Capable of displaying flexible hierarchies in a user-friendly way;
3. Enable users to main an overall understanding of the hierarchical structure when changing the hierarchical structure by introducing animation;
4. Enable users to make comparison between different items more easily by modifying the layout of traditional icicle plot.

## Modification of Traditional Icicle Plot

As we discussed in the introduction, the "left-to-right and top-to-bottom" orientation layout (Barlow & Neville, 2001) makes icicle plot a better visualization than other diagrams for displaying hierarchies. Therefore, we keep such characteristic in our visualization. In traditional icicle plots, items (clusters) are represented by rectangles. One major drawback of traditional icicle plot is that it uses horizontal distance — the width, to represent the values of the items but all items at a specific level start with different X coordinates. Such "inconsistency" makes it difficult to see the difference in the values of the items and to make comparison between them, especially when the difference is small.

**Item Height Modification**

In our visualization, instead of using horizontal distance to represent values, we use vertical distance — the height to do so. In such a way, all items at a specific level start with the same Y coordinate, which makes it easy to see even slight difference among the items. Figure 18 shows the traditional approach icicle plot uses to represent two items with 5% difference in the width, but it is difficult for us to distinguish which item is longer and which item is shorter. Such inaccuracy leads to the difficulty in making comparison between multiple items. Figure 19 shows our approach that uses the height to represent the same items as shown in Figure 18. We can easily distinguish the difference between such two items with slight value difference.

*Figure 18.* The Width of the Left Rectangle is 5% Shorter than the Width of the Right Rectangle.

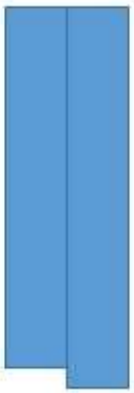

*Figure 19.* By Using Vertical Length to Represent Item Values and Aligning Items on Y Axis, it is Easy to See the Difference and Make Comparison Between Items.

The strength of our approach will be more obvious if there are more than two items with similar values. In Figure 20, the left diagram shows the traditional approach icicle plot uses to represent multiple items with slight difference in their values. When represented horizontally, all these items seem to have equal width. The right diagram uses height to represent these items. When represented vertically with aligned Y coordinates, it is easy to see the difference between these items. Notice that the layout of items in the right diagram resembles an inverted histogram with its bars grow downwards, we call this approach an "inverted histogram" approach.

*Figure 20.* In the Left Diagram, the Items with Different Values Seem to Have Equal Width; in the Right Diagram, the Same Items are Represented in a Vertical Way that it is Easy to See Their Difference.

**Active Layer and Inactive Layer**

One problem that emerges from our "inverted histogram" approach to visualize the item values is that we cannot use such layout for all hierarchical levels. Since the items on a specific hierarchical level are represented as an inverted histogram, if there still are levels below the level that uses height to represent values, the positions of the items on the lower levels will be affected and cannot be aligned horizontally.

Therefore, we only use the “inverted histogram” approach for the items on the bottom level and we call this bottom level the “active layer” and call any level above it the “inactive layer”. For the inactive layers, we don’t use the “inverted histogram” approach to display the item values. Instead, we let them share the same height. In such a way, it is guaranteed that there is no level under the active layer and that all the levels above the active layer share the same height, which enables the items on the active layer to start at the same Y coordinate. Figure 16 shows such a layout our approach generates.

**Item Width Modification**

For the items on the active layer, since we use height to represent the items value, we don’t complicate the understanding of the value by differentiating the width. So users can easily learn about the value by concentrating only on one dimension. Instead, we use width for the items on the inactive layers to represent the number of sub items a cluster has. Therefore, the clusters with more child nodes are longer than the clusters with less child nodes. Figure 21 shows a hierarchical structure with three levels. Cluster A has seven items, cluster B has two items and cluster C has four items. A, B, and C have the same height but have different width due to the difference of sub items they have. Table 1 shows the usage of item height and item width in our visualization.

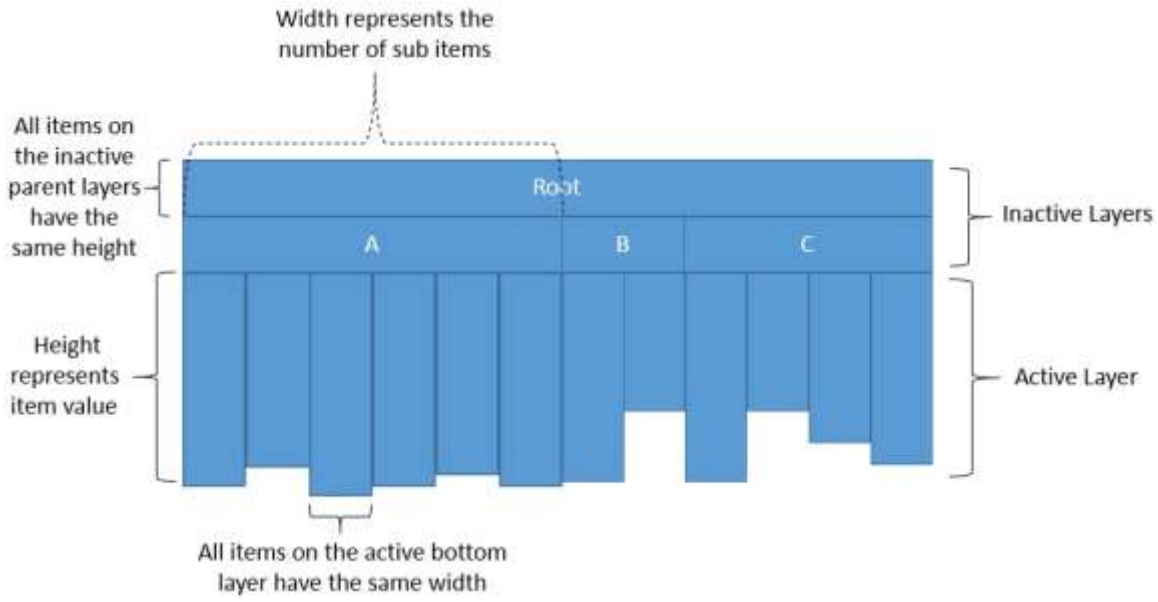


*Figure 21.* Width of Items on the Inactive Layers Represents the Number of Sub Items, whereas Height of Items on the Active Layer Represent the Item value. Items on the Inactive Layers Have the Same Height and Items on the Active Layer Has the Same Width.

Table 1

*The Usage of Item Height and Item Width*

| | Width | Height |
|---|---|---|
| Active Layer | Same | Represent Value |
| Inactive Layer | Represent the number of sub items | Same |

## Color Configuration for the Items

For the active layer, other than using height to convey the information of item values, we also use the as another display parameter because the combination of such two visual features, called “redundant representation”, can improve the understanding of the

values (Rheingans & Landreth, 1995). For categorical attributes that fall into data types such as Ordinal, Interval or Ratio data types according to Stevens, (1946) (e.g. “price”, “percentage”, or “age”), we use the sequential color scheme. For categorical attributes that are nominal data according to Stevens, (1946) (e.g. “product region”, “product type”, or “product color”), we use both the sequential and qualitative color scheme, and we provide a button for the user to switch between these two color schemes.

For the inactive layers, since the height remains the same all the time and they contain the information of the parent items of the lower items on the active layer, they actually serve as the X axis for the vertical bars of the active layer. Additionally, in order for the user to know about the value of the items on the active layer, a reverted Y axis with marks on it is positioned on the left of the active layer. Therefore, we have inactive layers as the X axis, Y axis, and the vertical bars (active layer) to be represented. In order for the user to easily distinguish the axes and the bars, we use different sets of colors for these different parts. We thus introduce the “opponent color theory” (Hurvich & Jameson, 1957), which presents three paired opponent colors that would be never perceived together: red versus green, blue versus yellow, and black versus white. If we use the sequential color scheme, then we use black and white to distinguish the axes and the texts on them, and use green and red to distinguish the normal items and problematic items on the active layer, which is shown in Figure 22. If we use the qualitative color scheme, we still use black and white to distinguish the axes and the texts on them, but we use another set of colors to display the items on the active layer and provide a legend to indicate which color represents which nominal attribute, which is shown in Figure 23.

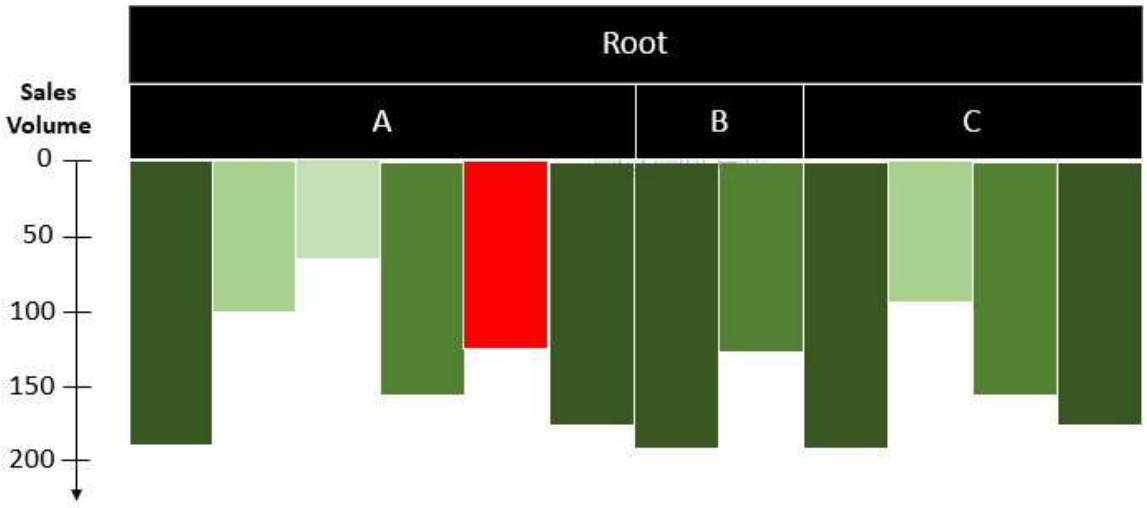


*Figure 22.* When Using the Sequential Color Scheme, the Inactive Layers (Colored Black) Serve as the X Axis for the Items of the Active Layer. A Series of Green Colors are Used for the Items on the Active Layer. Red is Used to Indicate the Problematic Item.

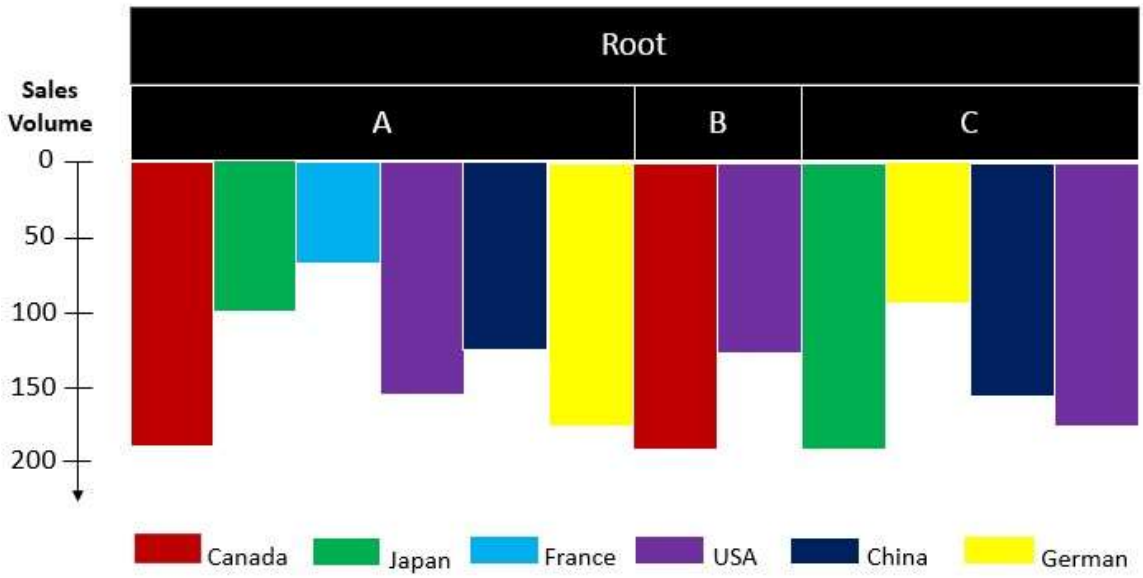


*Figure 23.* When Using the Qualitative Color Scheme, the Inactive Layers (Colored Black) Serve as the X Axis for the Items of the Active Layer. Another Set of Colors are Used for the Items on the Active Layer. A Legend is Provided at the Bottom.

## Dynamically Fold / Unfold Hierarchical Levels

The way the active layer uses to visualize items makes it easier for the user to review the item values. Therefore we provide a dynamic way for people to fold / unfold the hierarchy to make it possible for any hierarchical level to be the active layer. Suppose we have a hierarchy with four levels. Figure 24 is the complete unfolded layout of the hierarchy (The number of levels is labelled from 0 to 3 on the right of the diagram). Since level 3 is the active layer, it uses the height to represent the item values and level 0, level 1 and level 2 serve as the X axis for the active layer. Figure 25 shows the diagram after the user folds the hierarchy by one level using right-click. In this case, level 2 becomes the active layer. Level 0 and level 1 serve as the X axis for active layer. Figure 26 shows a similar case when the user folds the hierarchy by two levels and level 1 becomes the active layer. Only level 0 serves as the X axis for the active layer. On the other hand, users can unfold the hierarchy using left-click to see the items under the active layer. In such a case, the current active layer becomes an inactive layer and the new unfolded level becomes the current active layer.

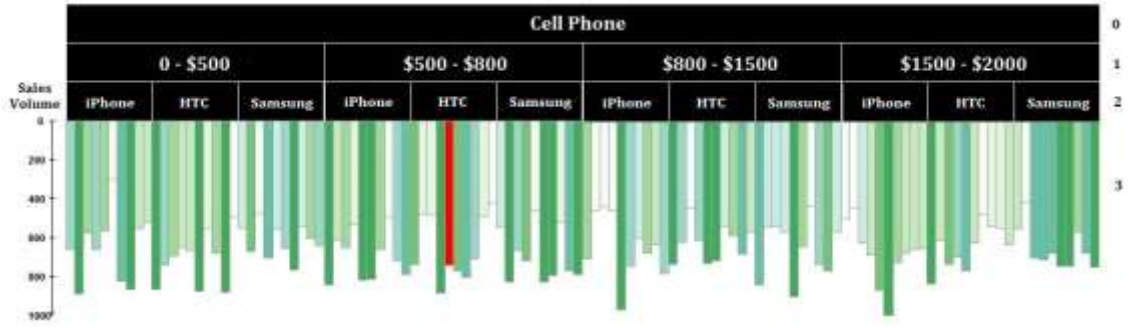


*Figure 24.* This is a Complete Unfolded Layout of the Hierarchical Structure with Level 3 (Colored Green) Being the Current Active Level. Level 0, Level 1 and Level 2 (Colored Black) Serve as the X Axis.

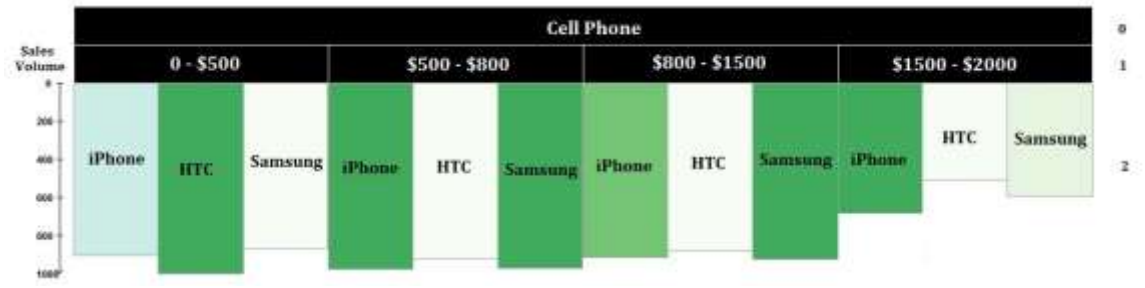


*Figure 25.* This is the Layout after Folding One Level from the Hierarchy in (a) with Level 2 Being the Current Active Level. Level 0 and Level 1 Serve as the X Axis.

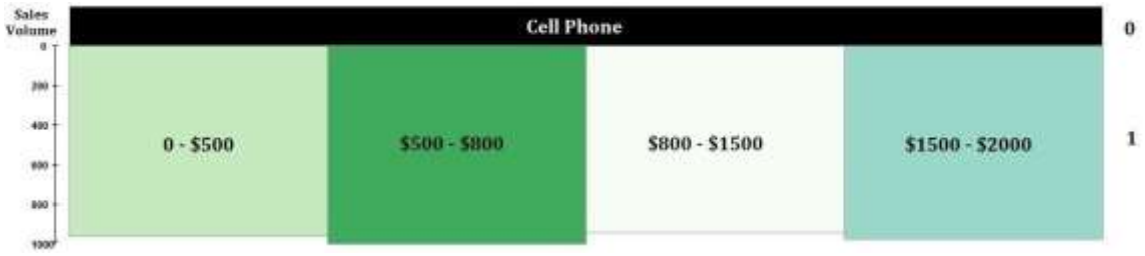


*Figure 26.* This is the Layout after Folding Two Level from the Hierarchy in (a) with Level 1 Being the Current Active Level. Only Level 0 Serves as the X Axis.

As we fold / unfold each hierarchical level, there is always an active layer appearing on the bottom which contains bars with different length, resembling a curtain with multiple folds, and there are always the inactive layers appearing above, resembling the rod of the curtain. The height of the curtain rod and the width of the curtain fold changes as we fold / unfold each hierarchical level. So, we call such visualization a "CurtianMap".

In additional to the Y axis that displays the value range of the items on the active layer, we also enable users to read the specific value of each item on the active layer through mouse hover. When the user places the mouse over an item, a message box containing the attribute name and value of that item will appear. This technique is particularly useful when the active layer contains lots of items and the width of each item

becomes really small. We also enable users to rearrange the positions of bars on the active layer based on their values. If the user clicks an item on the inactive layer right above the active layer, all its sub items will be rearranged from left to right in descending order (see Figure 27 for an example). In this way, it is efficient for users to study the values over a series of items and hence facilitates the decision making process.

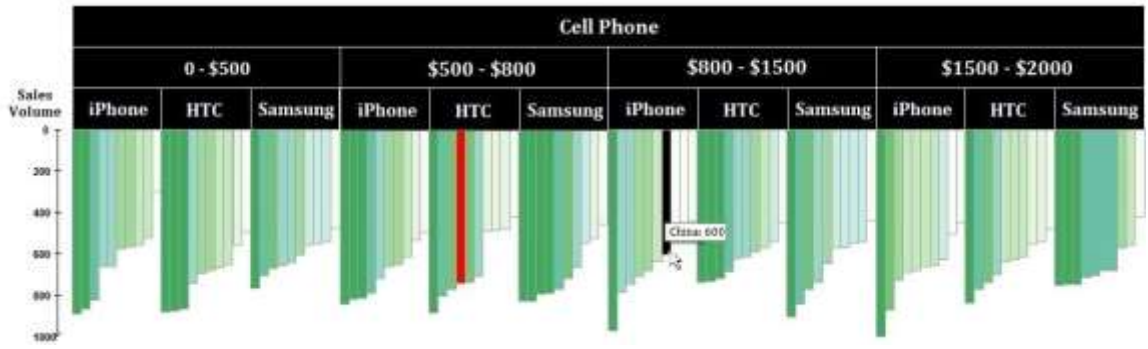


*Figure 27.* All the Items on the Active Layer are Rearranged in Descending Order by Clicking the Items Right Above the Active Layer. By Hovering an Item on the Active Layer, its Attribute Name and Value are Shown in a Message Box.

**Node-Link Structure**

As the hierarchy changes, in order for the user to maintain an overall understanding of the structure of the current hierarchy, we add a node-link structure to display the order of filters that have been applied to the corresponding hierarchy. When the user unfolds or folds the hierarchy, the node-link structure will change accordingly. Figure 28-30 displays different examples in which the node-link structure represents the order of filters that have been applied to the corresponding hierarchy.

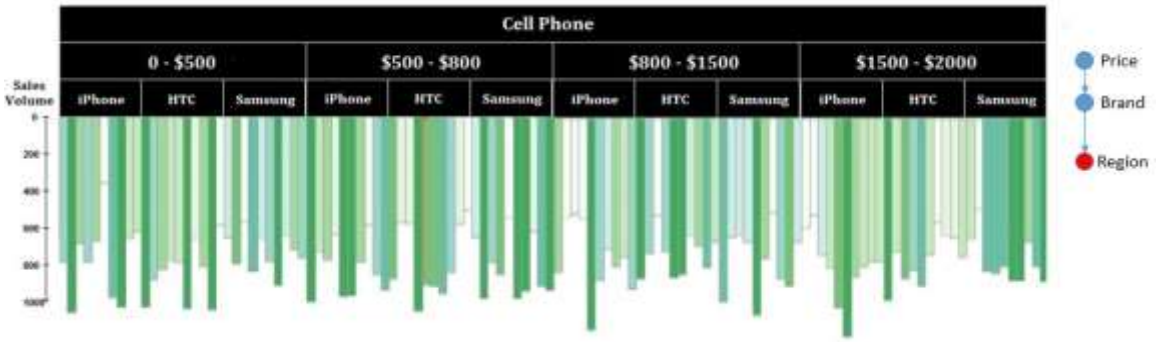


*Figure 28.* The Node-Link Structure Displays the Order of Filters Applied to the Corresponding Hierarchy: First "Price", Then "Brand", and Finally "Region".

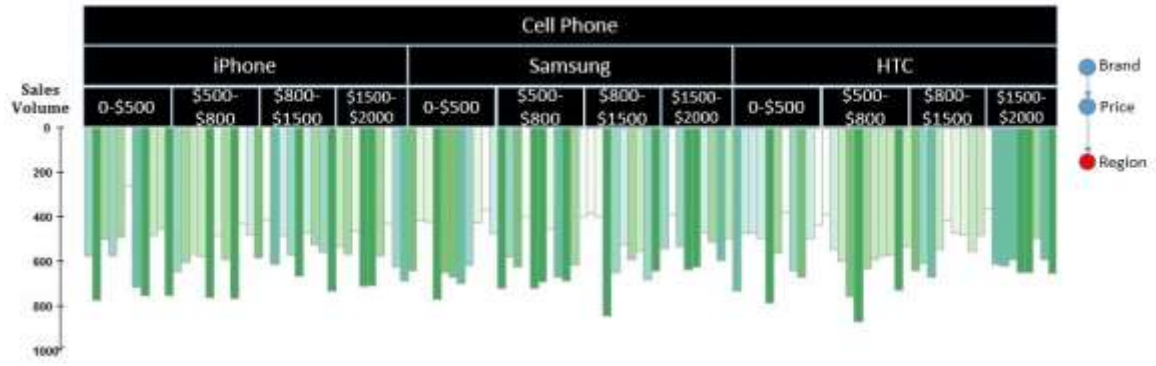


*Figure 29.* The Node-Link Structure Displays the Order of Filters Applied to the Corresponding Hierarchy: First "Brand", Then "Price", and Finally "Region".

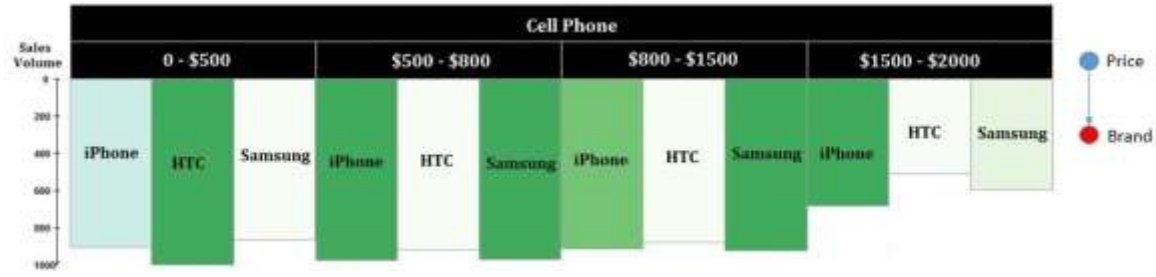


*Figure 30.* The Node-Link Structure Displays the Order of Filters Applied to the Corresponding Hierarchy: First "Price", Then "Brand".

### Flexible Change of Hierarchy

Other than enabling users to fold / unfold the hierarchical structure by right clicking and left clicking, we also provide user the freedom to change the categorical attribute being represented on the bottom level. By scrolling the mouse wheel, the categorical attribute used on the bottom level will change to another categorical attribute available in the data set and will keep changing in a loop if the user keep scrolling the wheel. In Figure 31, the categorical attribute used in the current bottom level is "price", after scrolling the mouse wheel forward by one step, such attribute changes to "phone type" (Figure 32). After scrolling the mouse wheel forward by one step further, such attribute changes to "years" (Figure 33). At this point, if the user continue doing the forward scroll, it will bring the categorical attribute - "price" back and return to the start point of the loop. Backward scroll is also available, but it loops through the attributes in a reverse order.

After the user select a categorical attribute for the bottom level and unfold one level further, the system will automatically select another categorical attribute for the new bottom level. In such a dynamic way, users can easily specify the order of categorical attributes applied on the data and examine the items they are interested. Compared to the "dragging" technique in Cattrees (Kolatch & Weinstein, 2001) and "select & add" technique in the extended Treemaps (Chintalapani et al., 2004), it is more user-friendly and less time consuming. Combined with the "fold / unfold hierarchical level" technique introduced in 3.2, users not only can see the aggregated items' values on high levels of the hierarchy, can examine the values in details on lower levels of the hierarchy, but also can flexibly change the hierarchy to see different attributes used on the bottom level

according to their interest, which conform to the standard of being an effective dynamic visualization for analyzing hierarchical data according to Senay and Saltz, (1997).

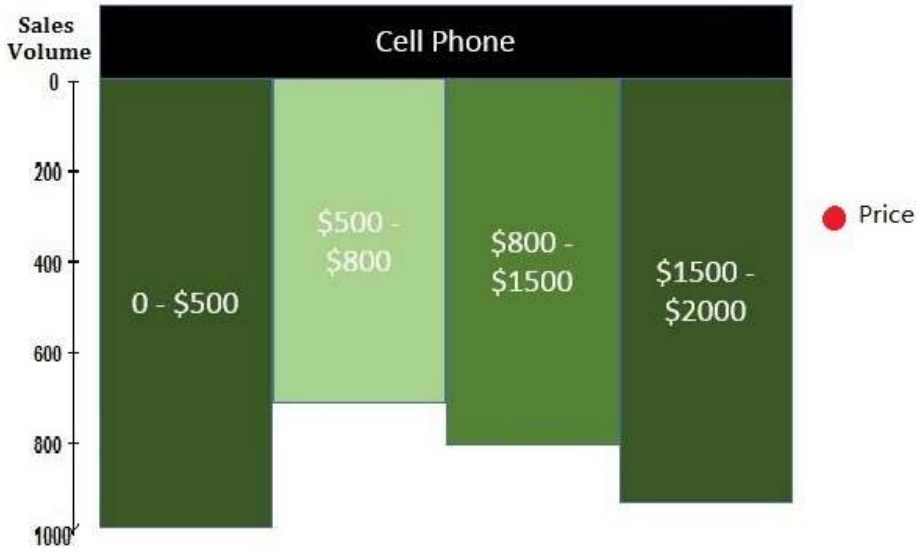


*Figure 31.* A Hierarchy with Two Levels with the Categorical Attribute - "price" Being its Current Bottom Level.

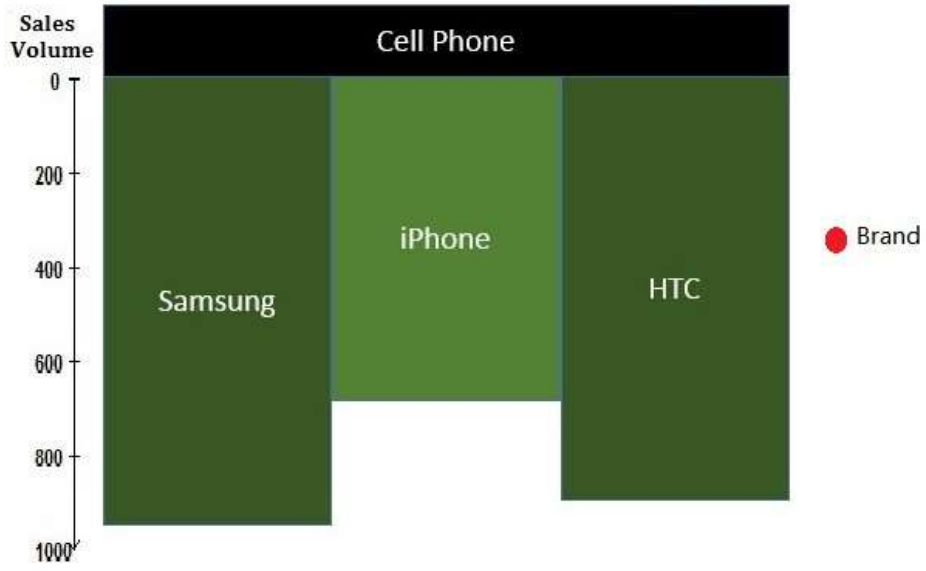


*Figure 32.* A Hierarchy with Two Levels with the Categorical Attribute - "phone type" Being its Current Bottom Level.

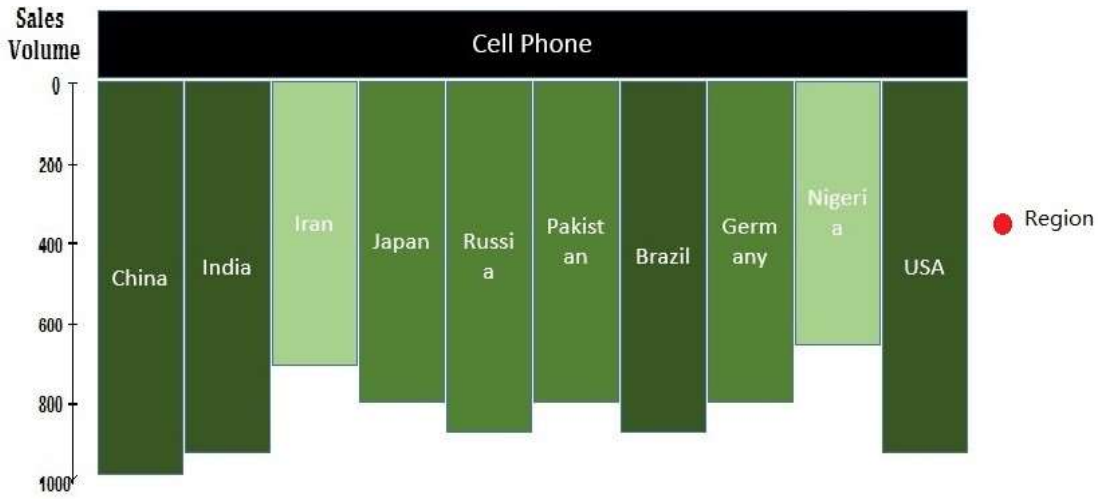


*Figure 33.* A Hierarchy with Two Levels with the Categorical Attribute - "years" Being its Current Bottom Level.

**Display Large Hierarchies using Interactive Technique**

Over decades the efficient visualization for displaying large hierarchies has always been an interesting and useful topic. As we discussed previously, the items on the active layer ends up being indistinguishable if the hierarchy keeps growing downwards, as shown in Figure 3. Different than the static visualizations that use limited screen space to display the entire hierarchy at once, interaction allows the user to review larger amounts of data according to Brath (1997). In our visualization, we maintain a threshold of the item width. At each step the user unfolds the hierarchy, we check the width of the items of the new active layer that will be shown. If the width goes below the threshold, we disable the unfolding operation. In such a case, instead of using unfolding operation, we let the user select the item of their interest and then the icicle plot will change accordingly to bring focus only to the selected item and its sub items.

**Focus an Item on the Active Layer**

Once an item on the active layer is selected with the mouse, such item and each item in the path from the selected item up to the top will be redrawn with the full width of the icicle plot, therefore leaving more space for displaying the sub items below the current active layer which otherwise would be indistinguishable. In Figure 34, the item "China" on the active layer is selected with the mouse, such item and each item in the path from it up to the top (red items) will be drawn with the full width of the icicle plot. Figure 35 is the result after the click operation, in which the selected item and all its ancestors (black items) are redrawn to occupy the full width of the icicle plot, and all other items are hidden, therefore leaving more space for displaying the new hierarchical level.

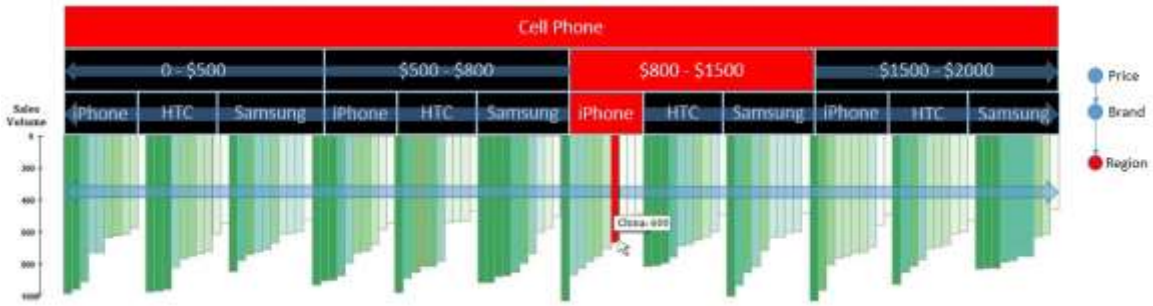


*Figure 34.* By Clicking an Item on the Active Layer, the Selected Item and Each Item in the Path from the Selected Item up to the Top (Red), that is Cell Phone, $800 - $1500, iPhone, and China, will be Drawn with the Full Width of the Icicle Plot.

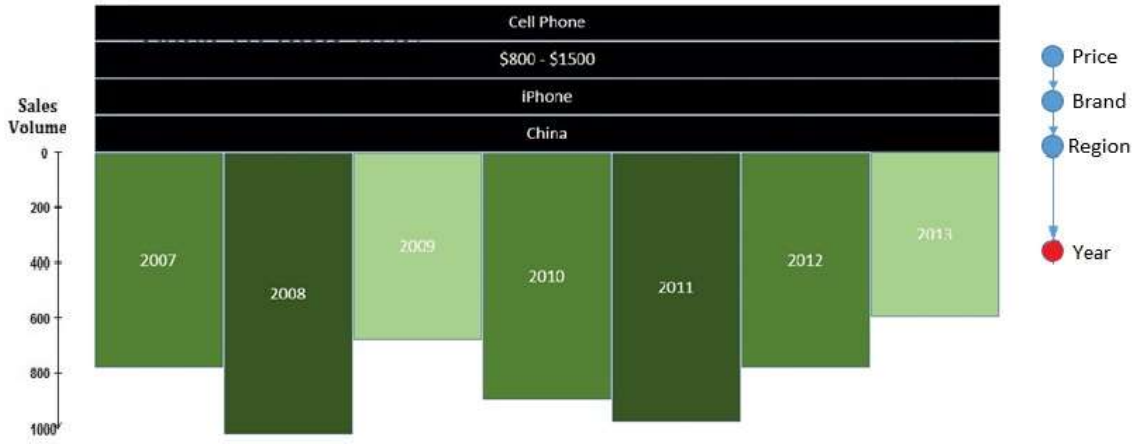


*Figure 35.* The Black Items that Occupy the Full Width of the Icicle Plot are the Selected Item and All its Ancestors in Figure 21. The Items on the Active Layer Belong to the Selected Item.

**Focus an Item on the Inactive Layer**

Additionally, we enable the user to select an item on the inactive layer. The selected item will occupy the full width of the icicle plot, and all its sibling items and their descendant items will be hidden. If the selected has already occupied the full width of the icicle plot, which means it has been focused, it will restore to its original size, and all its sibling items and their sub items will be shown. In both cases, if the width of items on a specific level go below the threshold, that level won't be shown. In Figure 36, the item "$800 - $1500" on the inactive layer is selected and Figure 37 displays how it looks like after the selection.

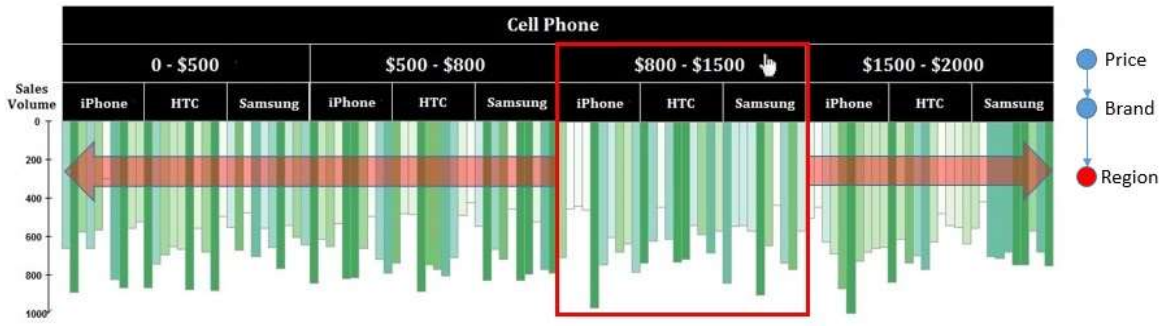


*Figure 36.* When "$800 - $1500" is Selected, the Selected Item and all its Sub Items Occupy the Full Width of the Icicle Plot, as Indicated by the Red Arrows.

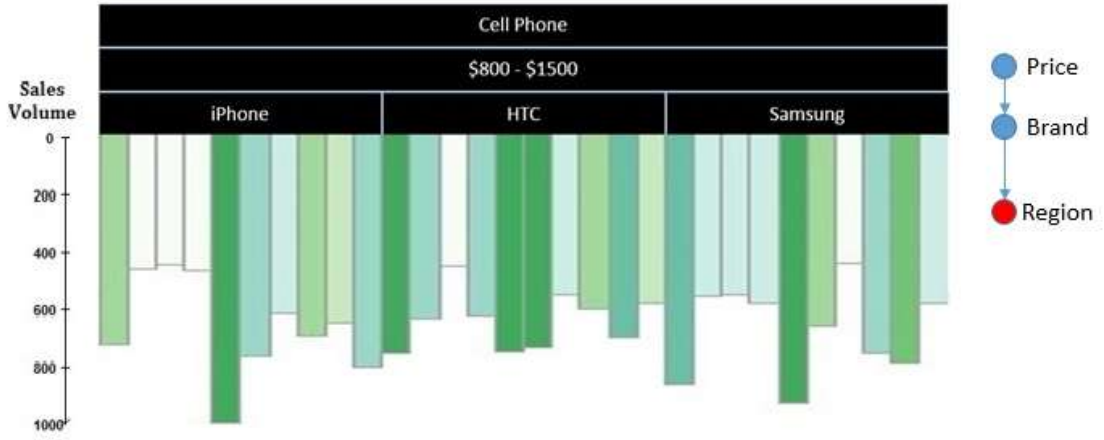


*Figure 37.* After the Selected Item and its Sub Items Occupy the Full Width of the Icicle Plot, All their Sibling Items are Hidden.

**Defocus the Item**

After an item is brought into focus, it will occupy the full width of the icicle plot. At this point, if the user selects (click) the focused item again, the width of both the selected item and its sub items will restore to their original width, and all their sibling items will be shown again. In Figure 39, if the item "$800 - $1500" is selected again, the structure will restore to the layout in Figure 38. Again, we don't display the items of which the width is below the threshold. Figure 39, 40, and 41 show how the diagram

looks like after the user respectively clicks “China”, “iPhone”, “$800 - $1500” in Figure 38.

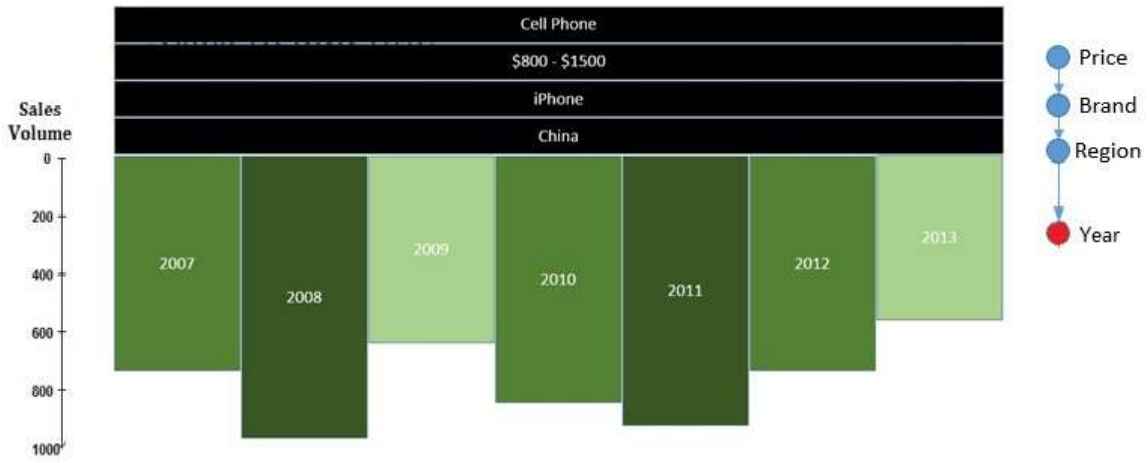


*Figure 38.* In this Status, the Items “$800-$1500”, “iPhone” and “China” All Occupy the Full Width of the Icicle Plot. It will Generate Different Layouts as the User Clicks Any One of Them.

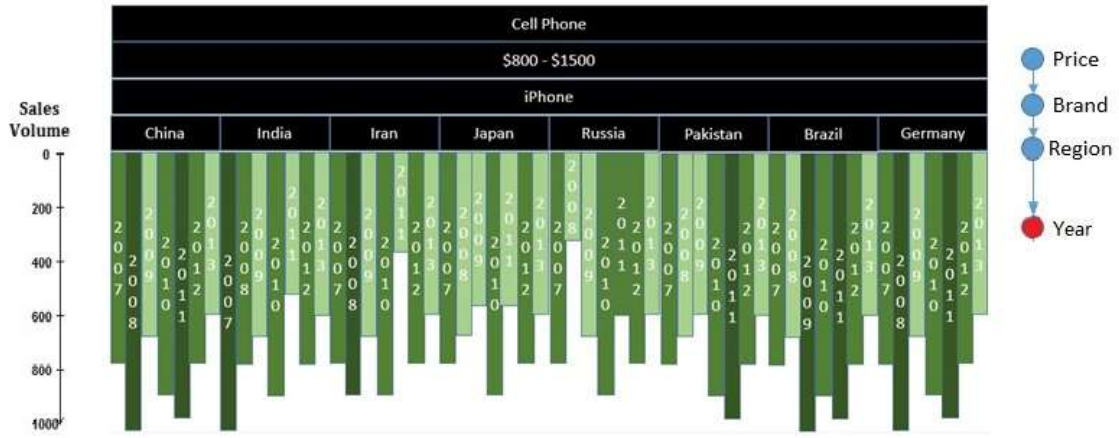


*Figure 39.* When the User Clicks “China” in (a), since “China” is Already in Focus in (a), it is Defocused and All its Siblings are Shown, and the Width of the Sub Items Change Accordingly.

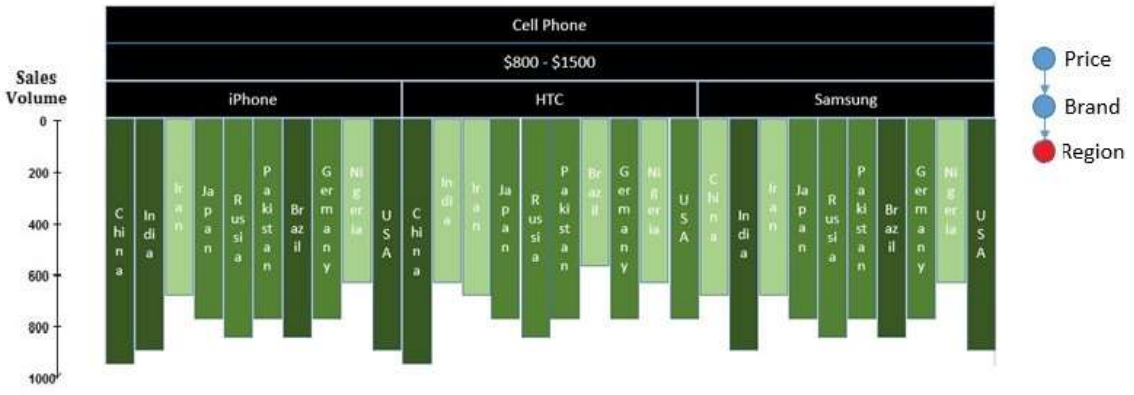


*Figure 40.* When the User Clicks "iPhone" in Figure 38, since "iPhone" is Already in Focus in Figure 38, it is Defocused and All its Siblings are Shown, and the Width of the Sub Items Change Accordingly. We don't Display the "years", Because the Width of the Item under the Active Layer is Below the Threshold.

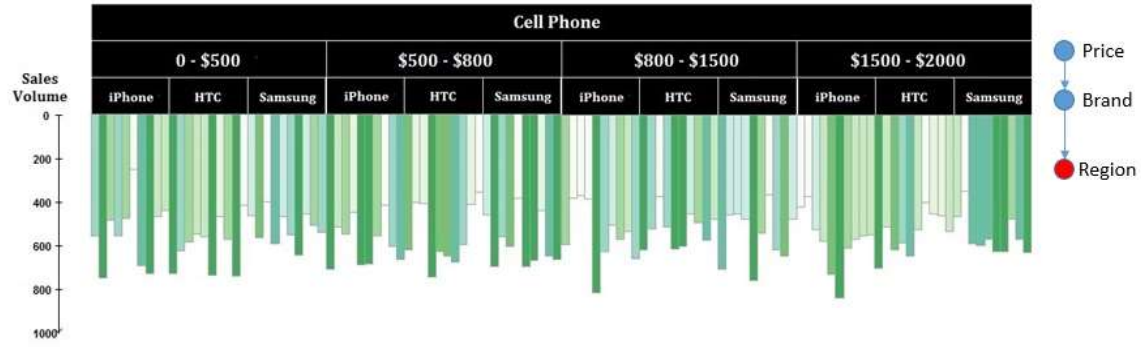


*Figure 41.* When the User Clicks "$800-$1500" in Figure 38, since ""$800-$1500" is Already in Focus in Figure 38, it is Defocused and All its Siblings are Shown, and the Width of the Sub Items Change Accordingly. We don't Display the "years", Because the Width of the Item under the Active Layer is Below the Threshold.

CHAPTER 4

CASE STUDY

## Large Hierarchy

The main advantage of our visualization is being able to efficiently display large hierarchies. Below is an example using the Cascading CurtainMap to display the population data all over the world, which is composed of five hierarchical levels. Figure 42 shows how the Cascading CurtainMap looks like at the beginning.

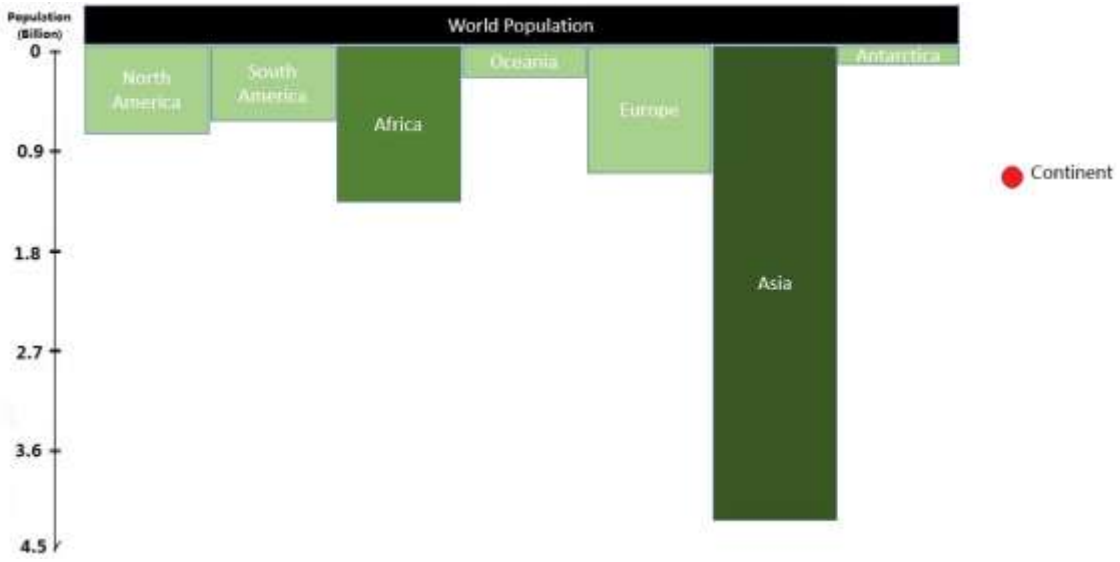


*Figure 42.* World Population Classified by Continents.

It only shows the user the population on the continent level and the user can interact with it to dig up details according to their interest. Let's say there are four types of users who are respectively interested in continent population, country population, state/province population, and city population. For the users interested in continent population, they don't need to do any interaction with the system since the continent level is the active layer in default. For the user interested in countries population, in order to see the population of the country to their interest, they can simply click the continent

where the country is, which is just one click away from original state. Figure 32 displays the result after the user clicks the North America in Figure 31.

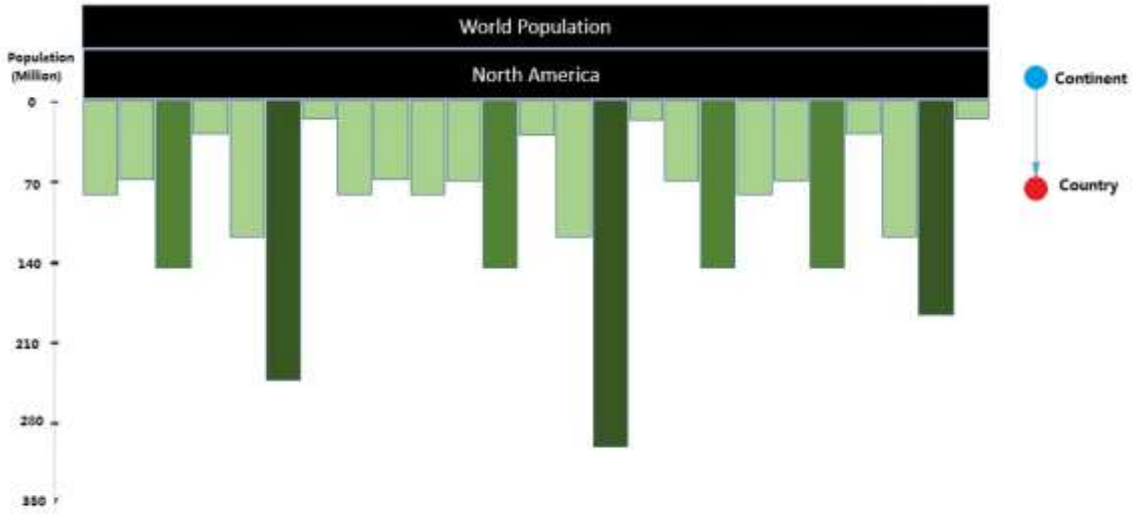


*Figure 43.* By Clicking the "North America" in Figure 42, the North America Population Classified by Countries can be Displayed. Users can Hover on the Items to See the Country Name and its Population.

For the user interested in state/province populations, they can first click the continent where the state/province is and the country where the state/province is (Figure 44), which is two clicks away from original state.

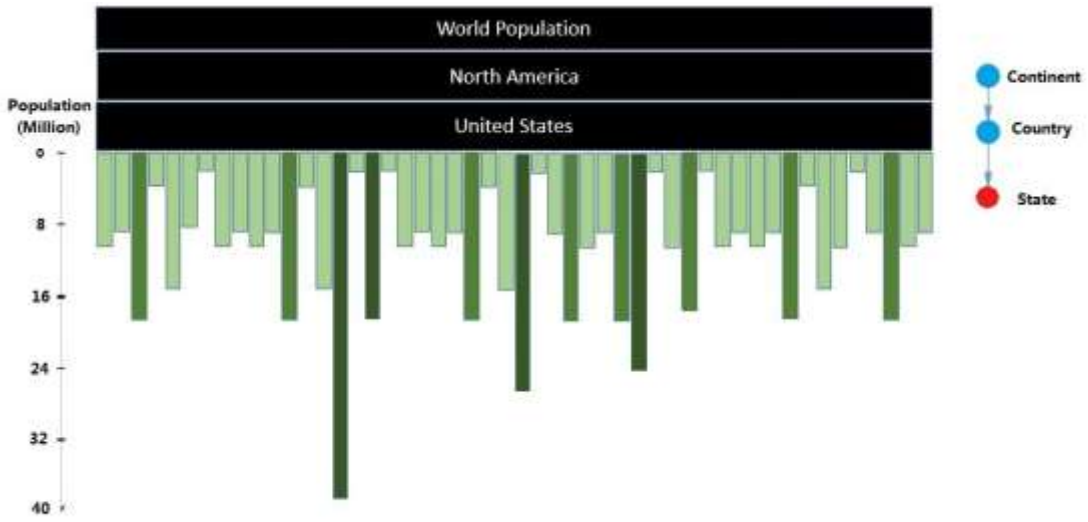


*Figure 44.* By Clicking the "North America" in Figure 32, the United States Population Classified by States can be Displayed. Users can Hover On the Items to See the State Name and its Population.

At last, for the users interested in cities populations, they first select the continent, then the country and lastly the state/province where the city is (Figure 45), which is three clicks away from original state.

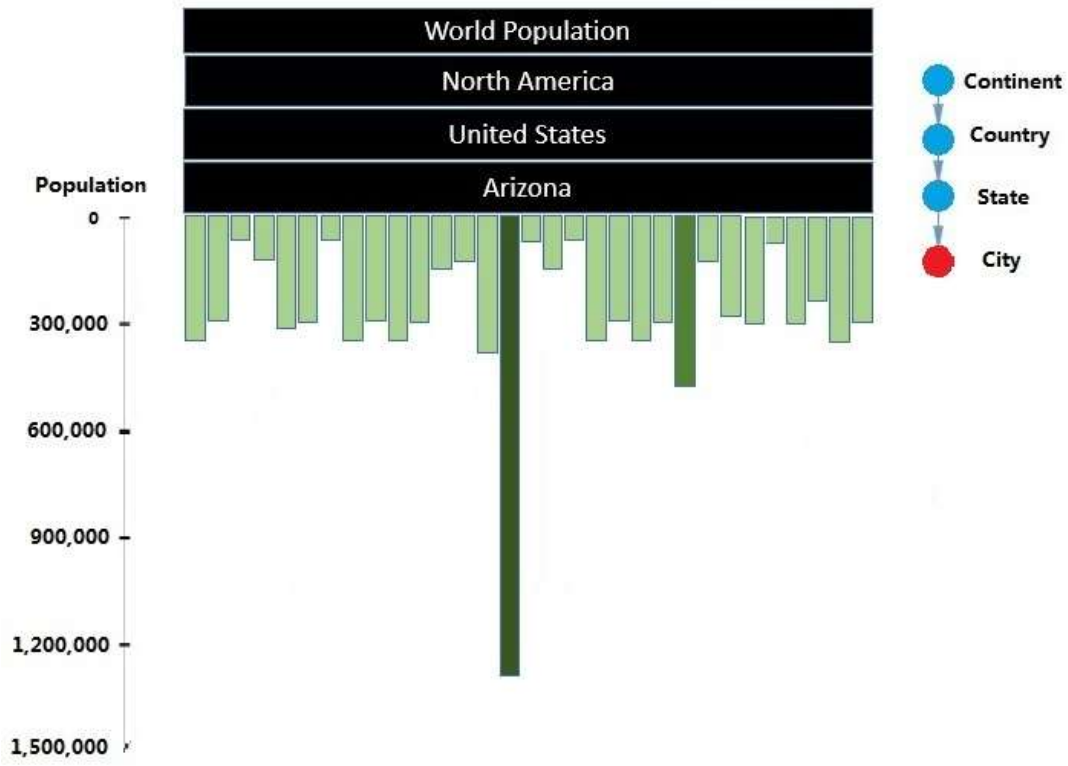


*Figure 45.* By Clicking the "North America" in Figure 33, the Arizona State Population Classified by Cities can be Displayed. Users can Hover On the Items to See the City Name and its Population.

Under any circumstance, in order to see the population on a higher level, the user only needs to click on that level and the level being clicked will become the active layer. For example, in Figure 45, if the user wants to get back to see the continent populations. They just need to click on "North America", which then will be defocused and all other continents will be shown. Therefore, this operation will directly guide the user to Figure 42 in a single step. Another way to see the items on a higher level is using right-click. Since each single right-click folds a hierarchical level, the time of right-clicks depend on the distance between the current active layer and the layer the user wants to investigate.

However, let's say, if the user wants to see the city populations in Sonora State in Mexico. They first need to click "United States" to display all the countries belonging to "North America" and find "Mexico", click it, then find "Sonora", click it again, which consists of three clicks. In this case, the time of the clicks depends on the distance between the items the user wants to see and the root of the first common ancestor of the items on the current active layer and the items the user wants to see.

Figure 46 explains such process. As we can learn in Figure 46, it takes only one step to go to the level right under the first common ancestor, which is North America, of cities in Arizona and cities in Sonora, and it further takes two steps to go to the cities in Sonora.

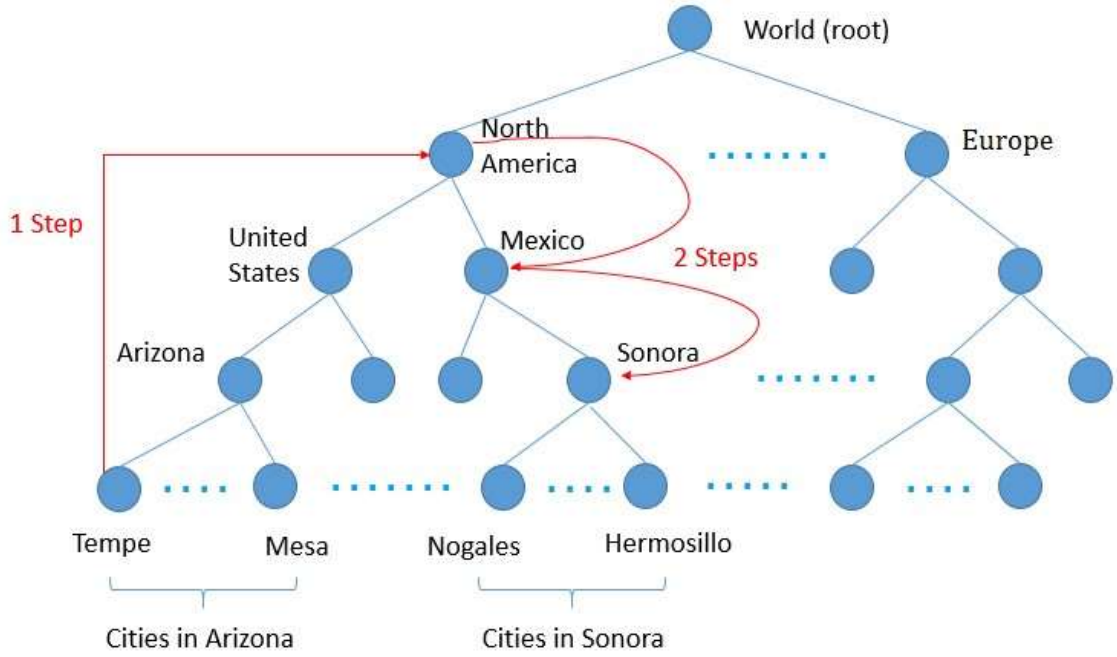


*Figure 46.* When "Cities in Arizona" is the Active Layer, it Takes One Single Click to go to the View in which "Countries in North America" is the Active Layer, and it Further Takes Two Steps to go to the View in which "Cities in Sonora" is the Active Layer. The Number of Steps is Equal to the Distance Between "North America", which is the First

Common Ancestor of “Cities in Arizona” and “Cities in Sonora”, and “Cities in Sonora”, which are the Items the User Wants to See.

For the “drill-down” process, instead of clicking “Mexico” and “Sonora”, the user can use left-click to unfold the hierarchy, which will cause the entire hierarchy to unfold one level or multiple levels depending on the number and width of the items to be shown. However, if the calculated width of the items on the level to be shown goes below the limit, that level won’t be shown. In such a case, the user still needs to focus only on the item they are interested and go through the process as shown in Figure 45.

**Flexible Hierarchy**

Another important feature of our Cascading CurtainMap is being able to display flexible hierarchies on the fly. As mentioned in the introduction, a flexible hierarchy uses attributes, which can also be called filter, to do the classifications in an order that could be changed according to users’ demand. A common example could be found when we are doing on-line shopping. In most online shopping websites, the items that match the keyword are classified by price range, product type, brand, relevance and so on. In such a case, the user can select the filters and narrow the scope. Using our Cascading CurtainMap, these filters can also be arranged in a specific order according to user’s interest, which enables the user to monitor some attributes they are concerned (e.g. sale volume, custom reviews and popularity) and find out the items that greatly match their interest. From the sellers and manufacturers’ perspective, there is also such a demand to change the order of filters applied on the products, because it enables them to discover some important patterns or potential problems.

In our Cascading CurtainMap, in order to specify the order of filters applied on the items, the user should follow the steps below:

1. Use the “Focus/Defocus” technique discussed before to initiate the data resolution of the visualization;
2. Specify the filter used on the current active layer using mouse wheel (see Figure 31-33);
3. Unfold the hierarchy through left-click and generate a new active layer;
4. Go to step 2 until it reaches the bottom of the hierarchy.

Whenever the hierarchy cannot be unfolded because the width of the items on the next level goes below the limit, the user needs to select (click) the item of their interest and keep drilling down the hierarchy. Such process is depicted in Figure 47.

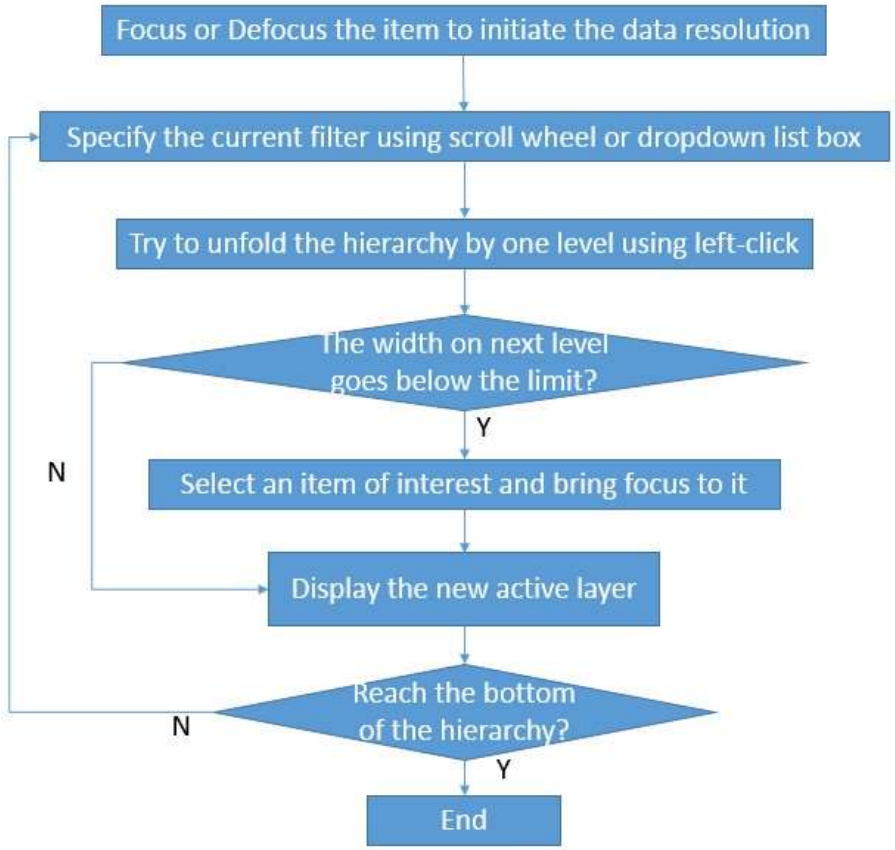


*Figure 47.* The Flow Chart of Specifying the Order of Filters Applied on the Hierarchy using Cascading CurtainMap.

The advantage of using scroll wheel to specify the order of filters is that it enables the user to go through the other filters that might catch their attention before they find the filter wanted, but the operation time in this way is n!, where n is the number of all the filters. Therefore it is better to use scroll wheel to specify the order when n is small. An alternative way of specifying the order of filters is to use the dropdown list on the right-hand side. Since it takes constant time to find the needed filter for each level, it only takes O(n) time to specify the order of filters. At any point, the user can jump to any hierarchical level by using "defocus" technique and specify another order of filters applied on the data.

## Big Data

As we mentioned in the introduction, big data is usually hierarchically structured and often contains a large number of hierarchical levels. As the time passes by, the data size grows further. Below is an example of how the supply chain data grows into big data.

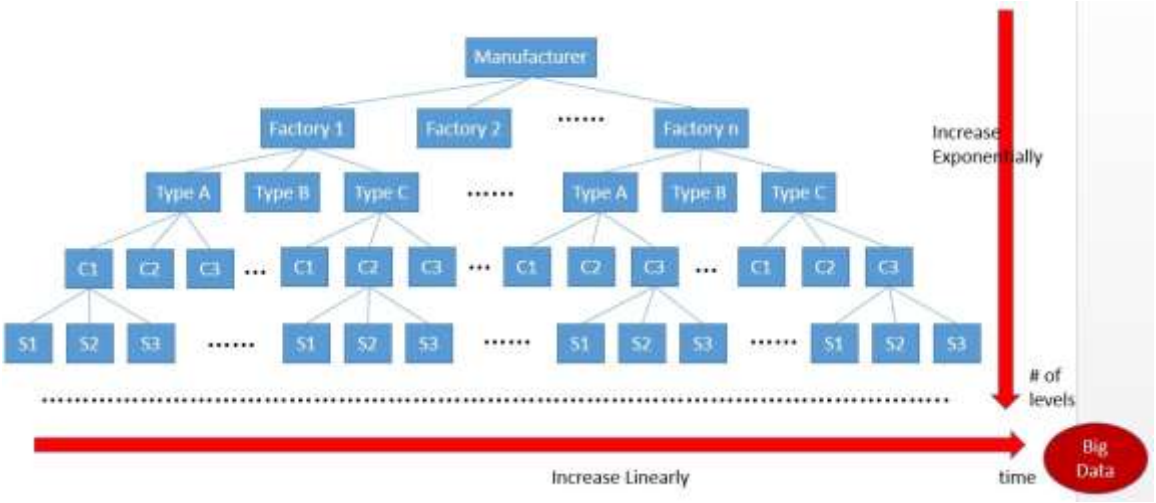


*Figure 48.* The Size of the Supply Chain Data Grows as the Number of Hierarchy Levels Increases or as the Time Passes By.

In the graph above, the supply chain of a manufacturing company has multiple factories. Each factory is responsible for processing a set of products, which can be further classified by filters such as product type, product series, and product class. The data size grows exponentially as the number of filters increases. Additionally, people are often interested in attributes such as product inventory, product sales volume and so on. At different point of time, these attributes usually have different values and the data size grows linearly as the time passes by. In this case, large hierarchy and data changing over time are the main characteristics that contribute to big data. We already talked about how

our system visualizes large hierarchies. It enables the user to see the data at a specific point of time. By connecting to a real time database and using interaction, our system is capable of displaying data which changes over time. Currently, our system is used to display the supply chain data and the data we are using contains about 600,000 rows of records. Figure 49 shows how we process and visualize the supply chain data.

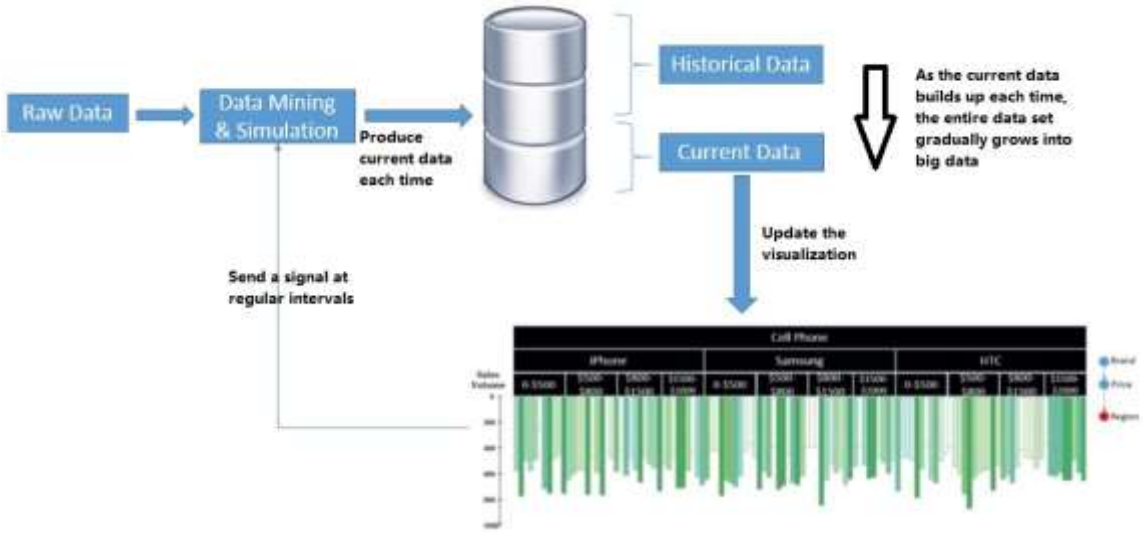


*Figure 49.* The Processing and Visualization of the Supply Chain Data.

First of all, the raw data of the supply chain is processed using data mining and simulation, which generates the current data in the database at each time. Then, the current data is used to update the visualization. At regular intervals, our visualization sends a signal to the "Data Mining & Simulation" part, which again generates the current data to be used to update the visualization. In this way we visualize the hierarchical data that changes over time.

# CHAPTER 5

# CONCLUSION AND FUTURE WORK

We have introduced our approach of visualizing large and flexible hierarchies. In visualizing large hierarchies, focused and contextual views are the issues that are discussed the most. Traditional approaches addressing focused and contextual views include Overview+Detail, Focus+Context, Zooming, and Cue-based techniques (Cockburn, Karlson, & Bederson, 2008), which put focused view and contextual view in a single window or separated windows to meet different kinds of requirements. In our approach, we simply hide the contextual view, because we felt that in this way it can leave more space for the focused view. The drawback of this approach is that the user loses the context and they cannot make comparison between the items in focused view and the items in contextual view, which is also the challenge most diagrams using techniques such as focus+context, focuse+context and zooming techniques face, because the items in both focused view and contextual view are distorted, it is also difficult to make comparison between them.

Our approach is similar to zooming in/out. The difference is that in our system, the zooming in/out is performed on the items selected by the user, whereas the general zooming in/out is to focus or defocus specific area based on the location of user's cursor. The process of drilling down the hierarchy using our approach is similar to the exploration of a small location in a large map. By bringing the focus to the area of interest, more details of that area will be shown, which enables the user to dig up the details further, but the price it has to pay is the temporal loss of the context.

In dealing with flexible hierarchies, the change of the hierarchy is based on the specification of what to be represented by the active layer on each hierarchical level. Instead of using width to represent value, the active layer uses length to do so, which makes it easier for the user to make comparison between different items. Additionally, we also introduced sequential and qualitative color schemes for the items on the active layer, which help the user to develop a better understanding of different data values. The node-link structure on the left-hand side of the diagram tells the user the current logical structure of the hierarchy and enables them to jump to any hierarchical level just by a single click.

Based on some drawbacks of our approach, there are a number of future improvements that could be made:

1. Instead of hiding the contextual view, it would be interesting to use some proper distortion on it and to see the its result;
2. In dealing with flexible hierarchies, the hierarchy can be specified by changing the node-link structure, which can greatly facilitate the process;
3. Our system can be combined with other diagrams (e.g. histogram, scatterplot and heat map) to improvement the efficiency of understanding and analyzing the data;
4. Providing more options for selecting the color scheme might facilitate the understanding process;
5. When the hierarchy goes extremely large, we can provide an overview tab for the user to locate where they are in the hierarchy;
6. It might be challenging to focus multiple items in the same time without spoiling the other functionalities of our system but it can facilitate comparisons.

Our objective in this work is to introduce the techniques to visualize large and flexible hierarchies. The prototype of our system is used to display hierarchical data with four levels and it is proved to be easily understandable and more efficient than other diagrams. However, it would be valuable to evaluate our system by applying it on different large hierarchical data sets and getting the users' feedback. Currently our system is based on a local server, we intend to implement it on the web, make it cross-platform and capable of reading data from remote machines.

# REFERENCES


Ahlberg, C., Williamson, C., & Shneiderman, B. (1992, June). Dynamic queries for information exploration: An implementation and evaluation. In Proceedings of the SIGCHI conference on Human factors in computing systems (pp. 619-626). ACM.

Andrews, K., & Heidegger, H. (1998, October). Information slices: Visualising and exploring large hierarchies using cascading, semi-circular discs. In Proc of IEEE Infovis' 98 late breaking Hot Topics (pp. 9-11).

Barlow, T., & Neville, P. (2001, October). A comparison of 2-D visualizations of hierarchies. In Information Visualization, IEEE Symposium on (pp. 131-131). IEEE Computer Society.

Brath, R. (1997, June). 3D Interactive Information Visualization: Guidelines from experience and analysis of applications. In HCI (2) (pp. 865-868).

Burch, M., Raschke, M., & Weiskopf, D. (2010). Indented pixel tree plots. In Advances in Visual Computing (pp. 338-349). Springer Berlin Heidelberg.

Chintalapani, G., Plaisant, C., & Shneiderman, B. (2004, July). Extending the utility of treemaps with flexible hierarchy. In Information Visualisation, 2004. IV 2004. Proceedings. Eighth International Conference on (pp. 335-344). IEEE.

Cleveland, W. S. (1985). The elements of graphing data (pp. 135-143). Monterey, CA: Wadsworth Advanced Books and Software.

Cockburn, A., Karlson, A., & Bederson, B. B. (2008). A review of overview+ detail, zooming, and focus+ context interfaces. ACM Computing Surveys (CSUR), 41(1), 2.

Cockburn, A., & McKenzie, B. (2001, March). 3D or not 3D?: evaluating the effect of the third dimension in a document management system. In Proceedings of the SIGCHI conference on Human factors in computing systems (pp. 434-441). ACM.

Collins, C. (2007). Docuburst: Radial space-filling visualization of document content. Knowledge Media Design Institute, University of Toronto, Technical Report KMDI-TR-2007-1.

Heer, J., Bostock, M., & Ogievetsky, V. (2010). A tour through the visualization zoo. Commun. ACM, 53(6), 59-67.

Heer, J., & Robertson, G. G. (2007). Animated transitions in statistical data graphics. Visualization and Computer Graphics, IEEE Transactions on, 13(6), 1240-1247.

Hurvich, L. M., & Jameson, D. (1957). An opponent-process theory of color vision. Psychological review, 64(6p1), 384.

Johnson, B., & Shneiderman, B. (1991, October). Tree-maps: A space-filling approach to the visualization of hierarchical information structures. In Visualization, 1991. Visualization'91, Proceedings., IEEE Conference on (pp. 284-291). IEEE.

Kolatch, E., & Weinstein, B. (2001). Cattrees: Dynamic visualization of categorical data using treemaps. Project report.

Kruskal, J. B., & Landwehr, J. M. (1983). Icicle plots: Better displays for hierarchical clustering. The American Statistician, 37(2), 162-168.

Lamping, J., & Rao, R. (1996). The hyperbolic browser: A focus+ context technique for visualizing large hierarchies. Journal of Visual Languages & Computing, 7(1), 33-55.

Lamping, J., Rao, R., & Pirolli, P. (1995, May). A focus+ context technique based on hyperbolic geometry for visualizing large hierarchies. In Proceedings of the SIGCHI conference on Human factors in computing systems (pp. 401-408). ACM Press/Addison-Wesley Publishing Co..

Plaisant, C., Grosjean, J., & Bederson, B. B. (2002). Spacetree: Supporting exploration in large node link tree, design evolution and empirical evaluation. In Information Visualization, 2002. INFOVIS 2002. IEEE Symposium on (pp. 57-64). IEEE.

Reingold, E. M., & Tilford, J. S. (1981). Tidier drawings of trees. Software Engineering, IEEE Transactions on, (2), 223-228.

Rheingans, P., & Landreth, C. (1995). Perceptual principles for effective visualizations. In Perceptual Issues in Visualization (pp. 59-73). Springer Berlin Heidelberg.

Robertson, G. G., Mackinlay, J. D., & Card, S. K. (1991, March). Cone trees: animated 3D visualizations of hierarchical information. In Proceedings of the SIGCHI conference on Human factors in computing systems (pp. 189-194). ACM.

Senay, H., & Saltz, J. S. (1997, June). Dynamic visualization of hierarchical data. In Electronic Imaging'97 (pp. 451-458). International Society for Optics and Photonics.

Stasko, J., Catrambone, R., Guzdial, M., & McDonald, K. (2000). An evaluation of space-filling information visualizations for depicting hierarchical structures. International Journal of Human-Computer Studies, 53(5), 663-694.

Stasko, J., & Zhang, E. (2000). Focus+ context display and navigation techniques for enhancing radial, space-filling hierarchy visualizations. In Information Visualization, 2000. InfoVis 2000. IEEE Symposium on (pp. 57-65). IEEE.

Stevens, S. S. (1946). On the theory of scales of measurement.

Tversky, B., Morrison, J. B., & Betrancourt, M. (2002). Animation: can it facilitate?. International journal of human-computer studies, 57(4), 247-262.

Van Wijk, J. J., & Van de Wetering, H. (1999). Cushion treemaps: Visualization of hierarchical information. In Information Visualization, 1999.(Info Vis' 99) Proceedings. 1999 IEEE Symposium on (pp. 73-78). IEEE.

Wang Baldonado, M. Q., Woodruff, A., & Kuchinsky, A. (2000, May). Guidelines for using multiple views in information visualization. In Proceedings of the working conference on Advanced visual interfaces (pp. 110-119). ACM.

Wetherell, C., & Shannon, A. (1979). Tidy drawings of trees. Software Engineering, IEEE Transactions on, (5), 514-520.

Yang, J., Ward, M. O., Rundensteiner, E. A., & Patro, A. (2003). InterRing: a visual interface for navigating and manipulating hierarchies. Information Visualization, 2(1), 16-30.