Interactive Visual Analysis of the NSF Funding Information

Shixia Liu* IBM China Research Lab
Nan Cao† IBM China Research Lab
Hao Lv‡ Shanghai Jiaotong University
IBM China Research Lab

ABSTRACT

This paper presents an interactive visualization toolkit for navigating and analyzing the National Science Foundation (NSF) funding information. Our design builds upon the treemap layout and the stacked graph to contribute customized techniques for visually navigating and interacting with the hierarchical data of NSF programs and proposals, supporting visual search and analysis, and allowing the user to make informed decision.

In this visualization toolkit, we propose two visualization techniques to simplify the navigation of the hierarchical data: 2.5 Dimensional treemaps to make the hierarchical structure more easily to be recognized, and labeled treemap to help the user to get a clear overview of the content of the structure and make the internal area of rectangles correspond to the weights of the data set. Furthermore, an incremental layout method is adopted to handle information on a large scale. The improved treemap visualization will help to visually analyze the static funding data and the stacked graph is utilized to analyze the time-series data. Through these visual analysis techniques, research trends of NSF, popular NSF programs are quickly identified. The primary contribution is a demonstration of novel ways to effectively present and analyze NSF funding data.

Keywords: Interactive visualization, Treemaps, Time series data visualization, Overview+detail, Cascaded rectangles.

1 INTRODUCTION

The NSF funds research and education in science and engineering. Each year, NSF receives approximately 30,000 new or renewal support proposals for research, graduate and postdoctoral fellowships, and math/science/engineering education projects; it makes approximately 9,000 new awards. As the amount of information on the NSF website increases continually, decision-makers are often overwhelmed, something must be done to allow decision-makers to easily extract the information it contains. Information visualization is a powerful tool to help find the relations and trends of the data. Applying visualization technology to the website can aid for allowing the decision-maker to find the important information more easily.

One interesting approach to presenting large volumes of data in a user-friendly manner is treemap visualization. The treemap visualization method, developed by Johnson and Shneiderman [1], maps hierarchical information to a set of nested rectangles whose areas correspond to some quantity (weight) in the data set. Treemaps are efficient and compact displays, they are very effective in visualizing large hierarchical collections of quantitative data in a single view and showing attributes of leaf nodes using size and color coding. More work has been done on treemap visualization [2]. Treemaps are originally designed for visualizing a filled hard disk, now they have been successfully applied to a wide range of applications including financial analysis [3, 4], sports data analysis [5, 6], decision making [7], software analysis [8, 9], etc.

*e-mail:liusx@cn.ibm.com
†e-mail:nancao@cn.ibm.com
‡e-mail:hlv@acm.org

IEEE Pacific Visualisation Symposium 2008
4 - 7 March, Kyoto, Japan
978-1-4244-1966-1/08/$25.00 ©2008 IEEE

-NSF organization
-Directorates (Assistant Director)
-Divisions (Division Director)
-Programs (Program Officer/Director)

Figure 1: NSF organization structure

Another interesting approach is the stacked graph which is adopted by Martin Wattenberg in his NameVoyager [10]. This method is very effective for visual exploration of thousands of time series.

In this paper, we present the design of the visual funding navigator, a visualization system for exploring and analyzing online NSF funding related data. This work extends our previous work [11] by designing more visualization methods and visual analysis methods for the visual funding navigator. This funding navigator builds on the treemap layout and the stacked graph to help the user visually observe and analyze the NSF funding data. The major feature of this navigator is that it provides an integrated support for static analysis and dynamic analysis. Treemap layout is adopted to analyze the static funding data. This analysis will help the user quickly discover the influential research organizations and projects in a particular topic area. Staked graph is utilized to dynamically analyze the time-series data embedded in the funding information, i.e., the distribution of grant amount of each NSF program over time. Such kind of dynamic analysis will help the user track the NSF program trends, thus help the user easily find the focused topic of interest.

The rest of the paper is organized as follows. Section 2 briefly introduces the corpus data. The visualization design and implementation techniques are presented in Section 3. Section 4 illustrates the visual analysis method and corresponding results. Conclusion and future work are presented in Section 5.

2 ANALYSIS OF CORPUS DATA

The visual funding navigator is based on a data set, derived from the NSF official website that tracks the NSF call for proposals and proposals in the United States. The NSF programs and the corresponding request for proposals, and NSF awards are organized by the NSF organization structure (see Figure 1), such as directorate, division, program.

In order to use the treemap layout to visualize the NSF programs and awards, a tree structure for organizing them is necessary. A natural way to organize them is by the NSF organization structure. Another way to organize them is by the locations where the investigators are located, such as state, research organization. In our toolkit, we will utilize both of these two tree structures to generate the treemap views to help the user understand the funding related data more easily and quickly.

Each NSF award has a start date, an estimated expired date, and awarded amount. From such kind of information, we could extract the grant amount with time in each NSF program. These time series are the input of stacked graphs. We will illustrate the construction of stacked graphs in Section 4.2 in detail.

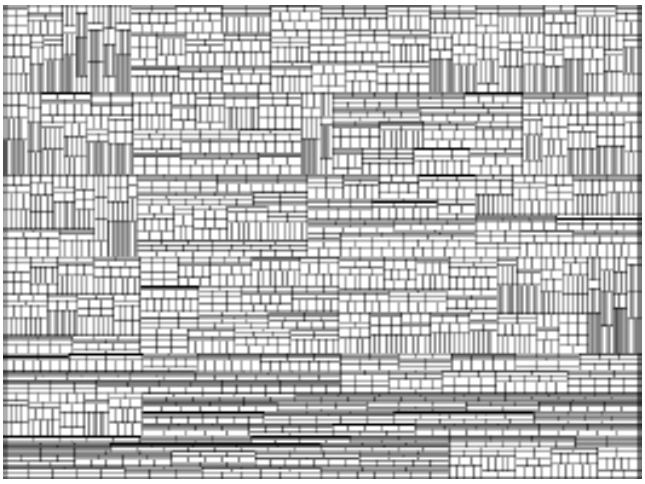

Figure 2: Treemap of a balanced tree.

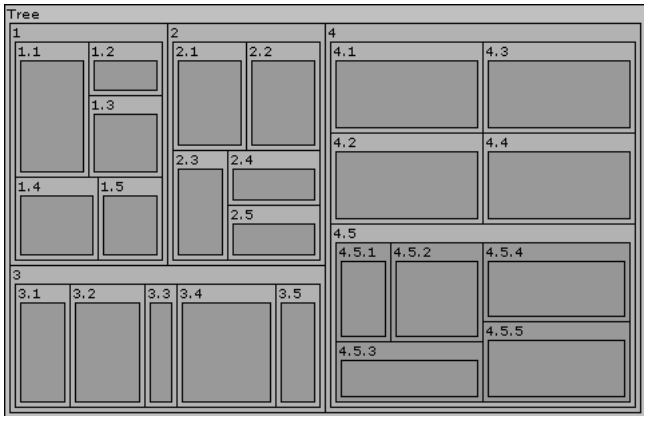

Figure 3: Labeled Treemaps.

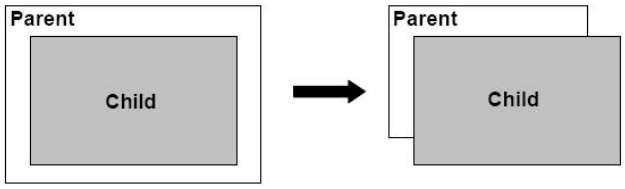

Figure 4: Representation of child-parent relations.

3 Visualization Design and Implementation

Our visualization design builds mainly upon the treemap layout and the stacked graph. Based on these visualization techniques, we contribute customized techniques for visually navigating and interacting with NSF Funding data, supporting visual analysis, and allowing the user to make informed decision.

3.1 Treemap Layout Improvements

Treemap enables users to compare nodes and sub-trees even at varying depth in the tree, and help them spot patterns and exceptions. However, treemaps have limitations [12]. One problem is that treemaps often fall short to visualize the structure of the tree. Thus it's not easy to discern the entire structure of the hierarchical data in the traditional treemaps, especially when the number of nodes comes up to thousands and the hierarchy structure is very deep (usually, when the maximum depth of the hierarchy is greater than 3). In such cases, treemap could only come to help to view the distribution of leaves with hierarchical structure sacrificed. The worst case is a balanced tree, where each parent has the same number of children and each leaf has the same size. The treemap degenerates here into a regular grid. Wijk et al. [13] showed such an example in their work about cushion treemaps (see Figure 2). This example is about an (artificial) organization chart which is modeled after the structure of their university. Six levels in the hierarchy are shown, the final 3060 rectangles denote individual employees. In this example, question such as "What is the largest section?", "Which division does a specific employee belong to?" are hard to answer by just from such a treemap view. Figure 3 is another example. In this figure, the root node of the tree is named "Tree". And it has four child nodes named "1", "2", "3" and "4", each of which has its own child nodes. The child nodes are named following the index convention of book sections. The names of the leaves are ignored in the figure. It could be seen that the route <4.5.5, 4.5, 4, Tree> is far less than obvious.

Since the structure of a tree is very useful for gaining an overview of the hierarchical data, it is therefore important to design a new representation of the treemap. Some initial efforts have been tried to get a clear perception of the structure. Nested treemaps [1] are a partial remedy. During the subdivision process instead of the initial rectangle a slightly smaller rectangle is used, such that each group of siblings is enclosed by a margin. However, this consumes screen space and needs the visual interpretation, especially for deeply nested trees, still requires effort from the viewer (see Figure 3). Cushion treemaps [13] employ the shading effects to improve the perception of tree structure by feigning specular reflection and thereby simulating a curved surface. This method is also adapted for the enhancement of nested treemaps to framed treemaps [14], resulting in quasi-three-dimensional borders. Steptree [15] is a three dimensional extension of Shneiderman's spacing filling treemap concept [16], it shows depth in tree as the heights of steps.

All these methods may reduce, but do not prevent, misinterpretations concerning the hierarchy. It is still not easy for the viewers to understand the whole picture when the hierarchy is complicated.

In this section, we describe a new method for visualizing the hierarchical funding related data based on treemaps. The major improvement of this method is that it uses node cascading technique and labels to show the tree structure.

3.1.1 2.5D Treemap

In this section, we propose a new approach, 2.5D treemaps, to tree visualization. The 2.5D treemaps aim at highlighting the child-parent relations of the hierarchy without losing the ability of visualizing large data set within constrained space. The intuition is to make the depth of the tree an additional dimension. In this method, we take advantage of a "2.5-Dimensional" representation in which a "height" coordinate is represented by an offset factor. Figure 4 is an illustration of this basic idea.

Figure 5 is a 2.5D treemap for the same hierarchical data of Figure 3. It can be seen that the hierarchy becomes apparent. The viewer can easily tell where a node lies in the hierarchy.

Given a nested treemap, the conversion is done recursively as follows.

1. The leaf rectangles are unchanged.

2. For a non-leaf rectangle with all its child rectangles converted, move its bottom right corner up and left to make it as small as possible while still covering all the center points of its child rectangles. See Figure 6 for an illustration.

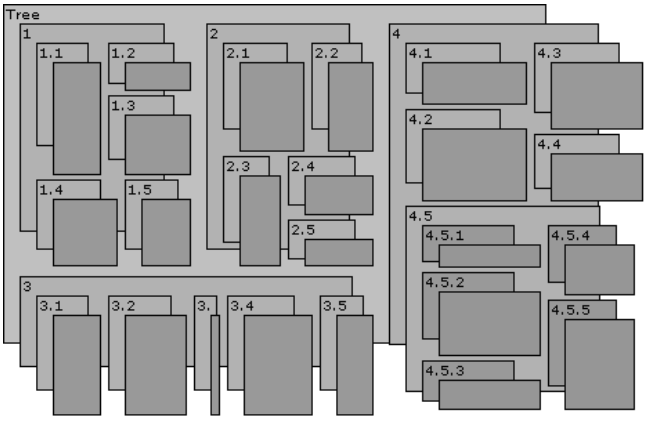

Figure 5: 2.5D labeled treemaps.

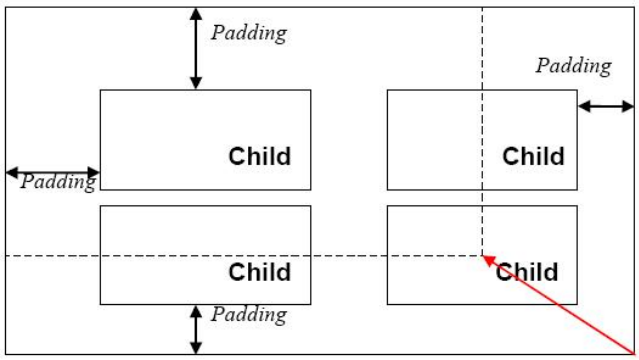

Figure 6: Convert nested treemaps to 2.5D treemaps.

In the conversion, since the top left corners of rectangles are fixed, the cascading offset is determined by the top and left padding spaces. In 2.5D treemaps, each rectangle must intersect with its parent (except the root) and may only intersect with its ancestors and descendants.

3.1.2 Improved Labeled Treemap layout

A Labeled treemap is another extension of the nested treemap. It extends the nested treemap in that rectangles corresponding to non-leaf nodes contain additional space for their names and borders. Labels on the non-leaf nodes make it efficient for viewers to identify the categories and find data that they are interested in. However, the labels and borders occupy space which is originally owned by the weight of the data set, original layout methods fail in making the areas of rectangles correspond to the weights of the data set. Images produced by these methods are difficult for recognition. Furthermore, these methods may make some nodes correspond to meaningless rectangles whose heights or widths are less than zero, which makes these nodes disappeared from the screen. Thus, a new method is needed to improve such case.

In this section, we propose a new method for constructing a treemap with labels. The goal of this method is to make the areas of leaves well approximate their weights and avoid the case of meaningless rectangles.

We first introduce some preliminary definitions which are useful for subsequent discussions.

Definition 3.1 *The density of a leaf rectangle is the amount of weight on one unit of area.*

Intuitively, we can impose requirement on the least amount of area for one unit of weight to ensure every item is visible. We present this intuition with the help of the definition below.

Definition 3.2 *A nested treemap is fit on density ρ if every unit of the weights of the leaf rectangles occupies area at least $\frac{1}{\rho}$.*

The following two facts about the fitness on density are obvious. The second fact is a corollary of the first fact.

Fact 3.1 *Given a hierarchy, if there is no nested treemap that is fit on ρ, then for every $\rho' < \rho$, there is no nested treemap that is fit on ρ'.*

Fact 3.2 *Given a hierarchy, if there is a nested treemap that is fit on ρ, then for every $\rho' > \rho$, there is a nested treemap that is fit on ρ'.*

In our algorithm, we generate the "good" nested treemaps. Among these generated treemaps, we calculate the approximate value of $\overline{\rho}$, and then try to find one of these treemaps which is fit on $\overline{\rho}$. We organize our algorithm into two parts. The first or outer part is to use binary search to find the most fit density $\overline{\rho}$. The second or inner part is to answer whether a certain $\overline{\rho}$ is fit on a given rectangle $\mathscr{R}$.

The first part is trivial and easy to implement. We focus on the second part of our algorithm. The process of the second part is done recursively. Given ρ and a set of nodes and a rectangle $\mathscr{R}$, the algorithm to answer whether ρ can be fit is described $Calc(A,\mathscr{R})$ as follows. We use $\mathscr{R}^\circ$ to represent the interior of the $\mathscr{R}$.

1. If A contains only one non-leaf item u, then map u to $\mathscr{R}$ and return $Calc(B,\mathscr{R}^\circ)$, where B is the set of sub items of u;
2. If A contains only one leaf item u then calculate average weight per unit of area represents. If it is smaller than ρ, then map u to $\mathscr{R}$ and return **true**, otherwise return **false**;
3. Sort and split A into two parts A_1 and A_2 with approximately equal weights;
4. Try all possible splits on the longer edge of $\mathscr{R}$ to split it into two sub rectangles $\mathscr{R}_1$ and $\mathscr{R}_2$.
   (a) Call $Calc(A_1,\mathscr{R}_1)$ and $Calc(A_2,\mathscr{R}_2)$ recursively;
   (b) If both returns **true**, then return **true**;
5. return **false**;

Steps 3 and 4 show our definition of "good", since they could generate all "good" nested treemaps. They split the interior of the rectangle on its longer edge into two sub rectangles, and then generate nested treemaps recursively on two subsets of siblings with approximately equal weights. It is much like the layout process of a squarified treemap [14].

Next, we evaluate the improvements of the new algorithm. The algorithm used in squarified treemaps [14] is selected to be compared with our algorithm. Both of the two algorithms will be applied to the same collections of data and with the same padding setting. The comparison is made on several factors of the resulting layouts. The factors are described as follows. The "total" factor represents the total number of leaf items. The "*missing*" factor represents the number of the leaf items that are being missed. The "$diff$" factor of two rectangles represents the difference between the density of them. It is calculated by the formula below, where w_i is the weight of the i-th leaf rectangle and a_i is the area of the i-th leaf rectangle.

$$diff_{(i,j)} = \left| \frac{a_i \times w_j - a_j \times w_i}{a_i \times w_j + a_j \times w_i} \right|$$

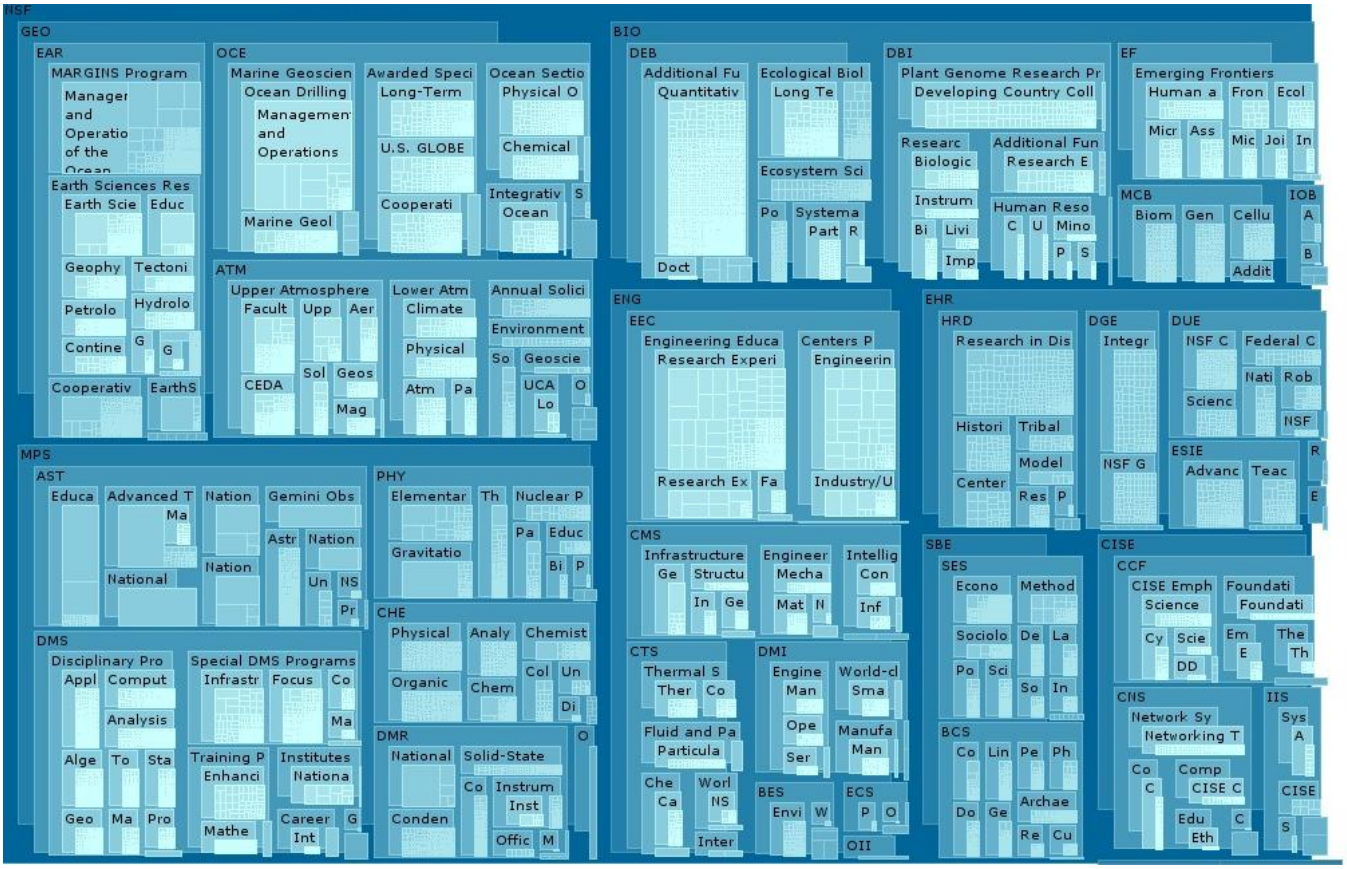

Figure 7: Visualization of the currently active awards on NSF website (31159 awards).

The "*ratio*" factor of a rectangle represents the aspect ratio of a rectangle. It is calculated by dividing the length of the longer edge by the length of the other. The value of this factor is at least 1. The "*dens*" factor of a rectangle represents the density, which is the amount of weight on one unit of area. The "$\overline{dens}$" factor of a treemap is the average density over all the leaf rectangles. The two factors are calculated by the formula below, where w_i is the weight of the i-th leaf rectangle and a_i is the area of the i-th leaf rectangle.

$$dens_i = \frac{w_i}{a_i},\ \overline{dens} = \frac{\sum w_i}{\sum a_i}$$

In the following tables, "E" is used to calculate the expectation and "D" is used to calculate the deviation. "DISPLAY" represents the size of display rectangle. "METHOD" represents which layout algorithm is used: "ours" is our algorithm and "squari" is the squarified treemap algorithm.

Table 1 is the layout result of some of the awards which are funded since Jan 1, 2005. They are generated with different pixel displays. Table 2 is the layout result of different ranges of data with a 1400×1050 pixel display. They are the awards funded since Jan 1, 2005 with 9373 leaves, the awards funded since Jan 1, 2004 with 16874 leaves, and all the active awards in NSF website with 31159 leaves.

From the results, we can see several facts. First, the results of the squarified layout algorithm have many missing items when the display area is not big enough, while the new algorithm does not miss any item. Second, by using the new algorithm, the average difference of densities between two leaf rectangles are much smaller and the densities of the leaf rectangles are much more uniformly distributed, which implicates that the sizes of the leaf rectangles well correspond to their weights. Third, the aspect ratio generated by the new algorithm is even better and more stable. We conclude that the new algorithm improves the case a lot and is really necessary.

total	3602			
DISPLAY	1024×768		1280×1024	
METHOD	ours	squari	ours	squari
missing	0	320	0	63
$E(diff)$	0.0462	0.182	0.0238	0.110
$E(ratio)$	1.70	2.41	1.75	2.33
$D(ratio)$	0.00964	0.885	0.0659	0.816
$E(dens)$	2.91	19.3	0.375	3.62
$D(dens)$	1.72	256	0.237	103
$\frac{E(dens)}{\overline{dens}}$	1.25	13.9	1.13	12.1

Table 1: Some of the awards funded since Jan 1, 2005

3.1.3 Visualization of Huge Data Set

Though the layout results are satisfactory, the improved layout algorithm described above is not efficient enough to be applied to huge data sets. Furthermore, if there are too many leaf nodes on the screen, then the area assigned to each leaf node is small. Especially for the leaf nodes with small weights, the allocated areas are much smaller than other nodes, therefore it may be a bit hard to select (or even see) these leaf nodes (see Figure 7).

To solve this problem, we designed an incremental layout method, in which a simple level-of-detail calculation is adopted to handle information on a large scale. The basic idea of this method

DISPLAY	1400×1050					
total	9373		16874		31159	
METHOD	ours	squari	ours	squari	ours	squari
missing	0	332	0	443	0	989
$E(diff)$	0.0251	0.121	0.0215	0.124	0.0179	0.119
$E(ratio)$	1.68	2.39	1.62	2.02	1.64	2.12
$D(ratio)$	1.17	15.8	0.010	0.364	0.264	6.12
$E(dens)$	1.16	1.54	2.10	2.96	4.19	5.17
$D(dens)$	0.724	7.63	1.14	24.5	2.22	21.4
$\frac{E(dens)}{dens}$	1.19	1.95	1.15	1.98	1.18	1.82

Table 2: Different ranges of data with a 1400×1050 display.

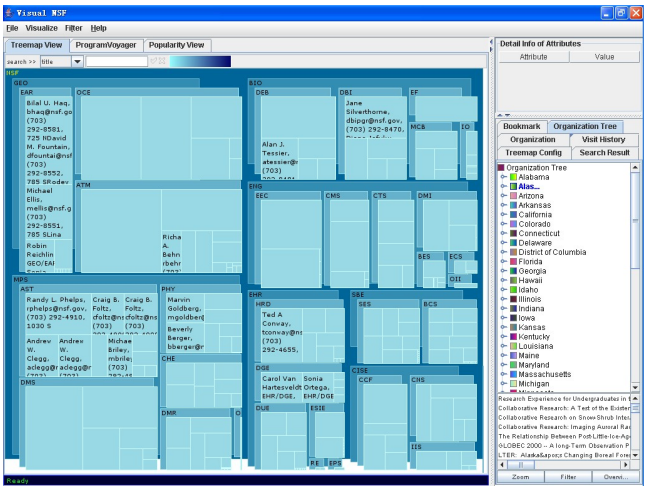

Figure 8: User interface of the visual funding navigator.

is that it will filter out low level nodes when the areas assigned to the low level nodes are too small and treat their direct parent nodes as leaf nodes. Then the treemap layout method described in Section 3.1 is utilized to map the filtered tree to the treemap first. Here, the weight of the parent node is the aggregation of the weights of its children. When the user clicks on the node in this map, it will then smoothly zoom in to the selected node area, and the pseudo leaf node area are split again according to the weights of its children. This process will be repeated if the current leaf nodes still have children. Figure 8 is the layout result of the same tree (the level of this tree is 5) in Figure 7 with fourth-level and fifth-level nodes filtered. Compared with Figure 7, the nodes in Figure 8 are more easily to be recognized and the initial layout requires less time.

Furthermore, even adopting the above method, there might still exist relatively small areas which are a little bit hard to be recognized and selected after the zooming operation, therefore another level-of-detail calculation is performed so that the leaf nodes whose areas are greater than the given threshold are rendered to the screen and the sibling leaf nodes whose areas are less than this threshold are combined together as a group leaf node. When the area of the group node is enlarged by the zooming operation, the enlarged area of this group node is then split again according to the weights of its sub-nodes.

3.2 Navigation and Interaction

For effective information acquiring and understanding, navigation and user interaction are as important as presentation. We have embedded our visual presentation in an interactive system for the analysis of the funding related information. In this section, we describe the navigation and interaction techniques adopted by the visual funding navigator.

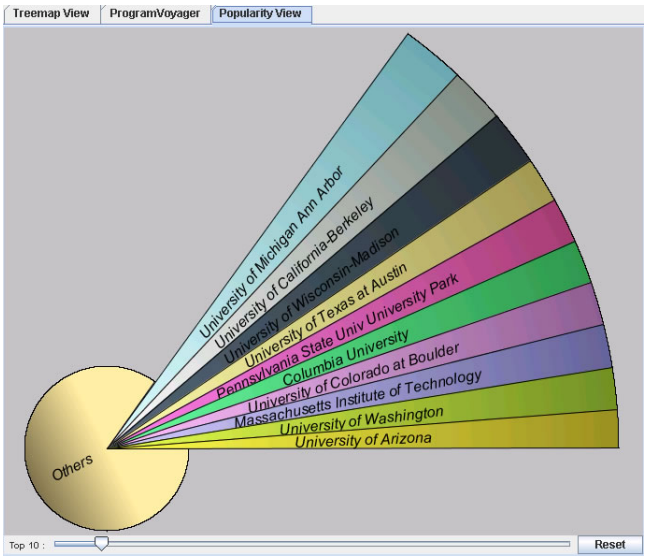

Figure 9: An overview of popularity view.

3.2.1 User Interface

The visual funding navigator contains two major parts: visual analysis part and information part. These two parts work together to help a user navigate and analyze the NSF funding related data. Figure 8 is an overview of the UI. The left part is for visual analysis, and the right part is for displaying information.

The goal of the visual analysis part is to help the user visually observe and understand the information for quick information finding and decision making. To achieve this goal, the following views are utilized: treemap view, popularity view, and ProgramVoyager view. The first two views focus on analyzing the static data, i.e., non-time-series data, and the last view is responsible for analyzing the time-series data.

In treemap view, we organize the NSF funding related data as the tree structures mentioned in Section 2. Hence, we can easily get an overview of the NSF funding data being displayed (see Figure 7).

The visual funding navigator enables users to get the top N popular research organizations which receive more grants than other organizations. By selecting a directorate, division, or program in the treemap view or ProgramVoyager view, the top N list in that unit is shown in the popularity view (see Figure 9).

ProgramVoyager view focuses on tracking program trends by utilizing stacked graph [10] to visualize the time-series data. Figure 10 is the visualization of the distribution of funding amount of each active program over time.

Furthermore, in order to help the user navigate and understand the NSF funding related data more easily, a information part is synchronized with the visual analysis part. This part displays search results or additional textual information in response to the exploration operations in the visual analysis part. It contains the following views:

- Attribute view: this views displays all the attributes of a selected item in the visual analysis part;
- Bookmark: in this view, the user can manage bookmark items and folders;
- Organization view: this view is used to display the research organizations and their corresponding projects which are granted by NSF;
- Search result: search result list is displayed in this view;

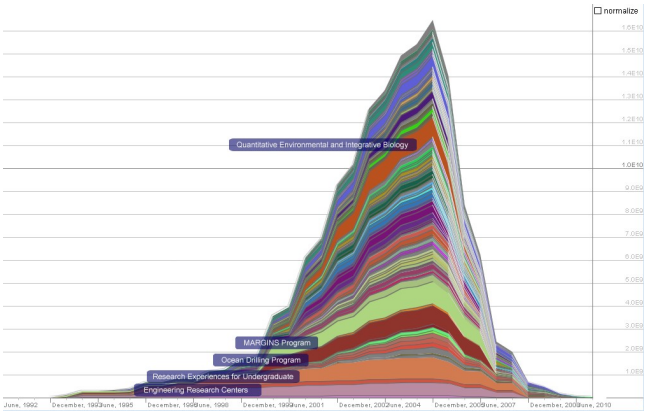

Figure 10: An overview of ProgramVoyager view.

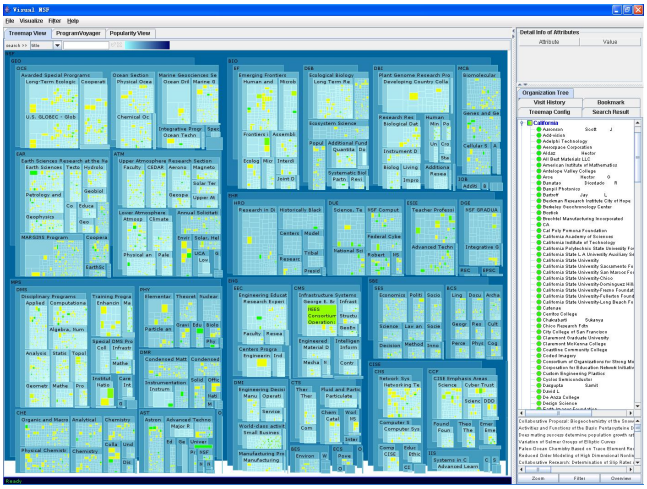

Figure 11: synchronization of multi views.

- Visit history: this view is utilized to display the top N visited awards.

- Treemap configuration: the user can customize the configurations, such as treemap color, search color, background color, etc, for the treemap view.

One major feature of this interface is that it enables coordinated visualization. The multiple views provided by these two parts work together seamlessly to help the user well understand and analyze the funding related data. Each view displays some certain aspects of data about just the object items in that view. When the user clicks on an object in one view to highlight it, other views will automatically highlight any related objects. For example, when the user clicks on an item, such as an award, in the visual analysis part, then the views in the information part will update automatically: the corresponding attribute items are displayed in the attribute view; while the visit history view will update the visiting information.

3.2.2 Search and Filter

The visual funding navigator includes an integrated support for search and filter. It supports keyword search of the funding related data. As the items become visible, their profile attributes are added to a in-memory search index. A search box allows users to type in search queries. The search results are color-coded by their ranking scores, highlighting the highly matched results (see Figure 13).

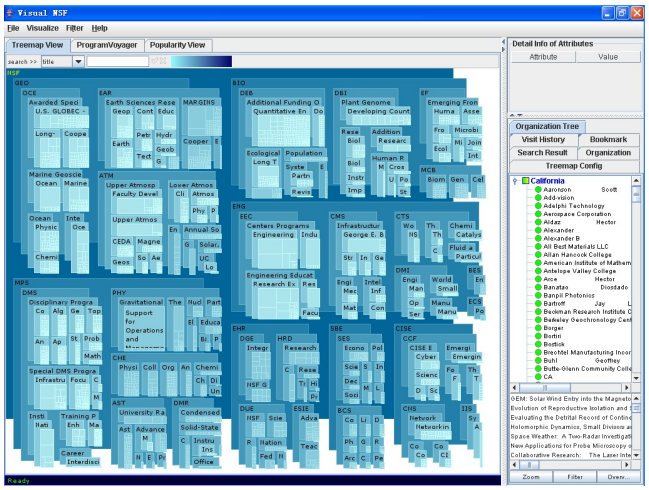

Figure 12: NSF awards of California.

Figure 13: Search function of the visual funding navigator.

The user can also select a project to see the detail information of this project. Furthermore, the user can find the related projects of a given project. Here, the results are color-coded by their ranking scores, also highlighting the highly matched results.

The user can navigate the tree, or click on the “Filter” menu to see a filtered view of the NSF funding tree. The Filter menu allows the user to filter the NSF funding tree in the following aspects: filter by the depth of the hierarchy structure, filter by funded year, filter by funding amount, and filter by program or project status. When filtering from low level nodes to high level nodes, more detail information is displayed on the high level nodes to enable informed exploration.

One interesting question is ”Which directorate(s), division(s), or program(s) offers more grants to the research organizations in a specific location, for example, in California?” In addition to just giving the unit list, we also provide the visual answer. When the user selects several research organizations from the organization tree (Here, locations, such as state, are used to form the organization tree.) in the organization views, the awards which belong to these organizations are highlight in the treemap view and ProgramVoyager view. In the treemap view, if the current leaf nodes are the awards, then the matched awards are highlighted; otherwise, the current leaf nodes are color-coded by the number of matched awards in it, highlighting the nodes which contains more matched

awards. In the ProgramVoyager view, the unmatched program stripes are displayed by the gray-color to un-highlight them, while the matched stripes are highlighted to attract the user's attention. Here we take the treemap view as an example. In Figure 11, when the user selects the research organization under node "California", all the awards belong to this set are highlighted in the treemap view. The user can also filter out the irrelevant awards and focus himself only on the awards which belong to the location he selected. Figure 12 shows the filtered result. In this figure, the user can easily find that nearly half of the awards in California are granted by directorates MPS (Directorate for Mathematical and Physical Sciences) and GEO (Directorate for Geosciences).

3.2.3 Animated Zooming

Animation can help catch a user's eyemaintain object constancy and thus improve task performance. To be able to explore the large hierarchies in the treemap view efficiently, we use an interactive smooth zooming to help users preserve their sense of position and context. When the user clicks on a non-leaf node, this toolkit will then smoothly zoom in to the selected node area. In other words, only the subtree of selected node are displayed. Such kind of zooming operation allows the user to focus on the funding data which belongs to this selected non-leaf item. The toolkit displays back the whole funding hierarchy (zoom out) when the user clicks on the left button again. Furthermore, when a user click a leaf node, a semi-transparent view with the detail description of the funding programs or proposals will dynamically expand to the whole screen, with the treemap structure shown as a background.

The similar animated zooming is also supported in the ProgramVoyager view. With such zooming operation, the user can drill down from summary data to detailed data continuously, thus encourage further exploration.

3.2.4 Bookmarks

One of the greatest challenges facing people who use large information spaces is to remember and retrieve items they have previously found and thought to be interesting. Bookmarks are the dominating approach to managing personal URL collections on the Web. They serve as convenient shortcuts to frequently used Web pages as well historical pointers to useful information that may otherwise be forgotten.

To help users to "re-find" information in the visual funding navigator, we introduce the bookmark into this toolkit to help the user remember and retrieve interesting funding information. With the bookmark function, the user can create their own personal information space and share it with others.

The major feature of the bookmark function is that it prioritizes bookmark items and folders according to the user's feedback over time. The following feedbacks are utilized to prioritize the bookmark items:

- Keystroke: important bookmark items should receive more keystrokes than the unimportant ones.

$$k_j^{(i)} = \frac{N_j^{(i)}}{N_s} \tag{1}$$

  Where $N_j^{(i)}$ is the keystroke received by the j-th bookmark item in the i-th iteration, and N_s is the total keystroke number received by all the bookmark items in the i-th iteration.

- Recency: important bookmark items should have interacted with the user more recently than the unimportant ones. The following declining function is used to measure the recency:

$$d_j^{(i)} = \frac{1}{\frac{time_j^{(i)}}{h_j^{(i)}} + 1} \tag{2}$$

  Where $h_j^{(i)}$ is the declining period in the i-th iteration, and $time_j^{(i)}$ is the time elapsed from last visited. Initially, the same declining period is set to all bookmark items. After the user's feedback, the periods of bookmark items are updated separately by their own time elapsed between two clicks.

By combining equations (1) and (2), we obtain the weighting function of bookmark item j.

$$w_j^{(i)} = \alpha_1 \times w_j^{(i-1)} + \alpha_2 \times k_j^{(i)} + \alpha_3 \times d_j^{(i)} \tag{3}$$

where

$\alpha_1, \alpha_2, \alpha_3 \geq 0,\ \alpha_1 + \alpha_2 + \alpha_3 = 1,\ w_0^0 = w_1^0 = ... = w_{n-1}^0 = 0$

Furthermore, the weight of the bookmark folder is the sum of the weights of the items contained in this folder.

4 Visual Analysis of NSF Research Trends

One major feature of this toolkit is that it provides a hybrid analysis that involves both static and dynamic analysis to help the user well understand the funding related information and reveal the research trend of NSF organization. In this section, we will introduce these two kind of analysis in detail.

4.1 Static Analysis

Static analysis will focus on the analysis of non-time-series data. Here we mainly focus on analyzing the funding related hierarchical data. In the visual funding navigator, we use two ways to organize such kind of data: by the NSF organization structure and by the locations where the investigators are located.

Treemaps are effective for trees where the nodes include quantitative variables, particularly when large values are important. Since the awarded amount is very important to both NSF officers and researchers, therefore treemaps are utilized to visually analyze such kind of data. In our treemaps of NSF funding data, the sizes of the leaf rectangles reflect the funding amounts of the awards. As for the non-leaf rectangle, its size reflects the cumulative funding amount of the awards in its children. Thus such kind of treemaps provide a snapshot of the NSF funding distribution. The user can easily find

- the award which has the largest funding amount;
- the popular research topics or locations which contains much more funding amount than others.

Here we use an example to illustrate the static analysis function offered by treemap visualization. Figure 7 is the visualization result of the currently active awards (31159 awards) on NSF website. In this figure, the user can easily find that MPS (Directorate for Mathematical and Physical Sciences) and GEO (Directorate for Geosciences) are the two largest directorates in NSF, which offer more grants than other directorates. Thus the research areas they are funding are much more popular than other research areas.

Furthermore, this toolkit provides an overview of the top N popular research organizations (see Figure 9) which receive more grants than other organizations, and the top N visited projects ((the visit history view in Figure 8)). In this way, the user can quickly discover the influential research organizations and projects in a particular topic area.

In the popularity view (see Figure 9), the focus + context technique is adopted to improve the interaction of the traditional pie charts. Users can select some of the interested slices and treat them

as the focus. Here we select the top N popular research organizations as the focus. The focus will be emphasized by increasing its radius and meanwhile decreasing the radius of other slices. Thus users can have a better view of the focused area, including its structure and the small slices.

4.2 Dynamic Analysis

Dynamic analysis will focus on the analysis of time-series data. Such kind of analysis accounts for the fact that data taken over time may have an internal structure (such as autocorrelation, trend or seasonal variation) that should be accounted for. In our toolkit, the stacked graph [10] is utilized to dynamically analyze the time-series data embedded in the funding information, i.e., the time distribution of the funding amount for each of the 459 active NSF programs from 1992 to 2011.

The method to visualize the time-series data is as follow: given a set of the time distribution of the funding amount, a set of stacked graphs is produced, as in Figure 10. The x-axis corresponds to date, and the y-axis to the total funding amount for all the active programs currently in view. Each stripe represents a program, and the thickness of a stripe is proportional to its funding amount at the given time step.

The above visualization method will help the user track the NSF program trends, thus help the user dynamically analyze the focus topic of interest. More exactly, it provides the following analysis capabilities.

First, it allows the user to see and analyze the distribution of funding amount over time. Then the user can easily detect which programs are the currently popular ones and which programs are much more popular than before. For example, in Figure 10, programs "Quantitative Environmental Integrate Biology", "Margins Program", "Ocean Drilling Program", "Research Experiences for Undergraduates", and "Engineering Research Centers" are the currently most popular programs. They received more funding in 2006.

5 Conclusions

In this paper, we present a visual funding navigator, which can help the user to navigate and analyze the funding data on the NSF official website. This toolkit allows the user to take the advantages of treemap visualization and stacked graph to understand the funding related information more quickly.

One major feature of this toolkit is that we have designed a new kind of treemaps, 2.5D labeled treemaps, to provide a clear view of hierarchies. The main novelty of the proposed method is to use cascading rectangles to represent the hierarchy instead of the nesting ones. We also design a new layout algorithm to address the problem with the proportioned area of rectangles caused by the increased margin space and label space. Furthermore, an incremental layout method is adopted to handle information on a large scale.

Another major feature of this toolkit is that it provides hybrid analysis that involves both static and dynamic analysis to help the user well understand the funding related information and reveal the research trend of NSF organization.

We regard the work presented as initial, there are improvements to be made as well as many directions to pursue. The future work will be focused on applying these techniques in a more traditional analytical context and conducting more rigorous evaluations. As the online proposal collection continue to spread and diversify, we also anticipate the need for additional design approaches balancing utility and holistic experience, enabling information visualization technologies to take on an increasingly important role in both analyzing the online proposal collection and detecting potential opportunities.

6 Acknowledgments

We would like to thank Martin Wattenberg for providing the source code of NameVoyager. We would like to thank Bernice E Rogowitz for constructive suggestions on the Treemap layout. We would also like to thank Haiyan Guo and Xinghua Lou for their advice and suggestion on the UI design.

References

[1] B. Johnson and B. Shneiderman., "Treemaps: a space-filling approach to the visualization of hierarchical information structures," in *Proceeding of the 2nd International IEEE Visualization Conference*, 1991, pp. 284–291.

[2] B. Shneiderman, "Treemaps for space-constrained visualization of hierarchies," http://www.cs.umd.edu/hcil/treemap-history/.

[3] G.-A. Jungmeister and D. Turo, "Adapting treemaps to stock portfolio visualization," Technical Report UMCP-CSD CS-TR-2996, University of Maryland, College Park, Maryland 20742,USA, 1992.

[4] M. Wattenberg, "Visualizing the stock market," in *CHI '99 Extended Abstracts on Human Factors in Computing Systems*, 1999, pp. 188–189.

[5] D. Turo, "Hierarchical visualization with treemaps: Making sense of pro basketball data," in *Conference companion on Human factors in computing systems*, 1994, pp. 441–442.

[6] L. Jin and D. C. Banks, "Hierarchical visualization with treemaps: Making sense of pro basketball data," *IEEE Computer Graphics and Applications*, vol. 17, no. 4, pp. 63–65, July/August 1997.

[7] T. Asahi, D. Turo, and B. Shneiderman, "Visual decision-making: using treemaps for the analytic hierarchy process," in *Conference companion on Human factors in computing systems*, 1995, pp. 405 – 406.

[8] A. Orso, J. Jones, and M. J. Harrold, "Visualization of program-execution data for deployed software," in *Proceedings of the ACM Symposium on Software Visualization*, 2003, pp. 67–76.

[9] M. Balzer, O. Deussen, and C. Lewerentz, "Voronoi treemaps for the visualization of software metrics," in *Proceedings of the ACM Symposium on Software Visualization*, 2005, pp. 165–172.

[10] M. Wattenberg, "Baby names, visualization, and social data analysis," in *Proceeding of IEEE Symposium on Information Visualization 2005 (InfoVis2005)*, 2005, pp. 1–7.

[11] H. L. SX Liu, N Cao and H. Su, "The visual funding navigator: Analysis of the nsf funding information," in *Proceeding of ACM fifteenth Conference on Information and Knowledge Management*, 2006 (accepted).

[12] S. G. Eick, "Visualization and interaction techniques," in *CHI97 Tutorial notes on Information Visualization. ACM SIGCHI*, 1997.

[13] J. van Wijk and H. van de Wetering, "Cushion treemaps - visualization of hierarchical information," in *Proceedings 1999 IEEE Symposium on Information Visualization (InfoVis99)*, 1999, pp. 73–78.

[14] K. H. Mark Bruls and J. J. van Wijk., "Squarified treemaps," in *Proceedings of the Joint Eurographics and IEEE TCVG Symposium on Visualization*, 2000, pp. 33–42.

[15] T. Bladh, D. Carr, and J. Scholl, "Extending tree-maps to three dimensions: a comparative study," in *Proceedings of the 6th Asia-Pacific Conference on Computer-Human Interaction (APCHI 2004)*, 2004, pp. 50–59.

[16] B. Shneiderman, "Tree visualization with tree-maps: 2-d space-filling approach," *ACM Transaction on Graphics*, vol. 11, no. 1, pp. 92–99, January 1992.