Balloon Focus: a Seamless Multi-Focus+Context Method for Treemaps

Ying Tu and Han-Wei Shen

Abstract— The treemap is one of the most popular methods for visualizing hierarchical data. When a treemap contains a large number of items, inspecting or comparing a few selected items in a greater level of detail becomes very challenging. In this paper, we present a seamless multi-focus and context technique, called Balloon Focus, that allows the user to smoothly enlarge multiple treemap items served as the foci, while maintaining a stable treemap layout as the context. Our method has several desirable features. First, this method is quite general and can be used with different treemap layout algorithms. Second, as the foci are enlarged, the relative positions among all items are preserved. Third, the foci are placed in a way that the remaining space is evenly distributed back to the non-focus treemap items. When Balloon Focus enlarges the focus items to a maximum degree, the above features ensure that the treemap will maintain a consistent appearance and avoid any abrupt layout changes. In our algorithm, a DAG (Directed Acyclic Graph) is used to maintain the positional constraints, and an elastic model is employed to govern the placement of the treemap items. We demonstrate a treemap visualization system that integrates data query, manual focus selection, and our novel multi-focus+context technique, Balloon Focus, together. A user study was conducted. Results show that with Balloon Focus, users can better perform the tasks of comparing the values and the distribution of the foci.

Index Terms—Treemap, focus+context, multi-focus, fisheye, magnification, visualizing query results, multi-scale viewing.

1 Introduction

The treemap is one of the most popular methods for visualizing hierarchical data. By dividing the display area into rectangular areas recursively according to the hierarchical structure and a user-selected data attribute, treemaps can effectively display the overall hierarchy as well as the detailed data values.

When the treemap is used to visualize very large scale data sets [7, 17], being able to visualize user selected data items, e.g. query results, becomes crucial. This capability allows viewers to focus on only a subset of items in which they are most interested. To achieve this goal, the main issue to address is how to highlight the selected items with details while displaying the contextual information.

There exist several options to display user selected focus items. One is to display the selected data only, either as a list of entries in a separate view, or by constructing a new treemap containing only the focus items, and their ancestors and descendants. These methods cannot effectively display the global context around the selected items such as their positions in the original treemap or their relations to the non-selected items. The key drawback is that the new view of the data may look very different from the original treemap, which forces the viewers to put extra efforts to identify and link the two views.

Another category of methods utilizes focus+context techniques to highlight the selected items within the main treemap view to preserve the context. An example is presented in [1], where two focus+context techniques are provided. The cue-based technique highlights the foci with bright colors and suppresses the non-focus items with muted colors. The zoomable interface can display more details in the user-selected sub-regions, which may contain some of the foci.

Neither cue-based techniques nor zoomable interfaces are sufficient in the general cases when there are multiple foci distributed in a large hierarchy. Color highlighting is effective only if the focus items are large enough to be clearly seen, but offers little help when the foci are too small. However, it is very common for a typical treemap to contain many very small items, considering the trend of using treemaps to visualize large hierarchies such as disk file systems [7], worldwide network traffic and social cyberspace data [17]. Zoomable interfaces are designed mainly for navigating in a large hierarchical view of data with a single focus or a few very closely clustered foci. However, since multiple foci can scatter across the entire treemap, zooming in one sub-region will lose the information in other regions. Therefore, it is not suitable for general applications with multiple foci such as visualizing query results.

It is highly desirable to have a *seamless focus+context method*, which can enlarge the foci while still keeping a similar global view. A typical example is the fisheye view, which provides a good balance between the local detail at the focus of the viewer's attention and the global context [8]. The advantage of this type of technique is that tasks such as comparing the contents among the foci, and observing the distribution of foci in the treemap, become much easier. The cost of tracking objects that undergo transformations when the foci are enlarged is also reduced.

To the best of our knowledge, no previous work has tackled the problem of developing seamless focus+context techniques for treemaps with multiple foci. An ideal multi-focus focus+context technique for the treemap needs to have several unique features, as discussed below, which cannot be easily handled by the existing focus+context techniques designed for other data types such as graphs or 2D maps [20, 21].

The specific desired features for treemaps include preserving the items' rectangular shape, being independent of the underlying treemap layout algorithms, and maintaining the same relative positions among the treemap items after the focus items are enlarged. In addition, the focus items should be made as large as the user desires, while the resulting treemap still has a stable and consistent appearance so that the user can easily track individual items. Detailed discussions on these features are provided in Section 3.1.

In this paper, we present *Balloon Focus*, a seamless focus+context technique for multi-focus treemaps that has the desired features aforementioned. To preserve the shapes and relative positions among the treemap items, a DAG (Directed Acyclic Graph) is created to represent the positional dependency constraints.

With the dependency graph, an elastic model is devised to govern the placement of treemap items when the foci are enlarged. We categorize the edges of the graph into two groups: *solid edges*, representing the focus items since the size of a focus is known given the zoom factor, and *elastic edges*, representing the size of the non-focus items

• *Ying Tu is with the Computer Science and Engineering Department at the Ohio State University, E-mail: tu@cse.ohio-state.edu.*
• *Han-Wei Shen is with the Computer Science and Engineering Department at the Ohio State University, E-mail: hwshen@cse.ohio-state.edu.*

Manuscript received 31 March 2008; accepted 1 August 2008; posted online 19 October 2008; mailed on 13 October 2008.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org.

computed based on where the focus items are placed. Elastic edges are similar to the spring coils in that they are compressible and the energy in the system is determined by the lengths of the springs. we build a linear system based on the forces of the spring coils to solve the positions of all treemap items.

We created a treemap visualization system that integrates data query, manual focus selection, and our novel Balloon Focus technique together. A case study on visualizing NBA statistics data using our system is provided in the paper. In addition, a user study was conducted, in which 12 subjects were asked to perform various visual analysis tasks. The results show that Balloon Focus can help the viewers improve accuracy and reduce the time required to perform the analysis.

The rest of this paper is organized as follows: In Section 2, we briefly review the related work on the focus+context techniques. In Section 3, we provide an in-depth analysis of the problem and requirements of multi-focus seamless focus+context techniques for treemaps. In Section 4, we describe our Balloon Focus algorithm in detail. In Section 5, we present a case study to demonstrate the usefulness of Balloon Focus. Our user study is described in Section 6. Finally, we summarize our work in Section 7.

2 Related Work

2.1 Focus+Context

Focus+context techniques have been used for various information visualization applications. For example, it has been shown that focus+context techniques can be used to assist visualization of trees [15, 18], graphs [20, 9], line graphs [14], maps [4], and tables [24].

Cockburn et al. [6] categorized focus+context approaches into four groups: spatial separation, typified by overview+detail interfaces; temporal separation, typified by zoomable interfaces; seamless focus+context, typified by fisheye views; and cue-based techniques which selectively highlight or suppress items within the information space.

Seamless focus+context Among the four categories, the seamless focus+context technique, typified by fisheye views, is a widely studied topic. Sometimes, the term focus+context is interchangeable with many other terms in that group, such as fisheye [8, 20], detail-in-context [11], nonlinear magnification transformation [13] or distortion [16], multi-scale [4] and others. The techniques can be further categorized into single-focus or multi-focus techniques.

Single-Focus Fisheye Single-focus fisheyes can help people navigate or browse effectively. They are often compared with another effective navigation method, pan&zoom, for various applications, as described in [19, 23].

Multi-Focus Fisheye Because of its usefulness, multi-focus techniques are a popular topic for the focus+context research. Works related to multi-focus that use image space approaches were proposed by Keahey [13, 11, 12], Carpendale [4, 5], and so on. In the area of graph drawing, Sarkar proposed the well known graphical fisheye [20] and the Rubber Sheet [21]. For radial space-filling hierarchy visualizations, InterRing [28] and Sunburst [25] include multi-focus techniques as an important feature. Schaffer et al. [22] and Toyoda et al. [26] studied multi-focus in the context of nested networks.

2.2 Focus+Context on Treemaps

Existing treemap systems such as the one described in [1] have adopted zoomable interfaces and cue-based techniques.

For seamless focus+context, Shi et al. [23] proposed a distortion algorithm by increasing the size of a node of interest while shrinking its neighbors. Because their work was focused on browsing in a treemap, it is not straightforward to extend their algorithm to multi-focus applications. Keahey [12] used a treemap as an example to show how to compound zooming with a graphical fisheye. In that method, the treemap is essentially treated as an image.

Comparing with the previous work, our method is focused on offering several desired features of seamless focus+context on treemaps. To the best of our knowledge, our work is the first to tackle the multi-focus problem for treemaps. In addition, our method can be applied to any other rectangular space filling visualization method.

3 Problem Analysis

In this section, we analyze and identify the desired features for seamless focus+context techniques applied to treemaps. These features are used as the principles to guide the design of our algorithm.

3.1 Desired Features

Preserve the treemap's most prominent property. The most prominent property of the treemap is that the entire space is filled with rectangular leaf items and optional hierarchy-highlighting borders. It is important to preserve this property since space filling makes the best use of the available screen area to display data, and rectangles make area comparisons easier. In addition, some of the techniques developed previously were based on the assumption that the items are rectangular, such as using bar charts [10] and images [2] to show the item content. Therefore, preserving this property is critical to the general usability of treemaps.

Apply the same scaling to all foci. There are two important reasons to uniformly scale up all foci. First, because the foci are equally important, they are supposed to change in the same way. Failing to do so may cause confusion since viewers may assume that the zoom factor implies importance. Second, when performing analysis tasks, users often need to compare the areas of foci. Uniform scaling will allow viewers to get the same comparison result as in the original treemap.

Be layout-algorithm-independent. Different treemap layouts are designed for different purposes, and not a single existing layout has been shown to be the best in all cases. For example, the squarified treemap is to optimize visibility, and the ordered treemap is to preserve the order between sibling items. In addition, with the growing popularity of treemaps, new layout algorithms may as well be proposed in the future to meet new requirements. To maximize the usability, a layout-algorithm-independent focus+context algorithm is highly desired.

Preserve positional dependency between items. For the purpose of treemap stability and visual consistency, it is important to preserve the relative positions, or called positional dependency, between the treemap items. For example, if *item A* is originally on the upper left side of *item B*, after enlarging the foci, *item A* should still be on the upper left side of *item B*. One straightforward way to scale up the focus items is to change their size attributes and rerun the layout algorithm, similar to the method in [27]. However, doing so will cause rapid and abrupt layout changes for most of the existing layout algorithms when the zoom factor is being adjusted interactively. These layout changes will lead to flickering, which will draw attention away from other aspects of the visualization, as have been pointed out by Bederson et al. in [3].

Maximize Foci's Possible Zoom Factor. A larger zoom factor allows viewers to see more details of the foci in the visualization and thus help visual analysis tasks. In addition, it allows viewers to select more foci and still see all foci clearly. It is specially desirable for visualizing large-scale data, where the items can be very small.

Scale down non-focus items as even as possible. As the foci are scaled up, the non-focus items need to be compressed. Although the non-focus items are not as important as the focus items, it is desired that the scaling factors among the non-focus items be as uniform as possible so as to maintain the treemap's visual consistency.

3.2 Problem Statement

Figure 1 illustrates the basic idea of the desired effects for seamless multi-focus+context treemaps. The treemap is a one-level strip treemap with 37 leaf items. Four foci in color are selected and scaled up. The example demonstrates the features mentioned above. We can

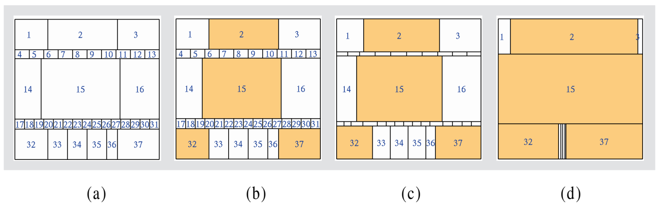


Fig. 1. Desired effects of the seamless focus+context technique on treemaps. (a) the original treemap. (b) foci are selected and colored. (c) foci are enlarged slightly. (d) the state when the foci are maximized.

observe that the items' rectangular shape is preserved, the positional dependencies are preserved, foci are uniformly scaled up, and non-focus items are uniformly scaled down.

In terms of maximizing the zoom factor, we can see that the width of *item 15* in Figure 1(d) is as wide as the width of the entire treemap; it is impossible to have a larger zoom factor along the horizontal dimension, so clearly the factor is maximized. If the orthogonal stretching technique in the *Rubber Sheet* [21] is used, the foci in Figure 1(b) cannot be enlarged along the horizontal dimension. This is because the horizontal projection of the foci completely covers the x-axis.

Here is the statement of our problem: Given a treemap, $TM_{original}$, with multiple selected focus items in different levels of the hierarchy, the focus+context algorithm should transform the $TM_{original}$ into a new treemap, $TM_{transformed}$, in which all focus items are enlarged by the same zoom factor R. The achievable zoom factor should be as large as possible. The focus items' relative positions should be preserved. Furthermore, the non-focus items should be scaled down as evenly as possible among themselves.

4 Approach

In this section, we describe the multi-focus+context algorithm for treemaps. Section 4.1 describes how to capture the prominent positional dependency among the items in a treemap when the foci are enlarged. Section 4.2 introduces the concept of using a graph to model the positional dependency. Section 4.3 shows how to model the dependency for a multi-level treemap. Section 4.4 describes an elastic model used to determine the final positions of all treemap items. Finally in Section 4.5, we briefly discuss the implementation issues. Throughout this section, the example in Figure 2 is used to illustrate our algorithm.

4.1 Positional Dependency

The goal of the dependency model is to capture the positional dependency constraints among the items in a treemap, so that the transformation algorithm can provide a consistent look between the original and new treemaps, denoted as $TM_{original}$ and $TM_{transformed}$. The more smoothly the original treemap can be transformed to the new treemap, the more easily the viewers can adapt to the new view.

Ideally, to provide the best layout consistency, the positional relations among all boundary edges should be preserved. For example, in the treemap shown in Figure 2 (a), the total order of vertical edges from left to right is: $g \prec c \prec e \prec h \prec a \prec f \prec i \prec b \prec d$. With the total orders of both vertical and horizontal edges preserved, the layouts of $TM_{original}$ and $TM_{transformed}$ will be very similar and thus have a consistent look. The maximum zoom factor allowed, however, is quite limited with this total order constraint. As shown in Figure 2 (b), the enlargement of the foci (*item* 1, 5, and 13) must be stopped when edges a, f, and i, have the same x coordinate. Clearly preserving the total order positional constraints restricts the maximum zoom factor for the foci to a small value. To overcome this limitation, we relax the dependency constraints, and propose a relatively loose dependency without negatively affecting the layout consistency.

When we studied the relation between preserving the positional dependency and the easiness for viewers to keep track of changes, we found that the positional relations between a focus and its nearby items play a much more important role than the positional relations between two arbitrary items, especially those far from each other. This is because when the foci are enlarged, the viewers are more sensitive to what happen to the foci and items in the foci's neighborhood.

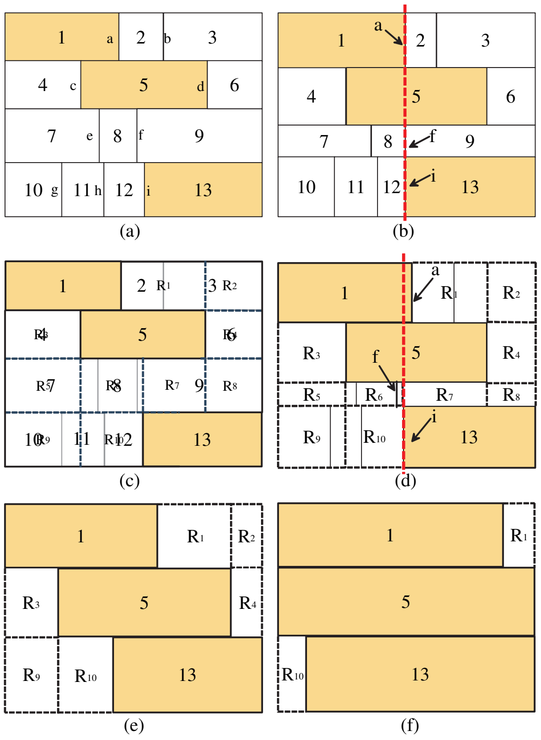


Fig. 2. One-level treemap example: (a) is the original treemap with three selected foci. (b) shows the maximum enlargement of the chosen foci based on the ideal positional dependency. (c) illustrates how the entire treemap area is divided into regions based on the foci. (d), (e), and (f) demonstrate how the foci are enlarged based on the region dependency, showing the beginning of enlargement, maximum enlargement with aspect ratios of foci kept the same, maximum possible enlargement, respectively. Numbers represent treemap items; lower case letters represent edges; dashed lines represent focus lines; R_i represents an enclosure.

Based on this observation, we propose a *region dependency* to capture only the prominent positional dependency, which is between every focus and the treemap items in its neighborhood. What we want to guarantee is the dependency among the regions, called *enclosures*, whose formal definition is given below. We define the following terms:

Focus edges. We define the focus edges of a focus item as its four boundary edges.

Focus lines. We define the focus line for a focus edge as its linear expansion in its both directions. The expansion stops when the line reaches another focus item or the boundary of the parent of the focus.

Enclosures. We define an enclosure in the treemap as the area that is enclosed by two vertical focus lines and two horizontal focus lines. There are no other focus lines going through this area.

As depicted in Figure 2 (c), the entire treemap is divided into 13 visible *enclosures* by the focus lines of the selected foci (*item* 1, 5, and 13): Obviously the foci themselves are enclosures, but there are also ten more outside of the foci (labeled R_1 through R_{10}). When focus lines overlap, such as the bottom line of R_1 and the top line of R_5, they create invisible enclosures whose area is zero.

When enlarging the foci, instead of preserving the total order of all item edges, we preserve the region dependency with respect to the enclosures. As shown in Figure 2(d), (e), and (f), after the foci are enlarged, each treemap item edge strictly stays in its original enclosure;

each pair of adjacent enclosures strictly keeps their positional relation.

However, when observing the map item edges, we can see that, as shown in Figure 2 (d), the order between some edges is not preserved. For example, compared to Figure 2 (a), *edge a* changes its position from the left side of *edge f* and *i* to their right. Even with those changes, still, from the viewer's perspective, the relative positions among all treemap items are not changed much; the transformed treemaps look consistent with the original one.

The main advantage of this region dependency is that it allows a much larger zoom factor. Figure 2 (e) shows the maximum zoom factor when the aspect ratio of the foci is preserved, and (f) shows the maximum possible zoom factor. The algorithm to enlarge foci will be discussed in Section 4.4.

We define that the region dependency is maintained after the treemap transformation if all the enclosures in $TM_{original}$ are preserved in $TM_{transformed}$, and no new enclosure is created in $TM_{transformed}$. Note that we allow the size of the non-foci enclosures to reduce to zero. The formal definition of a region dependency is as follows:

Representation of an enclosure. We define the representation of an enclosure as a 4-tuple, $<V_{left}, V_{right}, H_{top}, H_{bottom}>$, consisting of the four focus lines that bound the enclosure.

Enclosure set of a treemap. We define the enclosure set of a treemap TM, $EncSet(TM)$, to be the set of all enclosures in the treemap.

Region dependency. We define that the region dependency is maintained between $TM_{original}$ and $TM_{transformed}$ if and only if $EncSet(TM_{original}) = EncSet(TM_{transformed})$.

The region dependency implies an important property that the order between two vertical or two horizontal focus lines is preserved if the focus lines bound an enclosure together. This property guarantees the prominent relative positions of the items to be preserved.

Although the region dependency only explicitly exerts constraints to the focus lines/edges, the dependency implicitly restricts that the non-focus edges stay in the same enclosure which they originally belong to, and all edges in an enclosure strictly keep the relative positions with one another. With these constraints, we can consider the inside of an enclosure as texture. Once the enclosures' new positions are decided, the textures are mapped, and the transformed treemap is created.

4.2 Dependency Graph

We introduce a directed graph, called *dependency graph*, to model the region dependency. As shown in Figure 3 (a), a treemap has two dependency graphs that model the horizontal and vertical edge dependency separately. The nodes in the graph represent focus lines. For any enclosure, there exists an edge in the graph that connects two nodes representing the focus lines which bound this enclosure along the vertical or horizontal direction. The edge represents the space between the two focus lines, and the direction of the edge represents the order of the two nodes. As a result of the partial dependency order, a DAG (Directed Acyclic Graph) is created as the dependency graph.

Since an edge is identified by two nodes, if multiple enclosures introduce the same edge, redundant edges are removed.

In the dependency graph shown in the bottom of Figure 3 (a), along the horizontal dimension, vertical focus lines (V_1 through V_6) are represented as nodes. There are three edges connected to the node V_4: $e(V_2,V_4), e(V_4,V_5)$, and $e(V_4,V_6)$, which are introduced by the enclosure R_6 or R_{10}, R_7, and the focus *item* 13, respectively. Note that $e(V_2,V_3)$ is introduced by an invisible enclosure.

Figure 3 (b) depicts the dependency graph for the treemap of $TM_{transformed}$. The dependency graphs are topologically the same before and after transformation. Therefore, we say that the region dependency is maintained.

4.3 Multi-level Treemaps

In a multi-level treemap, the focus items can be either internal nodes or leaf nodes in the tree. To construct a dependency graph for a multi-level treemap, the basic idea is to nest the local dependency graphs

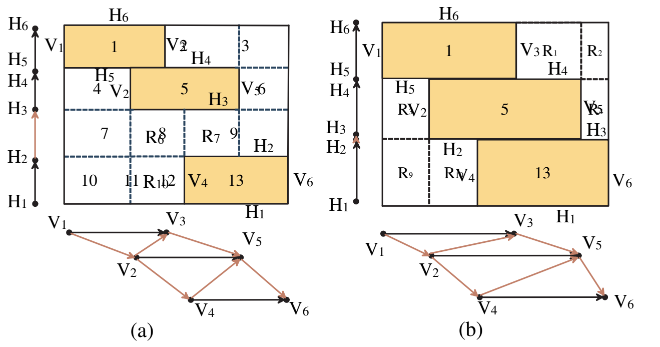


Fig. 3. Dependency graph example: (a) shows the dependency graph for both dimensions in $TM_{original}$. (b) shows that the positional dependencies are preserved in $TM_{transformed}$. The graph on the left side of a treemap is for the dependency of horizontal edges, and the bottom graph is for the dependency of vertical edges. Different colors are used to distinguish the edges introduced by a focus or non-focus enclosure.

into a global dependency graph. Figure 4 shows an example of constructing a nested dependency graph for the horizontal dimension. In this example, *node* 5 and 13 are not selected as foci but some of their children are selected instead.

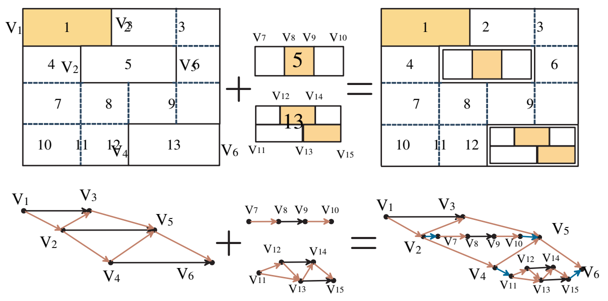


Fig. 4. The Dependency graph for a multi-level treemap.

We consider that for an internal item T and its parent P, if the descendants of T include any focus items, from P's point of view, T is a focus. So in P's *local dependency graph*, which represents the dependency among P's children, T is represented by an edge in the graph.

To construct a dependency graph for a multi-level treemap, we first calculate the local dependency graph for every internal item who has focus descendants, then link the graphs to construct a global dependency graph. When a lower level graph is linked in, an edge in the upper level graph is replaced by the linked-in graph with two *margin edges*. Margin edges are introduced to bridge two neighboring-level graphs, representing the margins (or borders) between two neighboring levels in the treemap. The treemap margin is to differentiate levels and help viewers understand the hierarchical structure. In the example shown in Figure 4, $e(V_2,V_5)$ and $e(V_4,V_6)$ are replaced, and four margin edges are added such as $e(V_2,V_7)$ and $e(V_{10},V_5)$.

4.4 Elastic Model

As discussed in Section 3.1, two of the desired features for a seamless multi-focus+context technique on treemaps are a uniform zoom factor for all focus items, and an even distribution of the remaining space to the non-focus items. To achieve these goals, we propose an elastic model to govern the space distribution.

Our elastic model is based on an observation of spring coils. In a system consisting of multiple spring coils of the same material that are connected together, the springs would shrink uniformly, i.e. proportionally to their lengths, when the total length is compressed. We found the spring coils can be used to represent the non-focus items, which need to be compressed as the focus items are enlarged.

In the following section, we describe in detail how to make the physical model from a dependency graph and how to solve the physical model using linear equations. After the x coordinates of vertical focus lines and y coordinates of the horizontal focus lines are solved independently, we have the new positions of the enclosures.

4.4.1 Physical Model

We classify the edges in the dependency graph into two types: the *solid edges*, and the *elastic edges*. Solid edges are the edges introduced by the focus enclosures, i.e. focus items, and they are solid because their lengths have been determined by their original lengths and the viewer-defined zoom factor. The margin edges are also considered as solid edges, in that their lengths stay unchanged with foci enlargement. Keeping the margin size is to achieve a consistent look from $TM_{original}$ to $TM_{transformed}$. The rest of the edges in the dependency graph are elastic edges, introduced by non-focus enclosures. The lengths of the elastic edges cannot be straightforwardly calculated, because they depend on the lengths of other edges.

On top of the dependency graph, we build a physical system. In this system, the solid edges are modeled as solid sticks and the elastic edges are modeled as spring coils with a uniform *elastic modulus EM*. Naturally the nodes are the joint points of multiple sticks and springs.

With a feasible zoom factor, the physical system should be in its rest state, i.e., the sum of all the forces acting on any joint point is zero. Figure 5 shows an example from a segment of a dependency graph, where n_i is connected with five nodes, n_1 through n_5.

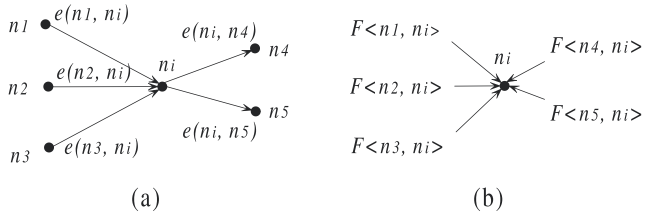


Fig. 5. (a) 5 edges are linked to the node n_i; the arrows represent the edge directions in the graph. (b) All the arrows point to n_i, representing the forces exerted on n_i. Note that the arrows do not necessarily indicate the positive directions of the forces.

For this example, we have the following equation to describe the equilibrium state of the node n_i, where $F_{<n_j,n_i>}$ represents the force that the edge linking n_i and n_j exerts to the point n_i.

$$F_{<n_1,n_i>} + F_{<n_2,n_i>} + F_{<n_3,n_i>} + F_{<n_4,n_i>} + F_{<n_5,n_i>} = 0$$

In addition, springs should follow Hooke's Law, which specifies the elastic force.

$$F = -EM \times (L_{new} - L_{original})/L_{original}$$

In this equation, $L_{original}$ and L_{new} represent the lengths of an elastic edge in the original state and the compressed state, respectively, and F is the elastic force along the spring. When $L_{new} = 0$, this equation may not apply, because in this case the spring becomes a solid point, and the force along the spring can be larger than $-EM \times (L_{new} - L_{original})/L_{original}$.

4.4.2 Solving the Model by Solving Linear Equations

We use a group of linear equations to describe the above physical model. First, we define the positions of the nodes and the forces along the edges as variables. So for a dependency graph with $|V|$ nodes and $|E|$ edges, we have in total $|V| + |E|$ variables. Specifically, the coordinate of a node n_i is denoted as P_{n_i}. For the edge $e_{(n_i,n_j)}$, the force that it exerts on the node n_j is denoted as $F_{<n_i,n_j>}$, and the force on n_i is denoted as $F_{<n_j,n_i>}$. Obviously we have $F_{<n_j,n_i>} = -F_{<n_i,n_j>}$. The original length of $e_{(n_i,n_j)}$ is denoted as $L_{<n_i,n_j>}$; $L_{<n_i,n_j>} = P_{n_j} - P_{n_i}$ in the original state.

For each node in the graph, n_i, except for the first and last nodes which represent the boundaries, we have the following equation:

$$\sum_{n_j:LinkedWith(n_i)} F_{<n_j,n_i>} = 0 \qquad (1)$$

And for the first and last nodes, we know the exact coordinates:

$$P_{n_{first}} = P_{begin} \quad P_{n_{last}} = P_{end} \qquad (2)$$

For each solid edge from n_i to n_j, we have the following two equations for margin edges and the rest solid edges respectively:

$$P_{n_j} - P_{n_i} = L_{<n_i,n_j>} \qquad (3)$$

$$P_{n_j} - P_{n_i} = L_{<n_i,n_j>} \times Factor_{ZoomIn} \qquad (4)$$

For each elastic edge from n_i to n_j, we have the following equation, where EM is a constant for all spring coils.

$$F_{<n_i,n_j>} = -EM \times (P_{n_j} - P_{n_i} - L_{<n_i,n_j>})/L_{<n_i,n_j>} \qquad (5)$$

The total number of equations is also $|V| + |E|$.

With these equations, we can solve the system by a typical linear system solver. The system can have three possible cases:

No solution. If the system has no solution, it means the viewer-defined zoom factor is not achievable, i.e. the factor is too large.

Single unique solution. If the system has a single solution, it means the viewer-defined zoom factor is feasible.

Multiple solutions. If the system has multiple solutions, it means the viewer-defined zoom factor is feasible as well. It also indicates that there are some solid edges on which the forces can be different values in different solutions, but the resulting node coordinates are the same. In this uncommon case, we can just choose any of the solutions.

4.4.3 Zero-Length Handling

For any elastic edge, we have to know when it will reach the critical point, that is, when its two end points meet and hence it has a length of zero. In this case, the elastic edge cannot be compressed any further, otherwise the edge length would be negative. An elastic edge e may reach its critical point when the focus items are enlarged to a certain zoom factor. $CRate_e$ denotes this critical zoom factor for e. When e reaches its critical point, all the edges whose $CRate$ is smaller than $CRate_e$ have already reached their critical points.

Because the critical zoom factors for elastic edges are determined by the nature of the physical model, we can calculate $CRate$ for all elastic edges immediately after the foci are selected. Then when the users are adjusting the zoom factor, we identify the edges that have zero lengths, i.e., whose $CRate$ is less than the current zoom factor, and then replace Equation 5 with Equation 6.

$$P_{n_j} - P_{n_i} = 0 \qquad (6)$$

We use the following algorithm to calculate $CRate$s for all the edges.

Procedure 1 Calculate *CRate*s

```
Input:  The dependency Graph, $G(V,E)$
        The type and original length for each element in $E$
Output: $CRate$ for each elastic edge in $E$

 1: Add the variable $Factor_{ZoomIn}$ to the linear system of $G$, thus the
    $Factor_{ZoomIn}$ in the equation $P_{n_j} - P_{n_i} = L_{original} \times Factor_{ZoomIn}$
    is no longer a known value.
 2: Set $VectorEdges$ to be the container containing all elastic edges
    whose original length is non-zero.
 3: Set $LastFactor_{ZoomIn}$ to be 1.
 4: repeat
 5:   for each edge $e_{(n_i,n_j)}$ in $VectorEdges$ do
 6:     Add an equation $P_{n_j} - P_{n_i} = 0$ to the linear system. Solve the
        system.
 7:     if the following three conditions are satisfied by the solution:
        a. The system has one or an infinite number of solutions;
        b. There is no edge in the solution whose length is negative;
        c. $Factor_{ZoomIn} > LastFactor_{ZoomIn}$ then
 8:       Assign $Factor_{ZoomIn}$ to the $CRate$ of $e_{(n_i,n_j)}$ and all other
          edges in $VectorEdges$ whose length is zero according to
          the solution. Remove these edges from $VectorEdges$. Re-
          move the equation just added. Assign $Factor_{ZoomIn}$ to
          $LastFactor_{ZoomIn}$. break.
 9:     else
10:       Remove the equation just added.
11:     end if
12:   end for
13: until no edges are removed from the last iteration
14: Assign $Positive\ Infinity$ to $CRate$ of all the edges in $VectorEdges$.
```

The time complexity of this algorithm is $O(n^2) * C_{es}$, where n is the number of the foci in the treemap, and C_{es} is the time complexity of solving the linear equation system. Because n is not related to the number of the treemap items, the scalability of algorithm would not be directly affected by the size of the dataset.

4.5 Implementation

We have implemented our algorithm in a treemap visualization system along with a query interface that can automatically or manually select foci. For the linear system, we implemented a stable variation of the Gaussian elimination algorithm to solve the matrix form of the linear system, whose complexity is $O(n^3)$. Thus the algorithm calculating the *CRate*s is $O(n^5)$, and the complexity of generating the transformed treemap for a specified zoom factor is $O(n^3)$. In fact, because the matrix form of our system is very sparse, floating-point multiplications needed for solving these sparse matrices are much smaller than dense matrices. Although we have not yet specifically optimized the performance, our implemented algorithm, on a 1.7GHz laptop, can achieve smooth treemap transformation interactively for the multi-level treemap with thousands of leaf items and up to a hundred foci, as the NBA dataset used in our case study and user study. The initial calculation of the *CRate*s takes a few seconds, and no delay can be noticed after the user starts adjusting the focus zoom factor. We believe there is still room for further performance improvement.

5 Case Study

The treemap used in this case study is created based on the four consecutive years of NBA data from the 2001-2002 season to the 2004-2005 season [1]. The hierarchical structure of the data is constructed by years, conferences, divisions, teams, and individual players in a top-down order. We use "Minutes/Game" as the size attribute to create the treemaps. This attribute is chosen because the "Minutes/Game" statistics reflect how important a player is to his team. The layout algorithm used to create the treemap is the squarified algorithm. By using this algorithm, items that have the largest sizes among their siblings are placed close to the upper left corners. Thus the importance of a player to his team can also be inferred by a player's relative position in his team on the treemap.

[1] http://www.usatoday.com/sports/basketball/nba/statistics/archive.htm

Each treemap item represents a player. We encode four attributes by colors in the four sub-regions of each item. The attributes are "Points/Game" (upper left), "Assists/Game" (upper right), "Rebounds/Game" (lower left), and "Fouls/Game" (lower right). For the first three attributes, the color changes from green to black to blue continuously to represent the highest to the lowest value. In other words, for these attributes, green is better and blue is worse. For "Fouls/Game", the color varies from red to black to white. Red means worse (more fouls), and white means better (fewer fouls).

The case study is based on the team Houston Rockets. Figure 6(a) is the original treemap that contains the foci generated by a query. The query is to select four year records of the players who played for the Rockets in the 2002-2003 season.

By displaying the foci in the original treemap, we can see the distribution information from the treemap context. For example, after the 2002-2003 season, half of the players left the Rockets, among which four players played for other teams in the 2003-2004 season. Two of them were quite important for their new teams. The treemap context helps the viewers quickly grasp those information. If the query result is displayed by other visualization techniques or a treemap which contains the foci only, some of the information here will not be so easy to see from the new view.

But because the sizes of the foci are not large enough, the players' names cannot be seen. It is difficult for us to know who left the NBA completely, and to which teams the players transferred. We also do not know whether they became better players for the new teams because the colors of the foci cannot be clearly seen. To address this issue, we enlarge the foci so that both the names and colors of the foci become quite easy to see.

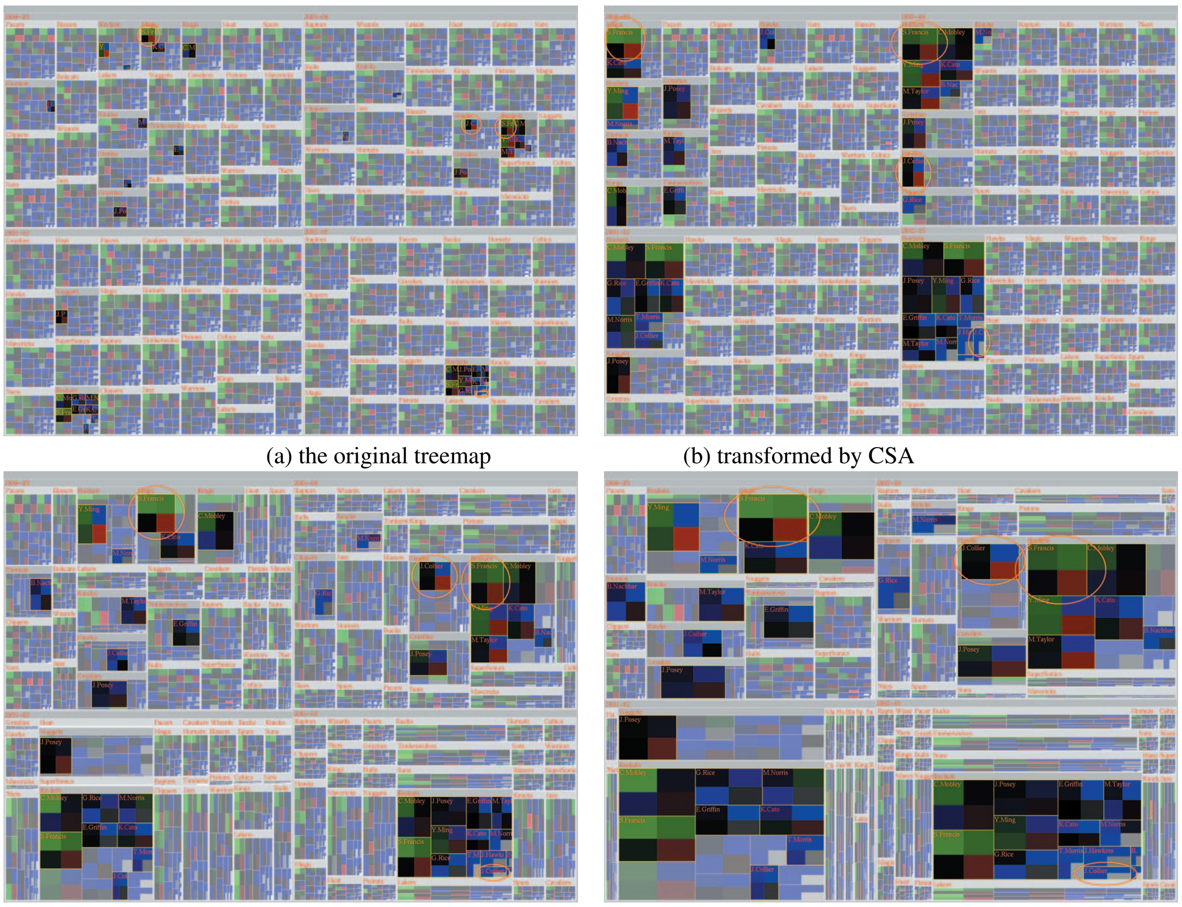
The treemap in Figure 6(b) is produced by the method which enlarges foci by changing the underlying values of the size attribute. We refer to this method as CSA hereafter. With the enlarged foci, we can clearly compare the performance change of transferred players. For example, "Points/Game" of S. Francis (Steve Francis) became higher; both "Minutes/Game" and "Points/Game" of J. Collier (Jason Collier) became much higher. However, from (a) to (b), the treemap undergos abrupt layout changes - all foci are clustered to the upper left corners of their parents. We could no longer estimate the importance of a player to its team by the item's relative position to its parent, due to the context loss regarding neighboring relations between items. And because it is hard to map items from (b) to (a), the knowledge gained from (a) could not be easily used on (b).

Balloon Focus, generating Figure 6(c) and (d), fixes the problem of CSA. By maintaining the positional dependency of the treemap items, the treemap context is well preserved. It allows the users quickly adapt themselves to the new treemap view and migrate the already acquired knowledge from the original treemap to the new treemap. From the relative positions of the items to their parents, we can see that J. Collier (Jason Collier) became more important to the Hawks in the 2003-2004 season than to the Rockets in the 2002-2003 season. S.Francis (Steve Francis) was still the NO. 1 "Minutes/Game" player for his team after he joined the Magic in the 2004-2005 season.

6 User Study

We conducted a user study based on the NBA statistics data set described in Section 5. The treemap configuration, i.e. size attribute, color attributes, and color encoding, was the same as in the case study. This study consisted of three sections, which compared Balloon Focus (BF) with no foci enlargement, with single-focus enlargement, and with CSA introduced in Section 5. Each of the methods in our study provided a certain degree of contextual information related to the data. We are interested in the difference in user performance and preference between the methods.

Participants Twelve graduate students participated in the study. 75% majored in computer science and engineering with various research focuses; 25% were from other departments. 42% were female



(a) the original treemap (b) transformed by CSA

(c) transformed by BF (medium zoom factor) (d) transformed by BF (maximum zoom factor)

Fig. 6. Treemaps created from the NBA statistics from the 2001-2002 to the 2004-2005 seasons. (a) is the original treemap where the foci are the multi-year records of the players who played for the Rockets in the 2002-2003 season. The foci are highlighted with yellow frames while the color of the non-focus items are muted. The treemap in (b) is produced by increasing the value of focus items' size attribute and regenerating the treemap; we refer to this method as CSA. In (c) and (d), the treemaps is transformed from (a) by Balloon Focus. Mapping the items from (a) to (c) or (d) is much easier than from (a) to (b) because Balloon Focus preserves the items' relative positions, while CSA does not.

and 58% were male. 25% were familiar with the NBA teams and players; 75% knew a little or had no knowledge. 83% did not know about treemaps before.

Procedure Before the test, we gave the subjects a tutorial, which was to provide them with the basic knowledge about the treemaps, the color encoding scheme, and how to use our treemap system to query and adjust the focus zoom factor. We allowed the subjects to get familiar with the interface with trial tasks. When they were performing the tasks, the time spent on each task were recorded. After the subject finished all tasks, a survey about the user experience was taken. In the study, the order of methods in each section was counterbalanced across subjects.

For each task, the subjects clicked a specific button, then the foci were selected automatically and color-highlighted in the treemap, as shown in Figure 6(a). Although the foci selected for each task were the results from a real query, we hard-coded the selection of foci, instead of asking the users to perform the query with the interface. This was to save time and, more importantly, guarantee that the difficulty of the tasks would not vary among the subjects. To adjust the focus zoom factor, the subjects dragged a slider bar back and force.

6.1 Compare BF with No Foci Enlargement

In this section, we studied whether foci enlargement helped the subjects answer questions related to the values of the item attributes. Four tasks were performed for each method. In each task, the treemap contained four foci, which represented the records of a particular player over the four years. The question was to find the foci that has the highest or lowest value of the specified attribute. We made sure that the size of each foci in the original treemap has at least eight pixels in each dimension to guarantee a moderate visibility even without enlargement.

We analyzed the experiment results by running a single factor Analysis of Variance (ANOVA) for the dependent variables. The same method was also used for the other sections. The measured time was not found to have a significant difference ($F(1, 22) = 0.186, p = 0.669$). The average time spent on a task was 29.2 seconds for BF, and 30.3 seconds for no foci enlargement. However, the error rate had a significant difference ($F(1, 22) = 28.3, p < .0001$), where the BF error rate (16%) was much smaller than No Foci Enlargement (50%).

The survey results show that all subjects preferred to enlarge the foci and then answer questions, even though they were able to see the unenlarged foci and the colors. The reasons provided by the subjects for such preference are: it was more efficient to find the answers with larger foci; it was felt much more confident with the answers when the foci were above a certain size. The average rate of usefulness for foci enlargement was 8.5 out of 10 (standard deviation = 1.67).

6.2 Compare BF with Single-Focus Enlargement

The purpose of this section is to verify our hypothesis that single-focus enlargement is insufficient for the treemap users when there are multiple foci selected. The task was to count the numbers of foci in different colors. We gave the subjects a color map which directly showed the mapping from colors to three classes. The users can enlarge the foci if so desired. With the single-focus method, only a single focus can be enlarged at one time.

Both the measured time and error rate from the two methods had a significant difference. The measured time (F(1, 22) = 10.09, $p < .005$) was 42 seconds for BF and 86 seconds for the other. The error rate (F(1, 22) = 9.08, $p < .01$) was 1.25% and 13.5% for BF and the other. The survey results show that compared with single-focus enlargement, all subjects preferred multi-focus. The major reason was that it took much longer to enlarge the foci one-by-one.

6.3 Compare BF with CSA

In the third section of our user study, we tested the importance of layout stability and visual consistency to the users when multiple foci were enlarged in context.

The task was to locate three target foci after the foci were enlarged. Before the task started, the users picked three target foci of their choice from all the foci displayed in the original treemap. Each target was assigned a number. After the subject found the targets in the new treemap, they tagged the targets with the numbers they originally assigned. The user must make sure that the correct numbers are tagged.

The average time each user spent on a task was 19.9 seconds with BF and 41.5 seconds with CSA, and the difference was significant (F(1, 22) = 27.7, $p < 0.0001$). When performing the tasks, with BF, the subjects found the targets in the new treemap by tracking; with CSA, however, the users had to scan the foci and match the labels. This observation explained the difference in their performance.

In the survey, 10 out of 12 preferred BF; 2 out of 12 were neutral; none preferred CSA. The benefits of the two methods that the users mentioned were as follows. With BF, it was easy and efficient to track the selected foci and it was comfortable to see the treemap and the foci change smoothly. CSA made better use of the space by avoiding thin and long items, and can allow a higher focus zoom factor. The average rate of the importance of smooth layout change is 9 out of 10 (standard deviation = 0.79).

In summary, we compared BF with the other three methods which also provide some degrees of context. The user study results show that enlarging foci in a treemap is very useful, single-focus enlargement is inefficient, and the layout stability of the treemap when enlarging the foci was highly valued by the users. Most users preferred using Balloon Focus to highlight and explore the query results.

7 Conclusions and Future Work

In this paper, we present a seamless multi-focus+context technique for treemaps, called Balloon Focus, that smoothly enlarges multiple focus items while maintaining a stable treemap layout as the context. In our algorithm, first we define a positional dependency among the treemap items, and use a DAG (Directed Acyclic Graph) to model the dependency in a whole treemap. Based on the dependency graph, an elastic model is employed to govern the placement of the focus items while maintaining the dependency, then the non-focus items are placed according to their dependency to the nearby foci.

The user study results showed that enlarging foci in a treemap was very useful in reducing the error rate, multi-focus enlargement was much more efficient than single-focus, and the layout stability of the treemap when enlarging the foci could greatly reduce the time spent on tracking and mapping items. Most users liked to use Balloon Focus to explore the query results. They highly valued multi-focus enlargement and the layout stability.

There are several directions for the future research. First, we will study the effects of using different parameters in the elastic model, for example, with different elastic modulus values for the elastic edges, and consider the users' preference in those effects. Second, we will attempt to evaluate whether it is possible or necessary to relax some of the positional dependency constraints so as to achieve a better zoom-in factor while still preserving the desired features on the treemaps. Third, we will explore the constraint-based layouts for other visualization methods, such as node-and-link-style tree and graph representations.

Acknowledgements

This work is supported by NSF ITR Grant ACI-0325934, NSF RI Grant CNS-0403342, NSF Career Award CCF-0346883, and DOE SciDAC grant DE-FC02-06ER25779.

References

[1] Treemap 4.1, human-computer interaction lab. http://www.cs.umd.edu/hcil/treemap.

[2] B. B. Bederson. Photomesa: a zoomable image browser using quantum treemaps and bubblemaps. In *UIST '01*. ACM.

[3] B. B. Bederson, B. Shneiderman, and M. Wattenberg. Ordered and quantum treemaps: Making effective use of 2d space to display hierarchies. *ACM Trans. Graph.*, 21(4):833–854, 2002.

[4] M. S. T. Carpendale, D. J. Cowperthwaite, and F. D. Fracchia. Multi-scale viewing. In *SIGGRAPH '96*.

[5] M. S. T. Carpendale, D. J. Cowperthwaite, F. D. Fracchia, and T. C. Shermer. Graph folding: Extending detail and context viewing into a tool for subgraph comparisons. In *GD '95*.

[6] A. Cockburn, A. Karlson, and B. B. Bederson. A review of focus and context interfaces. Technical report, University of Maryland, 2006.

[7] J.-D. Fekete and C. Plaisant. Interactive information visualization of a million items. In *INFOVIS '02*. IEEE Computer Society.

[8] G. W. Furnas. Generalized fisheye views. *SIGCHI Bull.*, 17:16–23, 1986.

[9] E. Gansner, Y. Koren, and S. North. Topological fisheye views for visualizing large graphs. In *INFOVIS '04*. IEEE Computer Society.

[10] M. C. Hao, U. Dayal, D. A. Keim, and T. Schreck. Importance-driven visualization layouts for large time series data. In *INFOVIS '05*.

[11] T. A. Keahey. The generalized detail-in-context problem. In *INFOVIS '98*.

[12] T. A. Keahey. Getting along: Composition of visualization paradigms. In *INFOVIS '01*.

[13] T. A. Keahey and E. L. Robertson. Techniques for non-linear magnification transformations. In *INFOVIS '96*.

[14] R. Kincaid and H. Lam. Line graph explorer: scalable display of line graphs using focus+context. In *AVI '06*. ACM.

[15] J. Lamping, R. Rao, and P. Pirolli. A focus+context technique based on hyperbolic geometry for visualizing large hierarchies. In *CHI '95*. ACM.

[16] Y. K. Leung and M. D. Apperley. A review and taxonomy of distortion-oriented presentation techniques. *ACM TOCHI*, 1:126–160, 1994.

[17] Miscosoft. Pre-rendered tree map views of all usenet and microsoft.public. http://netscan.research.microsoft.com/treemap/.

[18] T. Munzner, F. Guimbretière, S. Tasiran, L. Zhang, and Y. Zhou. Treejuxtaposer: scalable tree comparison using focus+context with guaranteed visibility. In *SIGGRAPH '03*. ACM.

[19] D. Nekrasovski, A. Bodnar, J. McGrenere, F. Guimbretière, and T. Munzner. An evaluation of pan & zoom and rubber sheet navigation with and without an overview. In *CHI '06*.

[20] M. Sarkar and M. H. Brown. Graphical fisheye views of graphs. In *CHI '92*. ACM.

[21] M. Sarkar, S. S. Snibbe, O. J. Tversky, and S. P. Reiss. Stretching the rubber sheet: a metaphor for viewing large layouts on small screens. In *UIST '93*. ACM.

[22] D. Schaffer, Z. Zuo, S. Greenberg, L. Bartram, J. Dill, S. Dubs, and M. Roseman. Navigating hierarchically clustered networks through fisheye and full-zoom methods. *ACM TOCHI*, 3(2):162–188, 1996.

[23] K. Shi, P. Irani, and B. Li. An evaluation of content browsing techniques for hierarchical space-filling visualizations. In *INFOVIS '05*.

[24] M. Spenke, C. Beilken, and T. Berlage. Focus: the interactive table for product comparison and selection. In *UIST '96*. ACM.

[25] J. Stasko and E. Zhang. Focus+context display and navigation techniques for enhancing radial, space-filling hierarchy visualizations. In *INFOVIS '00*, Washington, DC, USA. IEEE Computer Society.

[26] M. Toyoda and E. Shibayama. Hyper mochi sheet: a predictive focusing interface for navigating and editing nested networks through a multi-focus distortion-oriented view. In *CHI '99*. ACM.

[27] D. Turo. Hierarchical visualization with treemaps: making sense of pro basketball data. In *CHI '94*.

[28] J. Yang, M. O. Ward, and E. A. Rundensteiner. Interring: An interactive tool for visually navigating and manipulating hierarchical structures. In *INFOVIS '02*. IEEE Computer Society.