Progressive Presentation of Large Hierarchies Using Treemaps

René Rosenbaum* and Bernd Hamann

Institute of Data Analysis and Visualization (IDAV)
Department of Computer Science
University of California, Davis, CA 95616, U.S.A.

Abstract. The presentation of large hierarchies is still an open research question. Especially, the time-consuming calculation of the visualization and the cluttered display lead to serious usability issues on the viewer side. Existing solutions mainly address appropriate visual representation and usually neglect considering system resources.
We propose a holistic approach for the presentation of large hierarchies using *treemaps* and *progressive refinement*. The key feature of the approach is the mature use of multiple incremental previews of the data. These previews are well-designed and lead to *reduced visual clutter* and a causal flow in terms of a *tour-through-the-hierarchy*. The inherent scalability of the data thereby allows for a reduction in the consumed resources and *short response times*. These characteristics are substantiated by the results we achieved from a first implementation. Due to its many beneficial properties, we conclude that there is much potential for the use of progressive refinement in visualization.

1 Introduction

The trend for creating and collecting massive amounts of data has led to a strong demand for its meaningful analysis. With the prime objective being to visually convey information inherent in the data, interactive visualization has always been an integral component to accomplish this task. In recent years, many valid approaches for the display of hierarchical datasets have been developed.

One of the most prevalent tree visualization techniques is the *treemap display* [1]. It is easy to understand [2], has been applied to a million items [3], and gained increasing acceptance in research and industry [4]. However, when it comes to the display of large datasets, it shares a common drawback with many other visualizations – details of the data are difficult to comprehend. Furthermore, usually significant resources for data processing, transmission, and display are required.

We show how to improve the treemap information display by *progressive data handling* and *presentation*. Instead of displaying all data at once, our approach takes advantage of the inherent scalability of the data to provide previews

* The author gratefully acknowledges the support of Deutsche Forschungsgemeinschaft (DFG) for funding this research (#RO3755/1-1).

which initially require little data and are refined over time. This general benefit of progression is used (1) to provide a pre-defined or interactive *tour-through-the-hierarchy* and (2) to efficiently use consumed system resources. Thereby, the approach combines the mandatory stages compression, transmission, and demand-driven display into a single holistic strategy and nicely blends with most of the enhancements proposed for treemaps. Thus, it is a meaningful extension of many existing treemap implementations.

After introducing related work for treemap displays (Section 2) and the key properties of progressive information presentation (Section 3), we show how both approaches can be combined into a single presentation system (Section 4). While Section 4.1 is concerned with the technical implementation of progressive treemaps, Section 4.2 is dedicated to show how this can lead to an advanced information presentation. Section 5 state the achieved results regarding semantical as well as technical aspects. Conclusions and directions for future research are presented in Section 6.

Fig. 1. Example of a *progressive treemap* providing a tour-through-the-hierarchy. Its scalability property significantly decreases resource consumption whenever it is suitable to reduce the number of displayed nodes.

2 The treemap display and related work

Visualization of hierarchical data has been extensively researched in the past leading to numerous approaches. The *treemap display* belongs to the class of visualizations implicitly communicating parent-child relationships. This is achieved by representing the tree as recursively nested rectangular cells (see Figure 1). This containment approach illustrates the structure of the tree and clearly identifies parent and child nodes belonging to the same subtree. Size, color and other graphical cell attributes thereby can be varied in order to represent nominal or ordinal data properties.

The advantages of the treemap approach are obvious. It fully uses the available space for data display and applies intuitive means to provide information to the topology of the hierarchy. It has been demonstrated to perform well for providing overviews. Furthermore, when combined with an appropriate color scheme, treemaps make it immediately obvious which elements are significant and might require more attention. Thus, it can help to spot outliers and quickly find patterns. However, there are also drawbacks that arise as the amount of data increases. This can result in several thousand cells, many of which are so small in size that they become indiscernible. In this case the treemap loses the ability to efficiently convey information. Especially in deep hierarchies it is difficult to determine size or containments and thus to see relationships between nodes.

To overcome these disadvantages, the treemap display has been steadily improved. Thereby, existing publications focus on layout and appearance. The traditional slice-and-dice approach [1] has been enhanced by squarified [5], ordered [5], Voronoi [6], or generalized layouts [4]. There are also radial layouts [7] and different three-dimensional approaches [8]. Cascaded treemaps [9] have been developed to emphasize containment within tree structure, but at the cost of space. There are strategies to emphasize important nodes by applying spatial distortion [10]. However, even when these enhancements are applied, the available screen space is usually too small to allow for the legible display of large trees. One possible solution for this problem is to apply suitable interaction [11]. Browsing large hierarchies, however, is still a demanding task usually coupled with many single browsing steps and loss of context.

Interestingly, only few research efforts have been done in enhancing the underlying data handling required to process large hierarchies. Besides the fact that layout calculation is computationally expensive, all data must be available in advance. This makes the visualization difficult to use for large data volumes and environments requiring data transmission. Similar problems in image communication have been overcome by the development and suitable applications of *progressive refinement* [12]. By taking advantage of encoding schemes that convert the image into a multiscale representation, the data can be sent, processed, and displayed piecewise. Using this approach it is possible that the consumed resources do no longer depend on the original data volume, but only on the required parts. Thereby, the used concepts of Regions of Interest (RoI) and Levels of Detail (LoD) strongly correspond to the Degree of Interest (DoI) approach well-known and successfully applied in computer graphics and visualization.

3 Progressive information presentation

Progressive refinement allows for a novel kind of information display that can basically be seen as a predefined or interactive animation of the data tightly coupled with a highly resource-saving processing and transmission system. Besides overcoming bandwidth-limitations for raster imagery, it has also been applied to other kinds of data [13] and system constraints [14]. The main features and beneficial properties can be summarized as follows:

First conclusions with much less data. Progressive contents are organized data sequences whereby the reconstruction of a truncated sequence leads to an abstraction of the content with less detail. Thus, content previews can be provided at any point. With little data available, first conclusions can already be drawn.

Efficient use of system resources. Progressive contents are compressed and stored in a modular hierarchical structure allowing for flexible and highly efficient access and delivery of different abstraction levels. This implements the paradigm - *Compress Once: Decompress Many Ways* [15, p. 410] and leads to the advantage that just the data required on the viewer's side is handled.

Additional properties of the data become visible. The different data views produced during refinement may allow for conveyance of additional properties and characteristics during presentation, which leads to a deeper understanding of the data. Due to the fact that a progressive presentation is usually designed to show important data in early stages, they stand out in the representation. RoI, LoD or similar concepts like Geometry of Interest (GoI) may be applied to describe this procedure formally [12, 16].

Progression is data-seeing. Progressive refinement provides a preview sequence. As each subsequent view adds detail to the display, an incremental buildup of knowledge about the data can be achieved (see Figure 2). Such a *tour-through-the-data* supports well-accepted visualization principles as the information-seeking mantra [17] – Overview first, zoom and filter, then details-on-demand.

To apply progressive refinement, the classic visualization pipeline must be enhanced by a few basic principles [16]. Thereby, a client-server environment is typically assumed to use its full potential. On the *server side*, the data is preprocessed and stored for multiple use. The preprocessing stage basically transfers the data in a *hierarchical data structure* that is further compressed for permanent storage and resource-efficient transmission. During a viewing request, this data structure is accessed and flexibly *traversed* in order to create meaningful previews and sequentially transmitted to the viewing device. The traversal can be predefined by a presentation author and interactively modified dependent on current interests. By using modular compression schemes it is ensured that the contents can be delivered without requiring further processing. The *client* serves only as *viewing and interaction* component. For progressive presentation, the different previews are successively extracted, decoded, and, if necessary, post-processed before display.

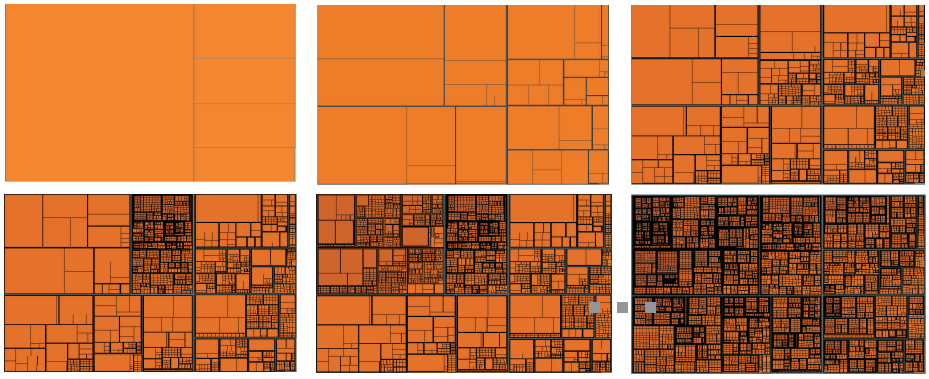

Fig. 2. A *progressive treemap* designed to convey tree topology and structural characteristics of important nodes applying simple breadth-first (top) and authoring-based (bottom) traversal.

4 Progressive treemaps

Progressive refinement provides many advantages for data visualization. We shown how to implement and take advantage of this approach for the treemaps display in order to reduce consumed resources (Section 4.1) and to improve the conveyance of information about the topology of the hierarchy (Section 4.2). The basic strategy is similar to the general progression procedure – instead of presenting all data at once, the different nested sets of hierarchy nodes are refined top-down, whereby certain nodes and belonging subtrees can be prioritized before or during interactive exploration.

4.1 Progressive refinement for treemaps

A basic requirement for progressive refinement is the conversion of the original data into a hierarchical structure. Due to the fact that the result of the treemap layout calculation is already a hierarchy of rectangles, however, this stage can be skipped for progressive treemaps.

To encode the rectangles appropriately, an approach similar to *delta coding* [18] taking advantage of the nesting property of the rectangles is applied. As the respective positions can be stated relative to the parent cell, a much smaller value range is required. This in turn can be used to reduce the number of bits needed to encode position. As all the values of the hierarchy are independently accessible, the approach fully supports *random access* within the encoded data and does not limit inherent scalability. It is also possible to increase compression performance by further encoding single values.

Due to the modular structure of the encoded data, nodes and subtrees can be almost arbitrarily assigned to the different previews. During *traversal and sequencing* of the hierarchy, it is only necessary to ensure that prior to including a

node all of its ancestor nodes have already been part of the sequence. This ensures that screen position can be decoded and restored properly. Due to the common overview-than-detail refinement, this, however, is usually not a limitation.

To describe prioritization between nodes and the corresponding subtrees, we vary the commonly used term Region of Interest to *Node of Interest* (NoI) to take into account the terminology used for hierarchical data. The traditional term *Levels of Detail* nicely corresponds to the different tree levels and is therefore kept. A single NoI is specified by a unique index, its LoD by the desired amount of subtree levels. Based on these specifications, appropriate traversal orders and previews can be developed by an author or interactively modified by the viewer.

Once created and traversed, *decoding and viewing* the progressive contents on the viewer side is straightforward. Based on a low complex protocol indicating parent-child relationships of incoming geometry, the data hierarchy is piecewise reconstructed and the respective data values decoded. Preview switches indicate the belonging of data to single previews. When a switch is signalled, all currently available data is released for display. As the treemap layout has been computed without prior knowledge to the respective screen properties, a *postprocessing* step scales the geometry to the screen dimensions before rendering.

4.2 Progressive information presentation

Appropriate information presentation by meaningful preview sequences depends strongly on the traversal of the data. Due to the variety of possible presentation goals and user preferences, however, strategies and options for valid hierarchy traversal are manyfold.

The simplest refinement strategy is a top-down breadth-first traversal. Even when straightforward it can still reveal important knowledge about the data, like the number of hierarchy levels or variations in the depth of subtrees. However, often not all nodes are of the same importance. In such cases it is appropriate to introduce prioritization or to limit the shown LoD. To influence the corresponding traversal strategies, we propose to apply either predefined *authoring-based* or dynamic *interaction-based refinement* or a combination of both.

Authoring-based refinement. Besides keeping the parent to child traversal order, the author has no limitations when creating a presentation sequence. The author is free to select an appropriate number of previews and to *manually* specify an LoD for every node and preview. For simpler authoring, however, we propose to use a *semi-automatic* specification approach. It interpolates the LoD values of single nodes whenever there is no explicit declaration. This allows for quick authoring even when the hierarchy and the number of previews are large.

Keeping in mind our goal to convey structure, localization, and relation of nodes and subtrees to others, the example depicted in Figure 2 demonstrates the beneficial properties of a typical progressive treemap presentation. In order to provide an overview and to indicate structural dependencies and relations, the first previews show the first three levels of the node hierarchy only (Figure 2, top). It can be seen how data refines globally. In order to prioritize and display an

important node that resides at a much deeper hierarchy level, only its ancestor nodes are added to the next previews. This leads to a local refinement of the desired geometry as well as its corresponding context(Figure 2, bottom, left). Once the selected node is displayed at the desired LoD, the refinement may be finished. Our example, however, continues with the refinement of the next higher level relative to the prior node in order to provide further context to local relatives (Figure 2, bottom, middle). This prioritization and refinement of one or multiple items is continued until all data is displayed (Figure 2, bottom, right).

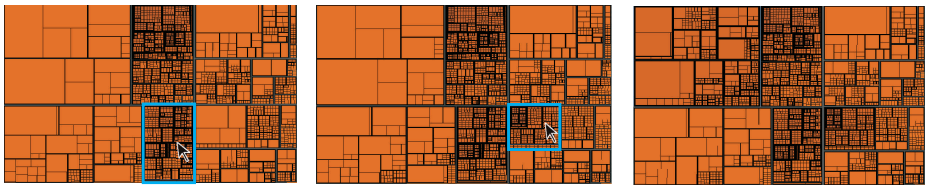

Fig. 3. The interaction-based refinement of a node postpones an authoring-based sequence until the desired LoD for the node is achieved (variation of the authoring-based presentation sequence shown in Figure 2).

Interaction-based refinement. In order to simplify the conveyance of local properties, we support interactive node selection and prioritization at any time. This can be easily implemented based on the introduced processing scheme. Once a node is selected, the traversal component reacts by postponing a possible authoring-based traversal and starts refining the desired NoI (see Figure 3, left). The node is refined until its specified LoD is achieved or the user moves to the next node (Figure 3, middle). It is also possible to assign same priority to multiple nodes leading to simultaneous node refinements. If all desired nodes are displayed in the respective LoD, the traversal continues in the prior refinement sequence (Figure 3, right). To avoid redundancies, it is always ensured that already displayed nodes are not handled twice. Typical control elements for animations as pause and forward buttons complete the suite of supported interaction means.

5 Results and general properties

Semantic aspects. By providing different well-designed incremental previews, progressive refinement allows for a novel kind of "animated" data presentation. It supports and fulfills many of the principles and requirements stated in the literature for successful data visualization and provides options to align the presentation to the respective goals. We show how answering different questions raised by the 2003 *InfoVis contest* concerning the display of hierarchical data can be significantly simplified by applying progressive refinement.

We found that a progressive treemap mainly supports answering questions about *topology*, without requiring the widely applied color coding. Typical overall characteristics of the hierarchy (e.g., *What is the deepest branch?*) are inherently communicated by breadth-first traversal, which is simple and does not require any external authoring. The same applies to questions that usually require interaction to be answered (e.g., *What are the properties of the first nodes?*). The provided means to suspend refinement, however, help to overcome possible time constraints. Questions related to single data items (e.g., *What is the path of this node?*) or to their local relatives (e.g., *What are the children or siblings of this node?*) can be easily answered by prioritization using authoring- or interaction-based refinement. In early presentation stages, local refinement is also a great means to emphasize the importance of nodes. Other problems, like appropriate labeling, may also be overcome by taking advantage of the uncluttered views in early progression stages, but have not been considered in our implementation.

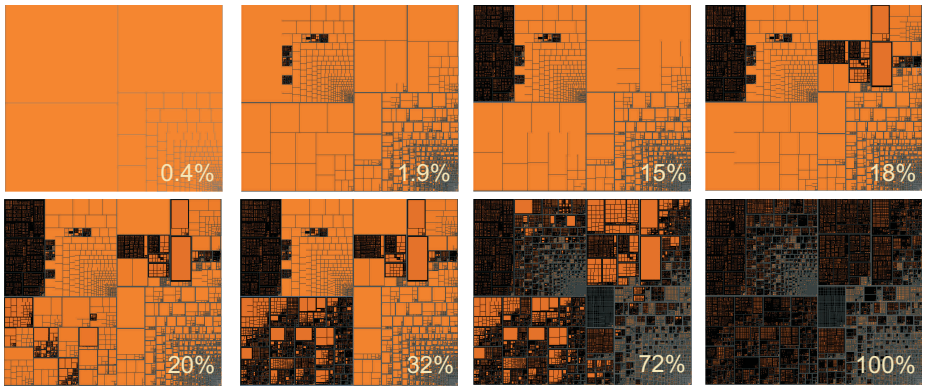

Fig. 4. A typical progressive presentation of a large hierarchy and the amount of data required to display the previews with respect to the whole volume. (Data set: *logs_A_03-01-01.xml*, part of the 2003 *InfoVis contest* concerning hierarchical data)

Technical aspects. Another major advantage of progressive refinement is its general resource-saving characteristic. It tightly combines the different processing stages to a single strategy and thus makes it rather useful for large data volumes. The modular and scalable structure of the data allows for accurate delivery of those parts needed for a certain representation and purpose. It decouples the required resource consumption from the original amount of data. This allows for fast feedback even for large hierarchies as well as low bandwidth environments and is a significant advantage especially for strongly limited viewing devices. As illustrated in Figure 1 these devices can usually process and display only a fraction of all available data in an appropriate manner. In this example just 1.03% (left) and 18.08% (middle) of all data is needed to provide an appropriate and uncluttered overview of the hierarchy. The same applies when

the viewer has gained the desired knowledge and aborts the presentation before all detail is shown. Figure 4 shows relative data volumes required during an example presentation. It can be seen that early previews can be quickly provided and only little system resources are required. As data volumes and therefore processing times increase, the response times in early progression stages are much shorter. If interactivity is mandatory, just the amount of displayed nodes must be adapted. This can be accomplished by selecting relevant NoIs only or by limiting LoDs. Contrary to the traditional approach requiring and displaying all nodes, this is of great importance for many applications.

General properties. The introduced approach can be applied together with almost all of the different layouts proposed for the treemaps display. As many interactions can be specified by the NoI/LoD concept, it can also be combined with existing browsing techniques, e.g., [11]. Furthermore, the use of a single server-side "multi-purpose" data structure avoids the costly layout computation and adaptation when the presentation is to be shown multiple times and with different goals. This is of beneficial advantage for smart environments [19].

However, there is no approach without shortcomings. Probably the main limitation of progressive refinement is the fact that not all visualization goals can be supported. This applies especially for all goals avoiding or inverting the overview-then-detail principle. Furthermore, abstractions provided in previews may lead to wrong conclusions. As the proposed postprocessing stage changes the cell aspect ratio, the approach cannot be applied to squarified treemaps. To overcome this limitation, it must be ensured that layout and screen dimensions match accordingly.

6 Conclusions and directions for future research

Progressive treemaps can be used to overcome the different problems imposed by large hierarchies. Progressive refinement is able to provide multiple data previews and can reduce the consumed system resources significantly. It further allows for an authoring-based *tours-through-the-hierarchy* and interactive manipulation of the refinement sequence. The strategy can be combined with other existing treemap approaches and applied to hierarchies of different size and topology. It enhances the extraction of knowledge from the data and increases system performance.

Although the feedback we received concerning our first implementation is promising, future research should provide an empirical proof of the usefulness of progression by a user study. By taking advantage of the foundations laid in this research effort, it seams also meaningful to apply the approach to other visualizations, e.g., the balloon focus [10], in order to reduce resource consumption.

References

1. Shneiderman, B.: Tree visualization with tree-maps: 2-d space-filling approach. ACM Transactions on Graphics **11** (1992) 92–99
2. Johnson, B.: Treemaps: Visualizing Hierarchical and Categorical Data. PhD thesis, University of Maryland (1993)
3. Fekete, J.D., Plaisant, C.: Interactive information visualization of a million items. Information Visualization, IEEE Symposium on **0** (2002) 117
4. Vliegen, R., van der Linden, E.J.: Visualizing business data with generalized treemaps. IEEE Transactions on Visualization and Computer Graphics **12** (2006) 789–796
5. Bruls, M., Huizing, K., van Wijk, J.: Squarified treemaps. In: Proceedings of the Joint Eurographics and IEEE TCVG Symposium on Visualization. (1999) 33–42
6. Balzer, M., Deussen, O.: Voronoi treemaps. In: Proceedings of IEEE Symposium on Information Visualization. (2005) 49–56
7. O'Donnell, R., Dix, A., Ball, L.: Ball exploring the pietree for representing numerical hierarchical data. In: Proceedings of HCI2006, London, Springer (2006)
8. Bladh, T., Carr, D.A., Kljun, M.: The effect of animated transitions on user navigation in 3d tree-maps. In: Proceedings of the Ninth International Conference on Information Visualisation, Washington, DC, USA (2005) 297–305
9. Lü, H., Fogarty, J.: Cascaded treemaps: examining the visibility and stability of structure in treemaps. In: Graphics Interface. (2008) 259–266
10. Tu, Y., Shen, H.W.: Balloon focus: a seamless multi-focus+context method for treemaps. IEEE Transactions on Visualization and Computer Graphics **14** (2008) 1157–1164
11. Blanch, R., Éric Lecolinet: Browsing zoomable treemaps: Structure-aware multi-scale navigation techniques. IEEE Transactions on Visualization and Computer Graphics **13** (2007) 1248–1253
12. Rosenbaum, R., Schumann, H.: Progressive raster imagery beyond a means to overcome limited bandwidth. In: Proceedings of Electronic Imaging - Multimedia on Mobile Devices. (2009)
13. Lee, H., Desbrun, M., Schröder, P.: Progressive encoding of complex isosurfaces. In: Proceedings of ACM SIGGRAPH, New York, NY, USA, ACM Press (2003) 471–476
14. Pascucci, V., Laney, D.E., Frank, R.J., Scorzelli, G., Linsen, L., Hamann, B., Gygi, F.: Real-time monitoring of large scientific simulations. In: Proceedings of ACM Symposium on Applied Computing. (2003)
15. Taubman, D., Marcellin, M.: JPEG2000: Image compression fundamentals, standards and practice. Kluwer Academic Publishers, Boston (2001)
16. Rosenbaum, R., Schumann, H.: Progressive refinement - more than a means to overcome limited bandwidth. In: Proceedings of Electronic Imaging - Visualization and Data Analysis. (2009)
17. Shneiderman, B.: The eyes have it: A task by data type taxonomy for information visualizations. Proceedings of the IEEE Symposium on Visual Languages (1996) 336–343
18. Deering, M.: Geometry compression. In: Proceedings of ACM SIGGRAPH Conference on Computer graphics and interactive techniques, New York, NY, USA, ACM (1995) 13–20
19. Thiede, C., Schumann, H., Rosenbaum, R.: On-the-fly device adaptation using progressive contents. In: Proceedings of International Conference on Intelligent Interactive Assistance and Mobile Multimedia Computing. (2009)