Improvements of Space-Optimized Tree for Visualizing and Manipulating Very Large Hierarchies

Quang Vinh Nguyen and **Mao Lin Huang**
Faculty of Information Technology
University of Technology, Sydney, Australia
1 Broadway, NSW 2007, Australia
{quvnguye, maolin}@it.uts.edu.au

Abstract

This paper describes some improvements over the original *Space-Optimized Tree* technique for the visualization and manipulation of very large hierarchies. The new system uses an improved algorithm to calculate geometrical layouts and it also provides better navigation capability. We introduce our new layout algorithm that can make more consistence of the display than the original layout technique made. We also combine *DualView* (a new focus+context technique) with the current modified semantic zooming in order to interactively navigate through the large and very large hierarchies.

Keywords: information visualization, space-optimized, 2d, focus+context.

1 Introduction

Visualizing and manipulating large hierarchies have been more and more important in many fields. However, the visualization of very big data sets at a limit small screen resolution is always a big challenge. Fortunately, several proposals and implementations have attempted to address and overcome the problems. Graph and/or tree drawing techniques for visualizing large hierarchies noticeably are *Cone-Tree* [Robertson et al. 1991], *Hyperbolic-Browser* [Lamping and Rao. 1995], *Disk-Tree*[Chi et al. 1998], *Tree-Maps* [Johnson and Shneiderman. 1991], *Botanical Visualization* [Kleiberg et al. 2001], and *Spaced-Optimized Tree* [Nguyen and Huang. 2002].

Space-Optimized Tree (SOTree) is a newly simple and fast tree-layout technique in 2-dimensional space where the entirely tree-like structure is visualized efficiently at a limit displaying area. Similarly to *tree-Maps* [Johnson and Shneiderman. 1991], *SOTree* [Nguyen and Huang. 2002] uses area division to define the layout of sub-trees. This property attempts to utilize the available space to display more information. *SOTree*, however, uses node link diagrams to show the relationships of hierarchical data. This property improves dramatically the clarity of the data structure compared to tree-maps. Thanks to its simple and fast algorithm, *Space-Optimized Tree* is highly applicable to visualize and manipulate large and very large relational hierarchies. Figure 1 shows the *SOTree* layout technique on a very large data set. *SOTree* is an excellent technique for visualizing entirely large relational hierarchies. However, its angular division might lead to the unbalancing where nodes far from centre often obtain more space compared to nodes close to the centre. In order words, the density of information near the centre area is much higher compared to border area (see Figure 1).

We present new improvements in geometrical layout and navigation over the original *Space-Optimised Tree*. Our new layout technique improves the optimization of space and the balancing of the visualization while it just slightly increases the executing cost compared to the original algorithm. In order to make the navigation more interactive, we combine the new focus+context technique *(DualView)* with the original modified semantic zooming. The semantic zooming is responsible for scaling down size of the displaying information if the density is too high. We also provide *DualView* focus+context technique to interactively browse particular information when the density is not so high.Section 2 describes technical details of our improvements of layout and navigation from the original *SOTree*. Subsection 2.1 focuses on geometrical layout and subsection 2.2 is about our navigation. Section 3 presents our comparison of the improved SOTree and original SOTree. Finial section is our conclusion.

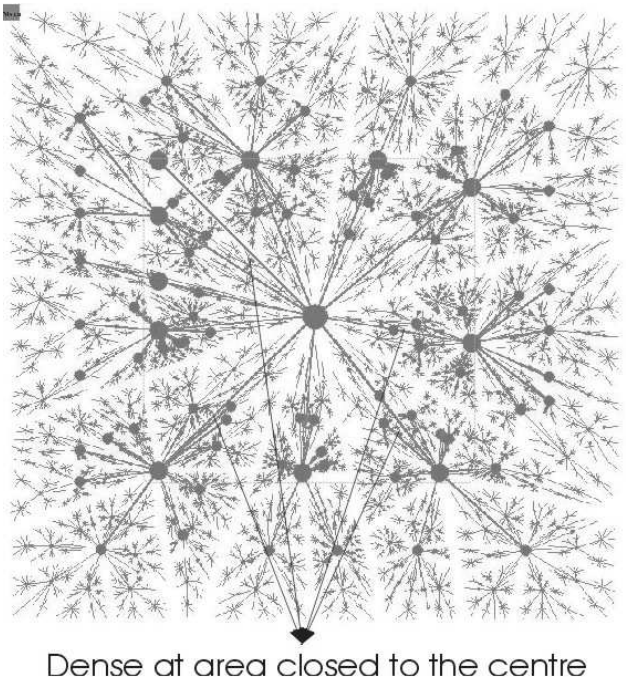


Figure 1. An example of SOTree with a very large data set of approximately 20 000 nodes.

2 Technical Detail

Similarly to the original *SOTree*, our new technique is only applicable to rooted trees *T*. Suppose that *v* is a vertex in *T*, then *T(v)* is the sub-tree rooted at *v* that is induced by all vertices on paths originating from *v*. The vertex *v* and its sub-tree *T(v)* are positioned inside a geometrical local region which is bounded by a polygon *P(v)*.

2.1 Geometrical Layout

The geometrical layout is responsible for defining position of vertexes or nodes of a given tree *T*. Each vertex *v* is bounded by a polygon *P(v)* where the location of v is calculated from *P(v)*. We assume that the root *r* of *T* is at the centre of the entirely rectangular displaying area, and this rectangle is also the geometrical local region of *r*. From the above property, we linearly calculate the all bounded polygons $P(v_1)$, $P(v_2)$, …, $P(v_n)$ and the locations of vertexes $v_1, v_2, \ldots, v_n$.

Suppose that a sub-tree $T(v_i)$ has *k* children where their vertexes are $\{v_l, v_{l+1}, \ldots, v_{l+k-1}\}$. The polygon $P(v_i)$ and vertex v_i are also already defined. We firstly calculate local regions $\{P(v_l), P(v_{l+1}), \ldots, P(v_{l+k-1})\}$ for the children $\{v_l, v_{l+1}, \ldots, v_{l+k-1}\}$ where $P(v_i) = P(v_l) \cup P(v_{l+1}), \ldots, \cup P(v_{l+k-1})$. We then calculate positions of vertexes $\{v_l, v_{l+1}, \ldots, v_{l+k-1}\}$ that are inside their local regions $\{P(v_l), P(v_{l+1}), \ldots, P(v_{l+k-1})\}$. The above calculation linearly repeats to all sub-trees from the vertexes $\{v_l, v_{l+1}, \ldots, v_{l+k-1}\}$ and so for until all leaves of the sub-tree are reached. We technically ignore the layout calculations for those sub-trees when their local regions are too small to be displayed at the current screen resolution.

The polygon *P(v)* of a vertex *v* is calculated depending on the weight *w(v)* of *v*. The value of *w(v)* is calculated from leaves to the root (i.e. postorder traversal) using the following formula:

- If *v* is a leaf (has no children), its weight is *w(v)* = 1
- If *v* has k children $\{v_l, v_{l+1}, \ldots, v_{l+k-1}\}$, its weight is

$$w(v) = 1 + C\sum_{j=0}^{k-1} w(v_{l+j}) \qquad (1)$$

Constant *C* is a scalar that determines the difference between the weight of a vertex and their children. In other words, the larger the *C*'s value is, the bigger of the difference of local regions between vertexes with more descendants and vertexes with fewer descendants is. We apply a constant $C = 0.45$ in our new prototype system.

Let *A(v)* is the area of *P(v)* at vertex *v*. Suppose that *v* has k children $\{v_l, v_{l+1}, \ldots, v_{l+k-1}\}$. The boundaries of the children $\{P(v_l), P(v_{l+1}), \ldots, P(v_{l+k-1})\}$ are respectively polygons that are formed by the intersection of segments through *v* and *P(v)*. $\{P(v_l), P(v_{l+1}), \ldots, P(v_{l+k-1})\}$ have areas respectively of $\{A(v_l), A(v_{l+1}), \ldots, A(v_{l+k-1})\}$. The division is dependent on the magnitudes of the above areas. The value of $A(v_{l+m})$ assigned to m^{th} child is calculated by the formula:

$$A(v_{l+m}) = A(v)\frac{w(v_{l+m})}{\sum_{j=0}^{k} w(v_{l+j})} \qquad (2)$$

Suppose that the vertex *v* and its local region *P(v)* have been defined. *P(v)* is a polygon with *n* vertexes $\{p_0, p_1, \ldots, p_n\}$ where p_0 is the location of the parent vertex of vertex *v*. The areas $\{A(v_l), A(v_{l+1}), \ldots, A(v_{l+k-1})\}$ of all local regions of the children $\{v_l, v_{l+1}, \ldots, v_{l+k-1}\}$ of vertex *v* are calculated from the equation (2).

From vertex *v*, we draw *n* lines to all vertexes $\{p_0, p_1, \ldots, p_n\}$ of the polygon *P(v)*. We then have *n* triangles $T(v,p_0,p_1)$, $T(v,p_1,p_2)$, …, $T(v,p_{n-1},p_0)$. The areas of these triangles are computed easily and are called $A(vp_0p_1)$, $A(vp_1p_2)$, …, $A(vp_{n-1}p_0)$ respectively. The local region $P(v_m)$ of m^{th} child v_m of vertex *v* is calculated as below:

- We reclusively compare each triangle's area from $A(v,p_0,p_1)$, $A(v,p_1,p_2)$, …, $A(v,p_{n-1},p_0)$ with $A(v_m)$ that is calculated by formula (2). If there is one $A(v,p_i,p_{i+1})$ that equals to $A(v_m)$, then we assign the corresponding polygon $P(v,p_i,p_{i+1})$ to vertex v_m as its local region.
- Otherwise we find one $A(v,p_i,p_{i+1})$ that has the value of area which is most close to $A(v_m)$. If $A(v,p_i,p_{i+1}) > A(v_m)$, then we have to find a point *K* on the side p_jp_{j+1} to reach $A(v,p_i,K) = A(v_m)$, and assign the corresponding polygon $P(v,p_i,K)$ to vertex v_m as its local region.
- If $A(v,p_i,p_{i+1}) < A(v_m)$, then we have to find a point *K* on the side p_{i+1},p_{i+2} of the next triangle $T(v,p_{i+1},p_{i+2})$ to reach $A(v,p_i,p_{i+1}) + A(v,p_{i+1},K) = A(v_m)$, and assign the corresponding polygon $P(v,p_i,p_{i+1},K)$ to vertex v_m as its local region.

A same procedure is applied to find local regions of all children. Figure 2 is an example of dividing v_i's local region into 4 sub-regions for its children. Figure 3 shows an example of area division of a tree.

Similarly to the original *SOTree* technique, the position of a vertex *v* is calculated from the boundary polygon *P(v)*. We firstly find a point *Q* in the polygon *P(v)* that the straight line connecting *Q* and the father *v'* of vertex *v* divides *P(v)* into two polygons of the same areas. The position of *v* is the midpoint of the segment of *Q* and *v'*. Figure 4 shows an example of finding a vertex from a given bounded polygon.

2.2 Navigation

We combine both *DualView* (a focus+context technique) and the semantic zooming for *SOTree's* navigation. The semantic zooming is applied for very high density visualization. While, the *DualView* interactively helps users browse the interested information.

We reuse the modified sematic zooming technique from Nguyen and Huang [2002] to enlarge the focused sub-tree. This interaction response to mouse-click event and the selected node moves toward the centre of displaying area. All ancestors and siblings are ignored at this

navigation. We, however, display directed ancestors at the history path for keeping track the hierarchy. Then, the sub-tree is expanded to entirely displaying area. In other words, the sub-tree layout is recalculated based on its new geometrical region (see figure 5).

Semantic zooming is an excellent technique for navigating very large hierarchies. However, the lack of context of the data structures might prevent users from viewing the other parts of the hierarchy within the global context. In order to overcome this problem, we combine the semantic zooming with *DualView*, a fast focus+context technique.

The *DualView* technique includes two transformations including *Browsing* and *Distortion*. We use *Browsing* transformation to bring interested information into the focus region while the fisheye-like *Distortion* is applied to increase the magnification of information at the focus area. The two transformations are applied independently onto both horizontal and vertical directions. In short, the functions of *Browsing* and *Distortion* Transformation are respectively:

$$T_{Browsing}(x_b) = Tangent(x_b) \quad (3)$$

$$\begin{cases} T_{Distortion} = b + \sqrt{c - (x_d - a)^2} & \text{for } x_d \geq 0 \\ T_{Distortion} = -b - \sqrt{c - (x_d + a)^2} & \text{for } x_d < 0 \end{cases} \quad (4)$$

Where a, b, c are constants.

During the focus+context navigation, the size of nodes moving outward the focus region, are decreasing gradually while nodes moving toward are increasing their size and displaying more information. Figure 6 shows the navigation for a very large tree in four steps: original tree layout, the layout after applying semantic zooming, the layout after applying *Browsing Transformation* and the layout after applying *Distortion Transformation*.

3 Statistical Comparison

Figure 7-11 are 4 typical experiments of our new improvement and the original SOTree layout techniques. The experiments use JavaApplet 1.4 and running on Pentium III 800 MHz.

4 Conclusion

We have presented our improvements over the original SOTree for visualizing large relational hierarchies. The comparison shows that our new layout technique is just slightly slower, but it improves a lot in geometrical layout of the tree hierarchies. In our new algorithm, the division of areas is much consistence and the display of nodes is more balanced (the display area is divided more equally to every node). In other words, the new layout reaches a better space-optimization. Furthermore, the viewing technique has been improved by the use of our new combination of DualView and semantic zooming techniques. We believe that we have added some values into the enhancement of the original SOTree technique.

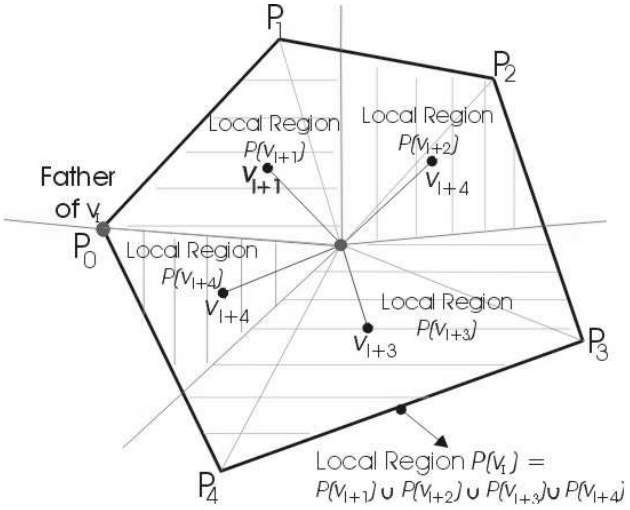


Figure 2. An example of dividing v_i's local region into 4 sub-regions for its children.

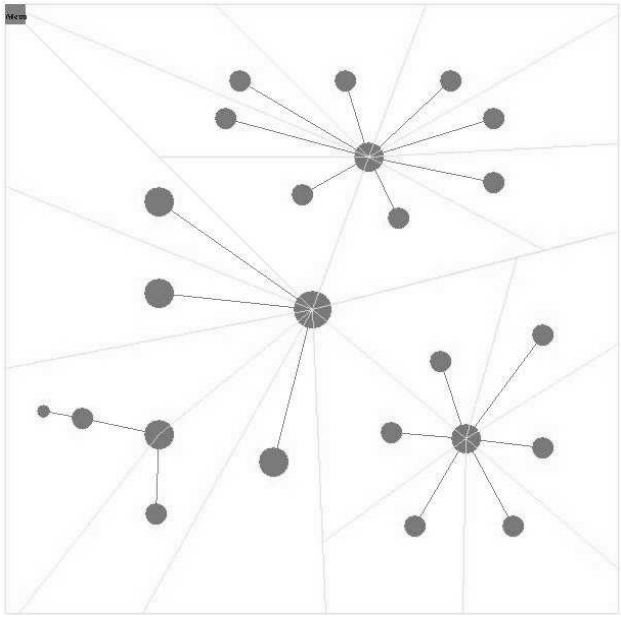


Figure 3. An example of area division of a small tree

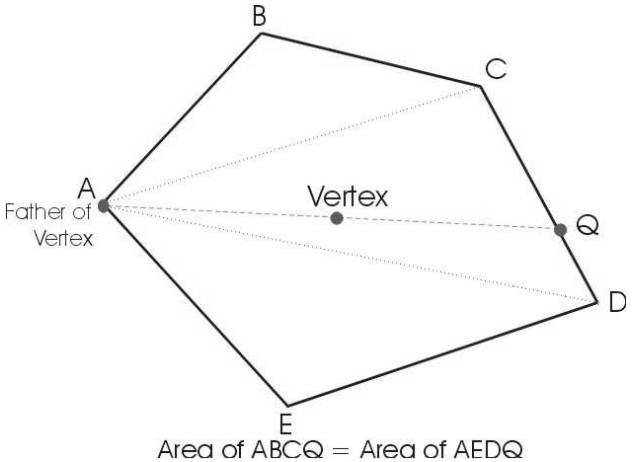


Figure 4. An example of finding a vertex from a given boundary polygon.

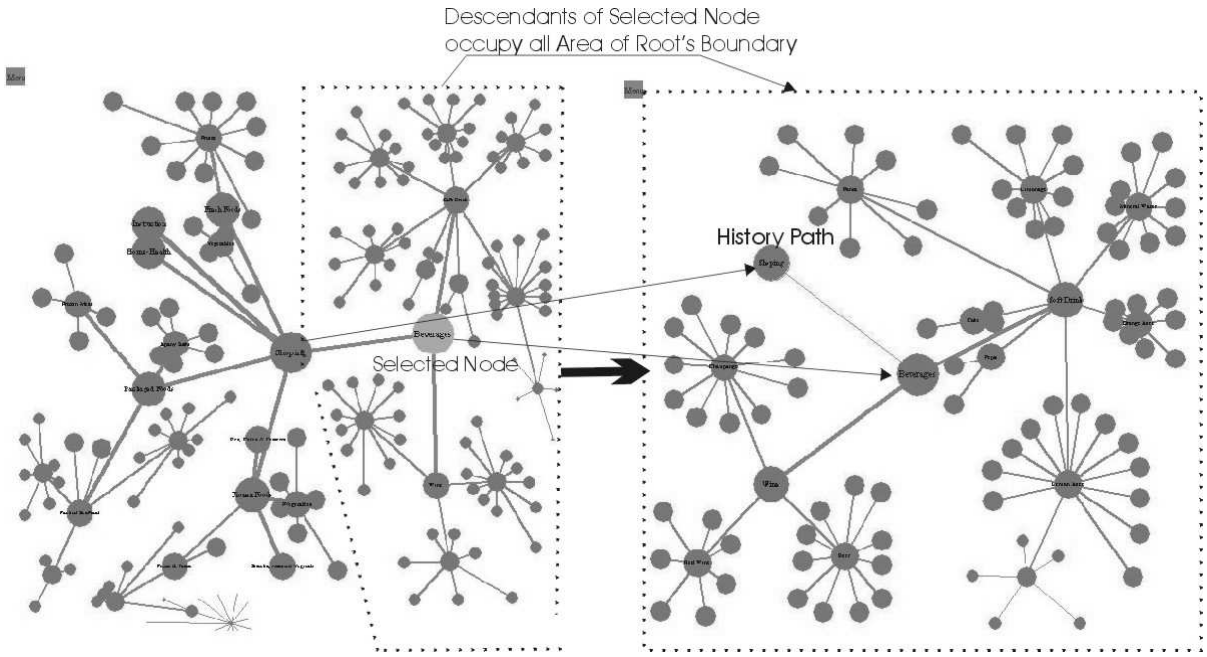
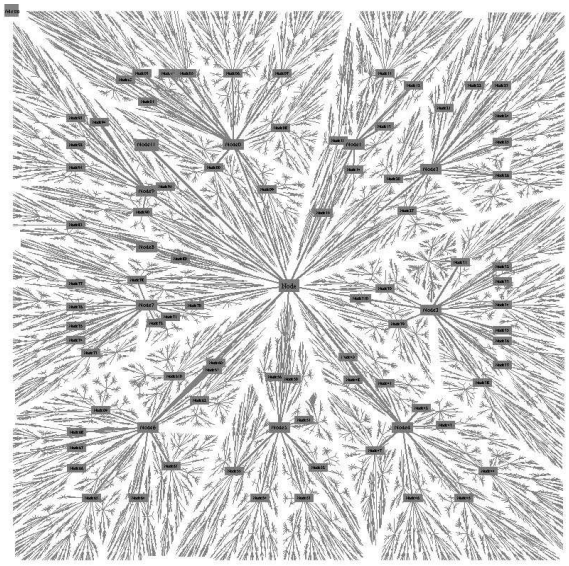
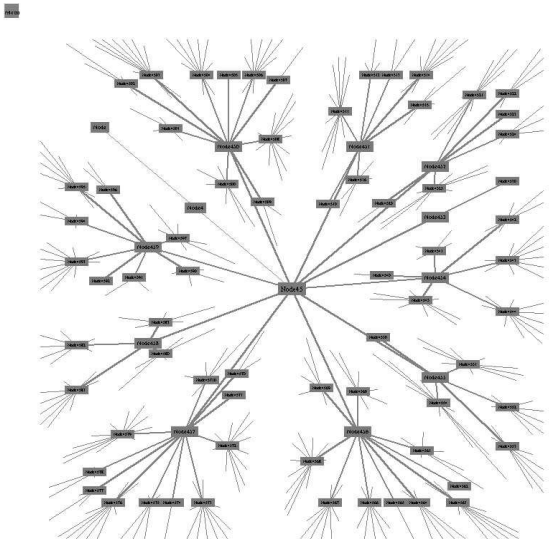


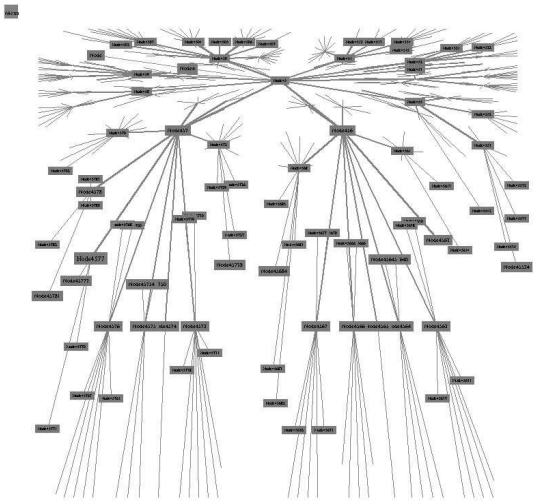
Figure 5. An example when a node is selected *(Semantic Zooming) [Nguyen and Huang, 2002]*



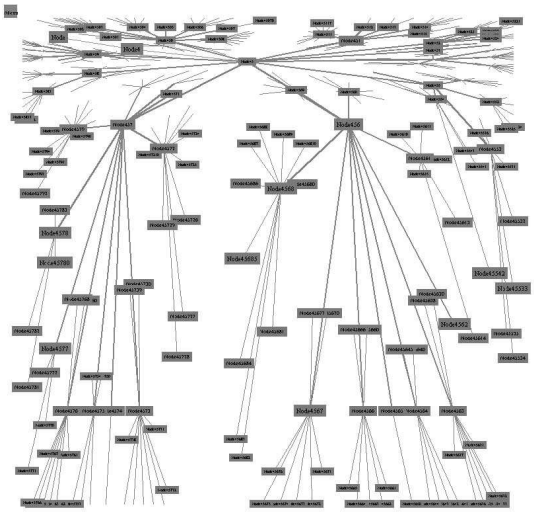
6(a)



6(b)

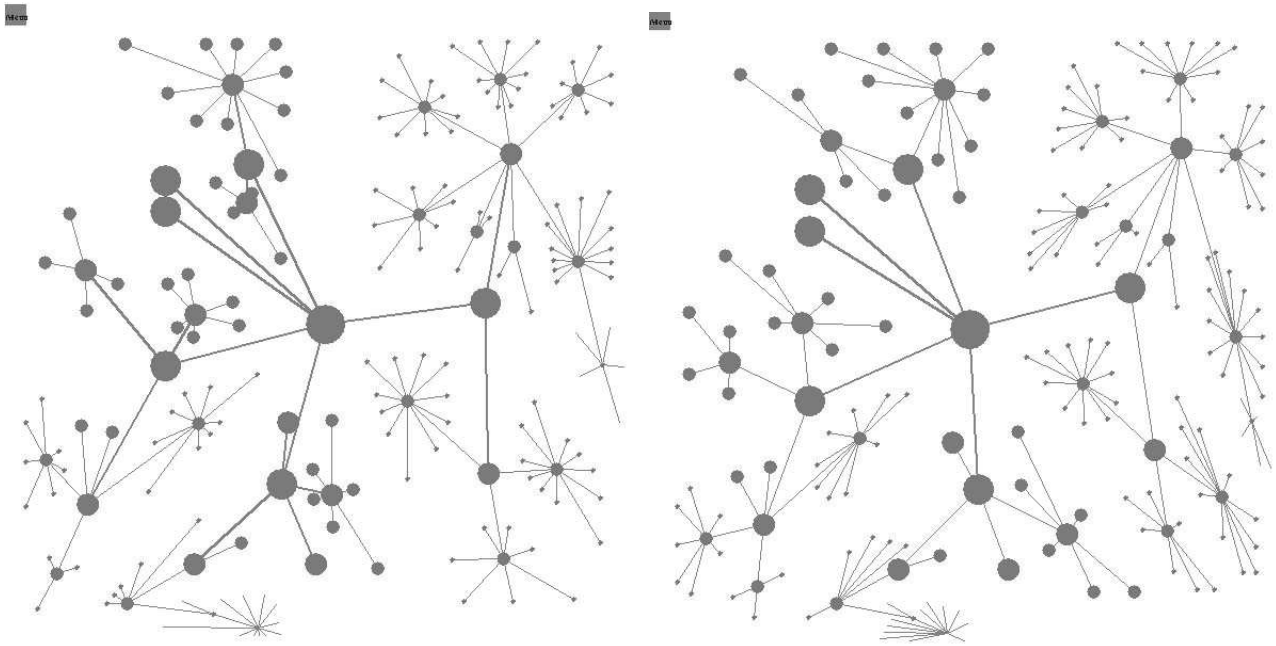


6(c)



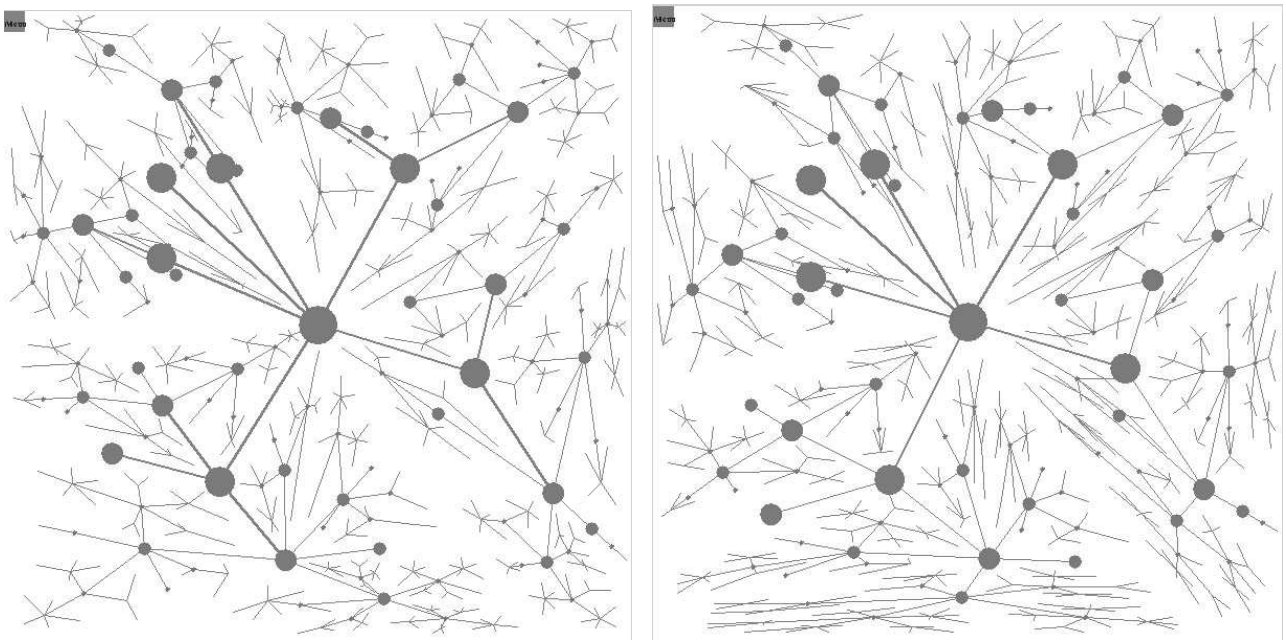
6(d)

Figure 6. An example of our navigation technique on a very large tree. Figure **(a)** shows the entirely tree layout which is too dense for the DualView focus+context technique. Figure **(b)** shows the sub-tree layout when we apply semantic zooming to reduce amount of displaying information. Figure **(c)** shows the layout when we apply *Browsing Transformation* to bring interested information into the focus region. Figure **(d)** shows the layout when we apply *Distortion Transformation* to magnify information at the focus region (i.e. around the centre). This transformation is applied when the information at the focus area is still dense.



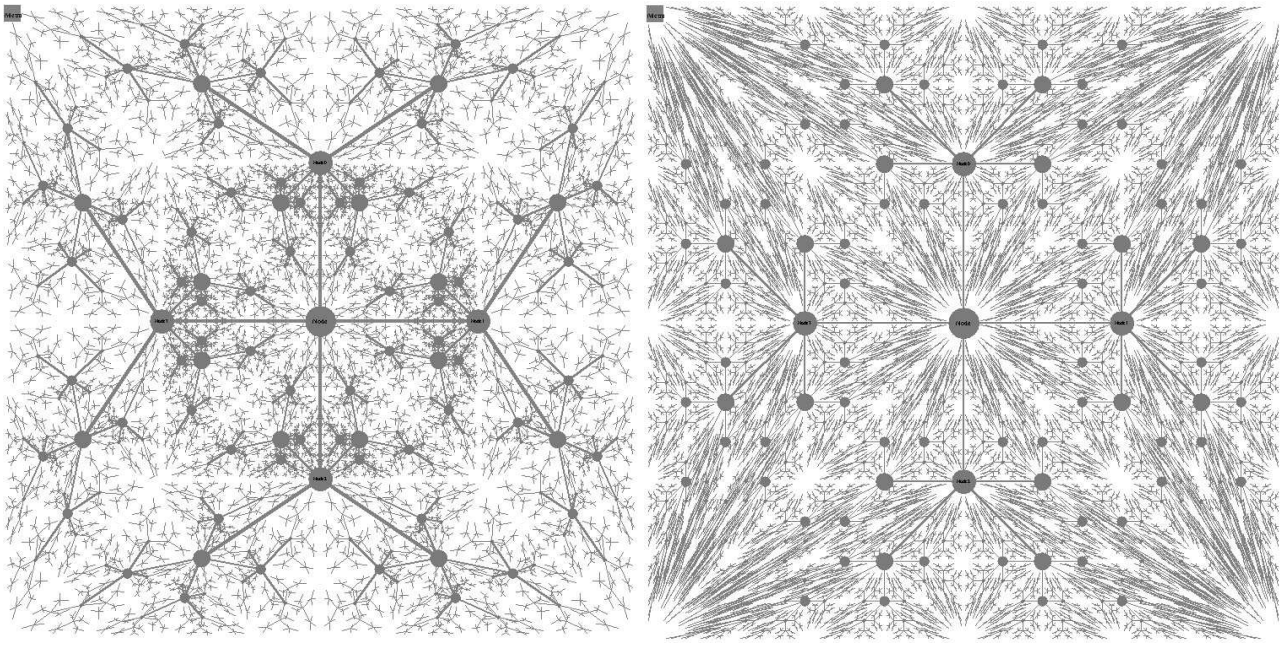
7(a) **7(b)**

Figure 7. An example of a medium large data set of approximately 170 nodes. Figure **(a)** shows the layout of the original SOTree, which running time is 1 seconds. Figure **(b)** shows the layout the improved SOTree, which running time is 1 seconds.



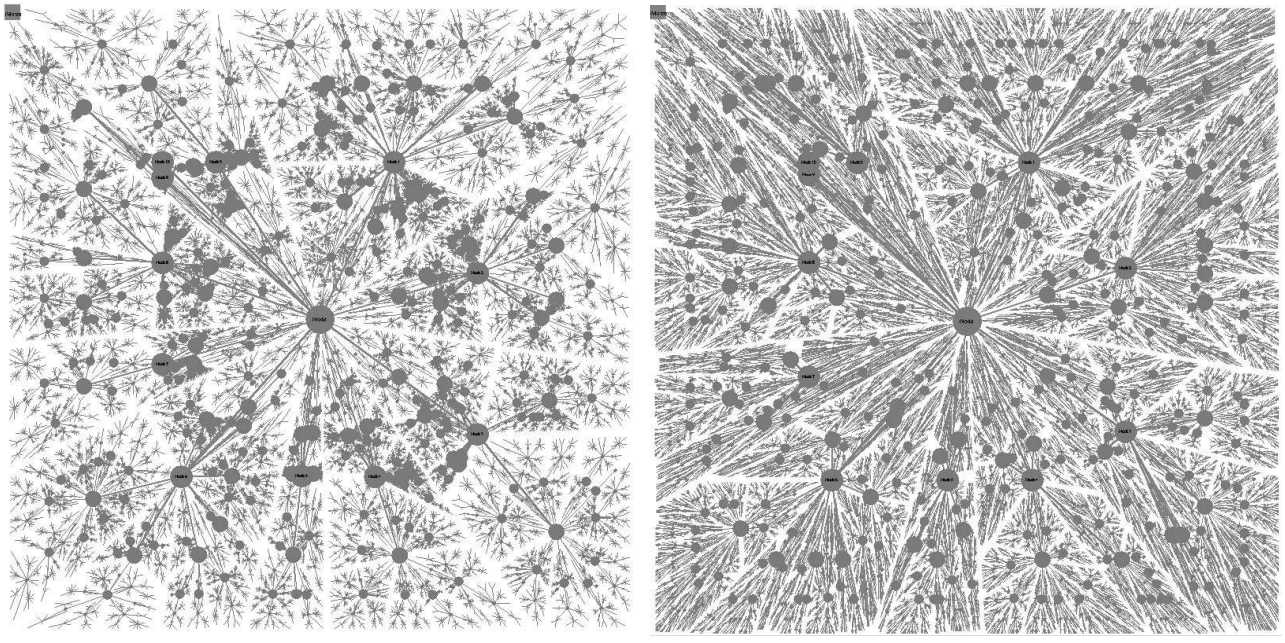
8(a) **8(b)**

Figure 8. An example of a large data set of approximately 750 nodes. Figure **(a)** shows the layout of the original SOTree, which running time is 2 seconds. Figure **(b)** shows the layout the improved SOTree, which running time is 2 seconds.



9(a) **9(b)**

Figure 9. An example of a very large uniform data set of approximately 22 000 nodes. Figure **(a)** shows the layout of the original SOTree, which running time is 4 minutes 50 seconds. Figure **(b)** shows the layout the improved SOTree, which running time is 5 minutes 10 seconds.



10(a) **10(b)**

Figure 10. An example of a huge data set of approximately 50 000 nodes. Figure **(a)** shows the layout of the original SOTree, which running time is 22 minutes 45 seconds. Figure **(b)** shows the layout the improved SOTree, which running time is 25 minutes 30 seconds.

5 References

NGUYEN, Q. V., HUANG, M. L. (2002): A space-optimized tree visualization. *To Appear at IEEE Symposium on Information Visualization 2002 (InfoVis'02),* Boston, USA.

NGUYEN, Q. V., HUANG, M. L. (2002): Using space-optimized tree visualization for web site-mapping. *Proc. International Conference on Internet Computing (IC'02),* Las Vegas, Nevada, USA, **3**:622-628.

HERMAN, I., MELANCON, G., MARSHALL, M.S. (2000): Graph visualization in information visualization: a survey. *IEEE Transactions on Visualization and Computer Graphics*, **6**(1):24-44.

CARD, S. K., MACKINLAY, J. D., SHNEIDERMAN, B. (1999): *Readings in information visualization – using vision to think*. San Francisco, California, Morgan Kaufmann Publishers.

Spence, R. (2001): *Information visualization*. ACM Press, Addison-Wesley.

JOHNSON, B., SHNEIDERMAN, B. (1991): Tree-maps: a space-filling approach to the visualization of hierarchical information structures. *Proc. the 1991 IEEE Visualization*, IEEE, Piscataway, NJ, 284-291.

QUINGWEN F. (1997): Algorithms for drawing clustered graphs. Ph.D. thesis. The Department of Computer Science and Software Engineering, The University of Newcastle, Australia.

KOIKE, H., YOSHIHARA, H. (1993): Fractal approaches for visualizing huge hierarchies. *Proc. the 1993 IEEE Symposium on Visual Languages*, IEEE/CS, 55-60.

REINGOLD E. M., TILFORD, J. S. (1981): Tidier drawing of trees. *IEEE Transactions on Software Engineering*, SE-**7**(2):223-228.

SHILOACH, Y. (1976): Arrangements of planar graphs on the planar lattices. Ph.D. thesis. Weizmann Institute of Science, Rehovot, Israel.

EADES, P. (1992): Drawing free trees. *Bulleting of the Institute fro Combinatorics and its Applications.* 10-36.

JEONG, C. S., PANG, A. (1998): Reconfigurable disc trees for visualizing large hierarchical information space. *Proc. IEEE Symposium on Information Visualization (InfoVis '98).* 19-25.

LAMPING, J., RAO, R. (1995): The hyperbolic browser: a focus+context technique for visualizing large hierarchies. *Journal of Visual Languages and Computing.* **7**(1):33-55.

BATTISTA, G. D., EADES, P., TAMASSIA, R., TOLLIS, I. G. (1995): Graph drawing. *IEEE Transactions on Visualization and Computer Graphics.* **1**(1) 16-28.

WILLS, J. G. (-): NichesWorks – Interactive visualization of very large graphs. Lucent Technologies (Bell Laboratories), USA. http://www.bell-labs.com/user/gwills/NICHEguide/nichepaper.html.

ROBERTSON, G. G., MACKINLAY, J. D., CARD, S. K. (1991): Cone trees: animated 3D visualizations of hierarchical information. *Human Factors in Computing Systems, CHI '91 Conference Proceedings*, ACM Press. 189-194.

CHUAH, M. C., ROTH, S. F., (1998): Dynamic aggregation with circular visual designs. *Proc. IEEE Symposium on Information Visualization*. North Carolina. 35-43.

HERMAN, I., MELANÇON, G., RUITER, M. M., DELEST, M. (1999): Latour - a tree visualization system. *Graph Drawing*. 392-399.

MELANÇON G., HERMAN, I. (1998): Circular drawings of rooted trees. *Reports of the Centre for Mathematics and Computer Sciences.* INS-9817, ISSN 1386-3681.

HERMAN, M. D., MELANCON, G. (1998): Tree visualization and navigation clues for information visualization. In: *Computer Graphics Forum 17*. 153-165.

CHI, H., PITKOW, J., MACKINLAY, J., PIROLLI, O., GOSSWEILER, R., CARD, S. K. (1998): Visualizing the evolution of web ecologies. *Proc. ACM CHI 98 Conference on Human Factors in Computing Systems*, ACM Press, Los Angeles, California. 400-407, 644-645.

Kleiberg, E., Wetering, H. V., Wijk, J. J. (2001): Botanical visualization of huge hierarchies. *Proc. The Symposium on Information Visualization (InfoVis'01),* 87-94.