EncCon: An Approach to Constructing Interactive Visualization of Large Hierarchical Data

Title	EncCon: An Approach to Constructing Interactive Visualization of Large Hierarchical Data
Authors	Quang Vinh Nguyen and Mao Lin Huang
Classification	Paper
Corresponding Author	Quang Vinh Nguyen

Department of Computer Systems, Faculty of Information Technology
University of Technology, Sydney
PO BOX 123 Broadway, NSW 2007, Australia
Telephone: +61 2 95144519; Fax: +61 2 95141807
Email: quvnguye@it.uts.edu.au

Abstract

This paper describes a new technique called EncCon for visualizing and navigating large hierarchical information. This technique consists of two components, the visualization and the navigation. The visualization uses a fast enclosure+connection method to calculate the geometrical layout for the display of large hierarchies in a two-dimensional space. Our technique uses a rectangular division algorithm for recursively positioning the graph. This visualisation aims to maximize the utilization of display space while retaining a good geometrical layout as well as a clear (explicit) presentation of the hierarchical structure of graphs. This paper also presents an experimental evaluation of EncCon's layout algorithm.

Besides the layout algorithm, EncCon uses a new focus+context viewing technique for the navigation of large hierarchies. We use the *zooming+layering* concept to achieve the focus+context viewing, rather than the traditional *enlarge+embedded* concept which is used by most of the available focus+context techniques. Technically, it employs semi-transparency to achieve the display of two layers of information in z-coordination at the same visualization. Both *context view* and *detail view* are drawn at two separate layers. These layers are then displayed in an overlapped manner at the same physical screen space.

Keywords: information visualization, tree layout, hierarchical visualization, navigation, focus+context, transparency, interactive visualization

Introduction

With the rapid growth of information, there is a huge amount of data that has become available for analysis. The size of this data is rapidly increasing each year. The traditional user interfaces provided for information systems, with the typical working mode of textual display plus scrolling bar that often generate long textual pages of output data for user to read, no longer meet the satisfaction, in terms of human cognitive process, of users, such as *Business Managers* and *System Analysers* who are using these systems. It is very time consuming for these users to read through the information line by line through the whole page or several pages to find particular data items they need. Furthermore, it is even hard for them to extract some inter-relationships among the data items presented across several pages. These problems with the current user interfaces have led many designers of information systems, to realize the need for the development of new generation user interfaces that can reduce the human cognitive cost for their next generation information systems. As a result, they have started to use a new technique called *Interactive Visualization* for the design of their new user interfaces.

Interactive visualization is defined by Colin Ware [1] as "*a process made up of a number of interlocking feedback loops that fall into three broad classes. At the lowest level is the data manipulation loop, through which objects are selected and moved using the basic skills of eye-hand coordination. ….At an intermediate level is an exploration and navigation loop, through which an analyst finds his or her way in a large visual data space…..At the highest level is a problem-solving loop though which the analyst forms hypotheses about the data and refines them through an augmented visualization process*."

Interactive visualization, as defined above, is a different way of thinking about information processing in information systems. There is no longer a task-driven user who formulates a precise query, types it into the database search form, retrieves the result and then leaves the computer. Information processing becomes a continuous exploration through the visual mappings. The user is in the information space more or less all the time, together with a multitude of heterogeneous information sources. She/he can explore, view, investigate, discover, learn and manipulate through the visual metaphors.

There are two important steps that are involved in the design of interactive visualization. The first step is to map the relational data into a geometrical plane. This is the fundamental problem of the visualization. We call it the graph drawing problem or the layout problem. The second step is the view navigation. The view navigation is about changing views interactively, step by step, to reach the target information that a particular user wants. This step is defined as the second loop of interactive visualization by Colin Ware [1]. We now introduce some basics about the layout design and navigation design.

The layout problem in visualization

In practice, there are many information sources that are organised in hierarchical form. Therefore, the research of hierarchical layout becomes one of the major areas in information visualization. Chaomei Chen stated in his book [2] that "*Hierarchies are one of the most commonly used structures. The organizational structure of a file system can be represented as a hierarchy; the structure of a classification system is a hierarchy; and a taxonomy of all animals is also a hierarchy. Hierarchical structures not only play significant roles in their own right, but also provide a means of representing a complex structure in a simplified form*".

Research in hierarchical visualization can be roughly classified into two main streams: the *connection* approach and the *enclosure* approach. They are both effective approaches for the visualization of hierarchies and the use of each approach depends primarily on the properties of the data in a particular application domain. Figure 1 shows an example of *Classical Hierarchical Layout* [3] which is a typical connection approach. Figure 2 shows a typical enclosure example which is called *Tree-maps* [4].

- The *connection* approach, examples are *Classical Hierarchical View* [3], *H-tree Layout* [5], *Radial View* [6], *Balloon View* [7], *Disk Tree* [8], *NicheWorks* [9], *Rings* [10], *Narcissus* [11], *Space-Tree* [12], *Hyperbolic Browser* [13, 14], *Cone-Tree* [15], *Botanical Visualization* [16], *Internet Mapping* [17], etc., uses a node-link diagram that displays the relationships of information explicitly. This approach generally gives users an immediate perception of the relationships.

However, it is not often efficient in terms of utilizing display space because most of the pixels are wasted as background.

- The *enclosure* approach is an excellent solution in term of optimising the use of display space. It was first proposed by Johnson and Shneiderman [4] in 1991. They used a new space-filling concept to map the hierarchical information to a rectangular two-dimensional space and it ensured 100% use of the display space. Later some other similar methods were proposed and implemented. Currently, the most commonly used enclosure techniques include *Tree-Maps* [4], *Cushion Tree-Maps* [18], *Squarified Tree-Maps* [19], *Sunburst* [20], *InteRing* [21] and *Venn diagram* [22]. The Tree-maps and its variations are highly capable of displaying a large amount of information within a limited display space. As a result, these techniques can be applied to visualise large hierarchies. The Tree-map techniques have made a great contribution to the field of information visualization through a number of successful applications in stock market data, file systems visualization, image browser, etc [23]. Although space-filling is a great approach for visualising large hierarchies, most of these techniques do not show the relational structure of information explicitly, except the *InterRing* technique. This costs extra cognitive effort for viewers to perceive and understand the relational structures that are presented implicitly in the enclosure manner (see more discussion from Nguyen & Huang [24]). Note that the *InterRing* and *Sunburst* techniques are not a pure space-filling approach which will still produce some unused spaces in their visualization. A comprehensive comparative study of various tree-maps techniques was carried out by Bederson et al. [23].

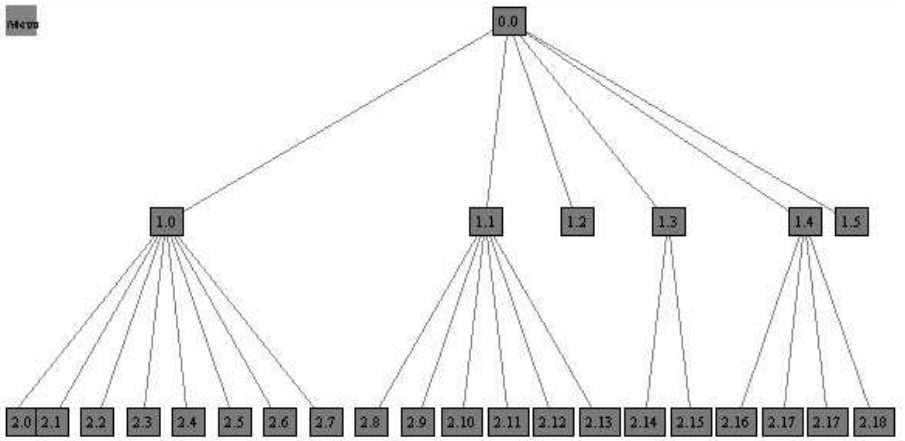


Figure 1. An example of Classical Hierarchical Layout (adapted from Reingold & Tilford [3]).

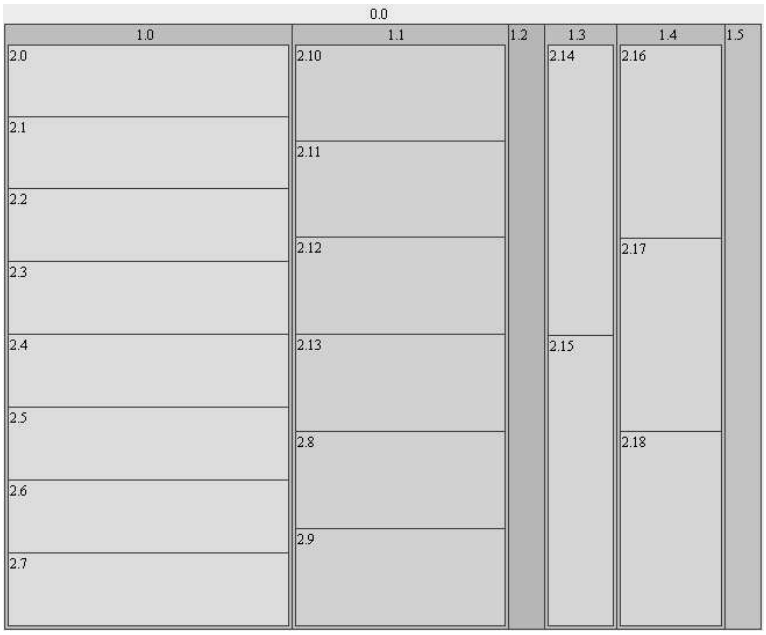


Figure 2. An example of Tree-Maps (adapted from Johnson & Shneiderman [4]).

To be able to inherit the advantages of both approaches, a compromised approach that combines both the *connection* and the *enclosure* concepts into one for the visualization of large hierarchies has been proposed. These techniques include *Space-Optimised Tree (SO-Tree)* [24] and *FlexTree* [25]. The *SO-Tree* essentially inherited the space-filling concept proposed by Johnson and Shneiderman [4] to maximise the utilization of

display space for visualizing large hierarchies. Technically, this technique uses a polygonal area division method for the recursive positioning of nodes and their sub-trees. *SO-Tree* also uses a node-link diagram which aims to enhance the understanding of the hierarchical structures. We classify this kind of technique as *connection+enclosure* approach. However, the polygons are sometimes not good shapes for viewers to perceive. Song et al proposed a multiple foci technique, called *FlexTree* [25], which enhances the scalability by introducing the space-filling concept into node-link diagrams, as claimed by the authors. This technique compresses the width of the tree presentation by reducing both the height of nodes and the empty space between nodes that are presented in the same level. This aims to increase the utilization of display space. Although the user can navigate through hierarchy via multiple foci, the compressed layout generally is not aesthetically nice and clear.

The view navigation problem

Another important step involved in the interactive visualization process is view navigation. Chaomei Chen stated in his book [2] that "*navigation in a hierarchical structure involves moving from one node to another, along the existing hierarchical links in the structure. When the size of a hierarchy becomes large, it is desirable to enable users to have easy access to contextual information, as well as local details".* One of the most important issues involved in navigation is that the users are always able to see (or have easy access to) contextual information. This allows users to maintain the perception of where they are and where they can move from during the navigation of large information spaces. This also assists users to make further decisions about where they should go next, as well as where they are, while interactively navigating through the information space.

To achieve easy access to both the *contextual* and the *local* information in large hierarchies, there are several optimized navigation solutions that have been developed to accommodate with the geometrical layout techniques. Most of the current navigation techniques can be roughly classified into three approaches: *focus+context*, *zooming+filtering*, and *incremental exploration*.

- Focus+context – this approach is defined as a viewing approach that provides users with a detailed view of a small focus area and a global view of the overall context. In other words, it can be defined as "*detailed views of particular parts of an information set are blended in some way with a view of the overall structure of the set*". Typical focus+context techniques are *Fisheye Views* [26, 27], *Polyfocal Display* [28], *Bifocal Lens* [29], *Perspective Wall* [30], *Hyperbolic Browser* [13], etc.

- Zooming+filtering – this approach is defined as a viewing approach that works by reducing the amount of context in the display. The reduction is done by filtering the information in the form of selecting a subset of the data along a range of numerical values of one or more dimensions. The typical zooming+filtering techniques are *Starfield Display* [31], *Tree-Maps* [4], *Pad* [32], *Pad++* [33] or the more recent version called *Piccolo* [34], etc.

- Incremental exploration – this approach is defined as a viewing approach that displays only a small portion of the full hierarchy incrementally following the user's exploration of information space. Thus, these techniques are able to handle huge data sets where it is impossible to display the entire hierarchy on the screen at a time. Incremental exploration techniques can be found at [35-37], etc.

When comparing the above three approaches, the focus+context techniques generally provide a better solution for accessing the contextual information during navigation. To achieve this, it uses *enlarge+embedded* or *enlarge+blended* concept (see Figure 3), which usually employs a distortion or/and a semantic zooming [22] technique to enlarge a portion of the information structure to form a *detail view* of the local information structure that a user is currently focusing on during the navigation. Therefore, we sometimes call this *detail view* as *focus view*. It then embeds this *focus view* into the *global view* of the overall contextual information structure for visualization. This approach is one of the most efficient and effective navigation strategies in the current design of interactive visualization, and it is widely used and commercialised in many visualisation and Graphic User Interface (GUI) tools. However, one obvious limitation of this approach is the “area division”, that is the whole geometrical area has to be divided into two parts; one for the drawing of local structure and another for the drawing of the rest of the global

structure. This, therefore, limits the amount of information that can be displayed in the *focus view*. Also that some focus+context approaches provide a (continuous) transition area between two views to reduce the human cognitive process in identifying the connection between two views (see Figure 3).

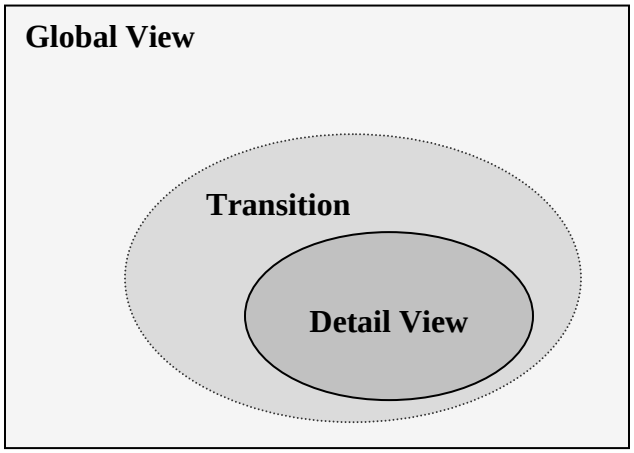


Figure 3. The traditional focus+context approach that uses the *enlarge+embedded* concept, divides one geometrical area into two for the display of the Detail View and the Global View.

The *zooming+filtering* and *incremental exploration* approaches do not provide standard strategies for accessing the contextual information during navigation. However, some of these techniques do provide extra mechanisms, such as landmark nodes and history path, to show the contextual information in a certain degree during the navigation. One such technique is *WebOFDAV* [38].

EncCon Visualization

Although there have been many proposed innovative techniques both in visualization and navigation to facilitate the design of interactive visualization, most of these techniques, however, do not consider all the aspects involved in interactive visualization design. Currently only a few interactive visualization solutions satisfy the multiple design requirements such as 1) space utilization, 2) fast computation, and 3) minimization of human cognitive process.

In this paper, we propose a new *Enclosure* and *Connection* approach (called *EncCon*) for the construction of interactive visualizations of large hierarchies. This technique consists of two components, the visualization and the navigation. *EncCon* uses a fast enclosure+connection algorithm to calculate the geometrical layout of large graphs in a two-dimensional space. It essentially inherits the advantage of space-filling techniques [*4, 18, 19, 24*] that maximise the utilization of display space by using area division for the partitioning of sub-trees and nodes. Note that the issue of space utilization becomes significantly important when visualizing large graphs with hundreds or thousands of nodes and edges because of the limitation of screen pixels. It is similar to the original *Tree-Maps* [4] and *Squarified Tree-Maps* [19] that use a rectangular division method for recursively positioning the nodes, rather than a polygonal division method used in *SO-Tree* [24]. This property aims to provide users with a more straightforward way to perceive the visualization and ensures the efficient use of display space.

In order to address the specific criteria of *EncCon* drawing, we use 'squarified' rectangles for the area division, which is similar to the *Squarified Tree-Maps* [19] algorithm. We will discuss our drawing criteria in the technical section. The *EncCon* drawing ensures that all sub-hierarchies are inside rectangular geometric local regions. Thus, there is no overlapping between rectangular local regions of nodes and their sub-trees. However, our area division algorithm is different from the *Squarified Tree-Maps* algorithm. In *Squarified Tree-Maps*, the partitioning is accomplished through the horizontal-vertical manner. In *EncCon*, this is achieved in the circular manner, in which all rectangles are placed in the north-east-south-west order around four sides of the parent rectangle. Both of the above partitioning algorithms ensure the efficiency of space utilization. The *EncCon* visualisation also uses a node-link diagram to present the hierarchical structure explicitly.

Besides the layout algorithm, EncCon also uses a focus+context viewing technique for the navigation of large hierarchies. It uses the *zooming+layering* concept (see Figure 4) to achieve the focus+context viewing, rather than the *enlarge+embedded* concept which is used by most of the other focus+context techniques. Technically, we employ the semi-transparency technique to achieve the display of two separate information layers in z-coordination on the same physical screen space. This makes the *context view* and

the *focus view* to be drawn and displayed in an overlapped manner on the same screen. This layering display enables the *focus view* can be displayed in a full screen size with more detailed information for users to see. These overlapped twin layers always keep one in highlighted full display and another in de-emphasized.

The transparency techniques have been used in a number of 3D visualization systems, such as the *Information Cube* [39], *Cone-Tree* [15] and the *Spiral Calendar* [40], etc. These applications mainly aim to solve the overlapping problem. However, these systems didn't used transparency techniques for navigating abstract graphs in two dimensional spaces. Lieberman [41] also presented a promising technique for navigating geometry-related information spaces. However, his research was focusing on the navigation of a spatial metaphor, rather than an abstract graph metaphor.

In EncCon, navigation is achieved interactively through zooming and swapping of views between full display and semi-transparent display. In addition, smooth transitions between views are achieved by fading in/out animations that help users maintain their orientations during the interaction and navigation, and preserve their mental map of views.

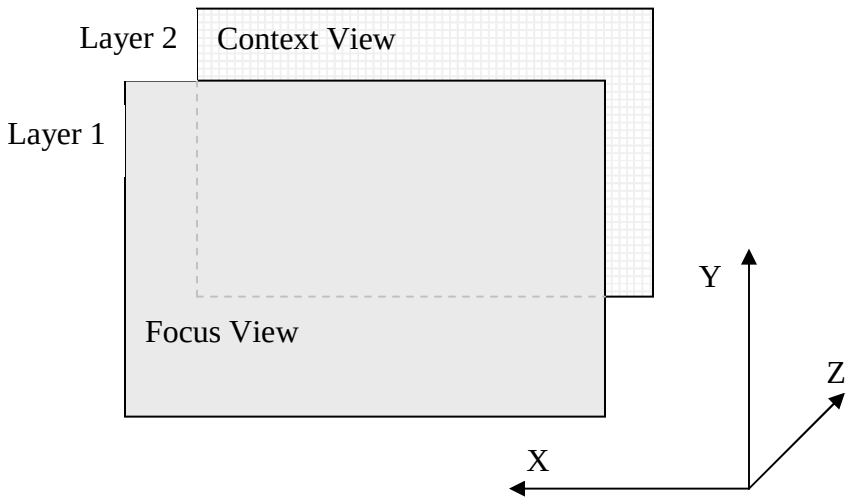


Figure 4. EncCon uses a new *zooming+layering* concept to draw the "context view" and "focus view" on two separate layers that are displayed in the same visualization by the use of semi-transparency technique.

Our *EncCon* technique can only be applied to rooted trees. We now review the terminology that is used in our technique.

Terminology

A tree is a connected graph without a cycle. A rooted tree consists of a tree T and a distinguished vertex r of T. The vertex r is called the root of T. In other words, T can be viewed as a directed acyclic graph with all edges oriented away from the root. If (μ,ν) is a directed edge in T, we then say μ is the parent of ν and ν is a child of μ. A leaf is a vertex with no children. If T contains the vertex ν, then the sub-tree $T(\nu)$ rooted at ν is the sub-graph induced by all vertices on paths originating from ν. We also use a node to represent a vertex ν with its displaying properties. This terminology is mainly mentioned in display section. Each vertex v has an associated value $w(\nu)$, which we call the weight. The local region $R(v)$ of the vertex ν is a rectangle, and it contains the drawing of a sub-tree $T(\nu)$. The rectangle $R(v_i)$ is proportional to the weight $w(v_i)$ of the vertex v_i.

A layered visualization LV consists of two graphical layers L_1 and L_2 of information, displayed in an overlapped manner in the same physical screen space, $LV = L_1 + L_2$ (see Figure 4). Graph $G = \{V, E\}$ is the model that presents the structure of an information space. Each graphical layer is the medium for the drawing of G or a sub-graph $G_i \in G$. At any time a graph G_i drawn in L_1 is always a sub-graph of G_j drawn in L_2. Thus we constantly have $G_i \in G_j \in G$.

Enclosure+Connection Layout

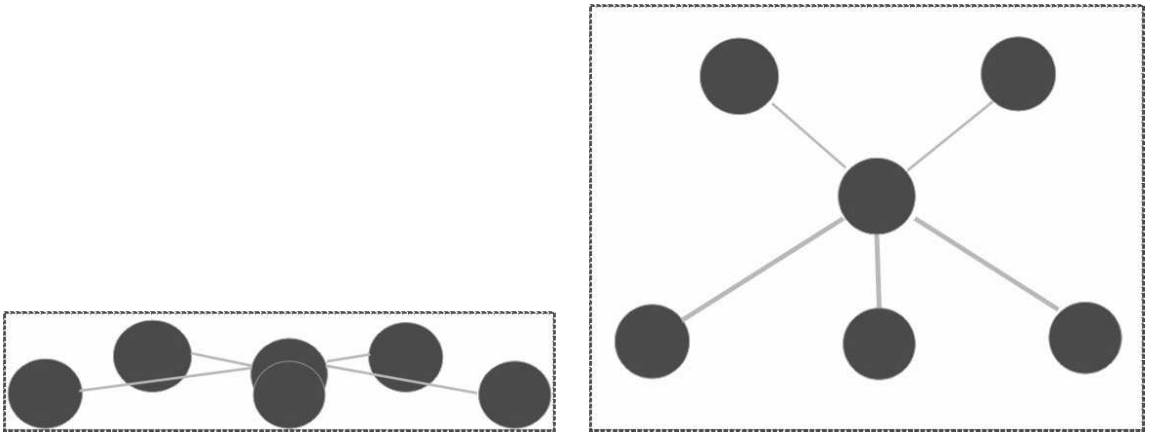
The layout algorithm, which is responsible for positioning of all the vertices $\{\nu_1,\nu_2, \ldots,\nu_n\}$ of the given tree T in a two-dimensional geometrical space, is governed by a particular area division algorithm. Each vertex v_i is bounded by a rectangular local region $R(v_i)$ and the drawing of the sub-tree $T(v_i)$ is restricted to inside the geometrical area of $R(v_i)$. Thus, the local region $R(v_i)$ of vertex v_i is the sum of the rectangular areas

assigned to its children. The position of vertex v_i is at the centre of the rectangle defined by $R(v_i)$. See the example in Figure 6a.

The area division algorithm, which is based on the space-filling concept initially proposed by Johnson and Shneiderman[4], is responsible for the calculation of local regions $\{R(v_1), R(v_2) \ldots, R(v_n)\}$ for the placement of all vertices $\{v_1, v_2, \ldots, v_n\}$ and their sub-trees in the given tree *T*.

The area division algorithm must address a specific area partitioning criteria, to derive a complete set of rectangular local regions $\{R(v_1), R(v_2) \ldots, R(v_n)\}$ that can produce a high quality layout of the node-link diagram of the corresponding tree *T*, in terms of satisfying the following general *Aesthetics rules* defined by Di Battista, Eade, Tamassia, and Tollis in their Graph Drawing book [42].

1. **Edge Crossings:** Edge crossings make it difficult to trace paths. So all edge crossings should be removed if the graph is planar, otherwise the total number of edge crossings should be minimised.
2. **Angular Resolution:** The angle between the edges of one vertex should be maximised. This aesthetic is especially relevant for straight-line drawings.
3. **Total Edge Length:** Minimization of the sum of the lengths of the edges. The average of all edge lengths should be as small as possible.
4. **Uniform Edge Length:** Minimization of the variance of the lengths of the edges. To minimize the number of long edges, that usually cost extra cognitive effort to perceive the parent-child relationships. To minimize the number of very short edges, that sometimes cause overlaps among the graphical nodes (see Figure 5).



(a) (b)

Figure 5. The left-hand side image shows an example of placing a node-link diagram in a thin rectangle. We can see that the quality of the graph presentation in this thin rectangle is much lower, in terms of *Angular Resolution* and *Uniform Edge Length*, than the quality of layout of the same graph presented in a square-like rectangle on the right-hand side. Note that the left-hand side diagram contains a node overlapping.

Obviously, from Figure 5a we can see that the use of thin (very thin) rectangular local regions for placing node-link diagrams will greatly reduce the quality of graph presentation. Therefore, the 'squarification' of local regions is essential in the design of the space partitioning algorithms, especially when node-link diagrams are placed to show relationships among vertices. This has led to the exclusion of original *tree-maps* [4] algorithm for the space partitioning in *EncCon*, since it naturally produces many thin rectangles (see Figure 2).

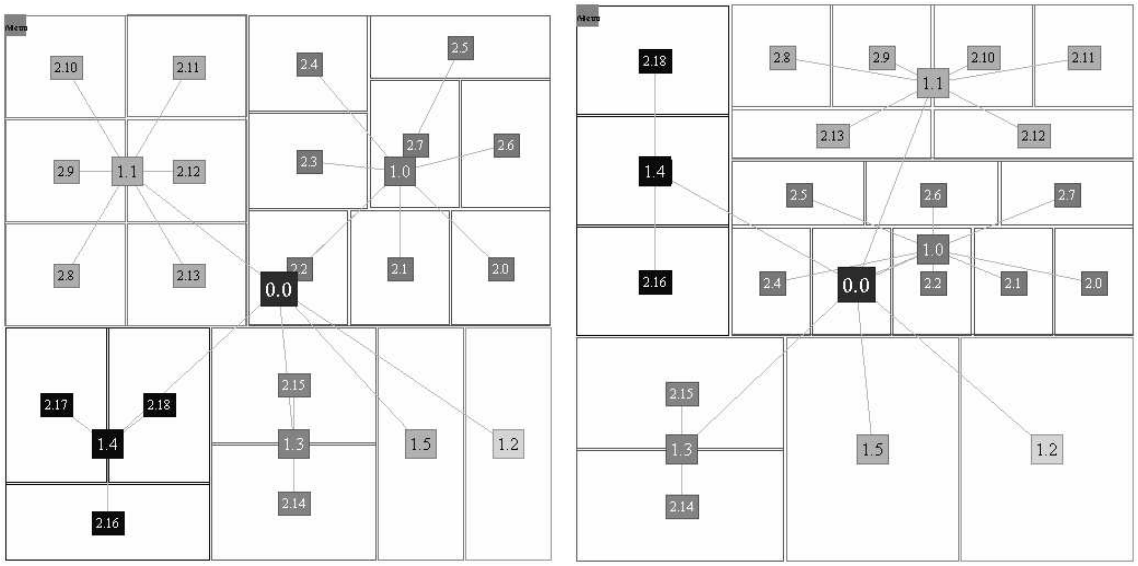
A drawing of sub-tree $T(v_i)$ rooted at v_i is calculated based on the properties of v_i and its local region $R(v_i)$. Each vertex is associated with a weight and the vertex's local region is calculated proportionally to its weight. There are several ways to define the weight of a vertex based on its physical or logical properties, such as a domain-specific property: the size of a file or a directory that is presented as a graphical node that we are visualizing. In this paper, however, we define the weight of vertex v_i to be proportional to the number of its descendants. The weights of all vertices are pre-calculated before the layout of the node-link diagram is drawn. This step is described in more detail in the next section.

Weight calculation

We assign a weight $w(v)$ to each vertex v for the calculation of the local region $R(v)$ that relates to the regions of its parent and siblings. Suppose that vertex v has k children $\{v_{l+1}, v_{l+2}, \ldots, v_{l+k}\}$. The calculation of $w(v)$ is done recursively from leaves to a vertex using the following formula:

$$w(v) = 1 + C\sum_{i=1}^{k} w(v_{l+i}) \quad (1)$$

Where C is a constant ($0 < C < 1$), and $w(v_{l+i})$ is the weight assigned to the i^{th} child of vertex v. The constant C is a scalar that determines the size difference of local regions of all vertices based on the number of descendants of those vertices. The value of C may be altered to adjust the layout for a better view that can better fulfil the aesthetics rules of graph drawing. We can see that in Figure 6a, the difference among the local regions is much higher than those in 6b. From our experiments that include those from Figures 12 to 15, and Application 1 and Application 2, it was found that the value $C=0.45$ generally gives the best view of layouts. Therefore, we set this value as the default value for the generation of other screenshots in the paper.



(a) (b)

Figure 6. (a) An example of the rectangular division using C = 0.45. (b) An example of the rectangular division using C = 0.1

Example of the partitioning

We first describe our partitioning algorithm using an example. Suppose that we have a rectangle with width 6 and height 4, we need to divide this rectangle into 5 rectangles whose weight are respectively *{4, 4, 2, 1, 2}*, and the starting partitioning side is the left side. The weight of this rectangle is 13.

The first step is to add a single rectangle with the weight 4 into the first division side (height_of_R_1 = 4*6/13, width_of_R_1 = 4). Next, we add the second rectangle (weight 4) below the first, i.e. they share the common left side from the original large rectangle. These two rectangles will have the dimension respectively of (*height_of_R_1* = 8*6/13, *width_of_R_1* = 4*4/8) and (*height_of_R_2* = 8*6/13, *width_of_R_2* = 4*4/8). Next as step 3 involves inserting the third rectangle (weight 2) at the bottom of two rectangles. However, this step is dismissed because the last produced rectangle is too thin (*height_of_R_3* = 10*6/13, *width_of_R_3* = 2*4/10, $\lambda_3 = 0.174$). We now start the second partitioning circle, moving from the left side to the top side. In the remaining rectangle, the division continues on the second side (on the top). Two more

circles are repeated on other two sides of the *remaining rectangle* until all rectangles have been positioned (see Figure 7).

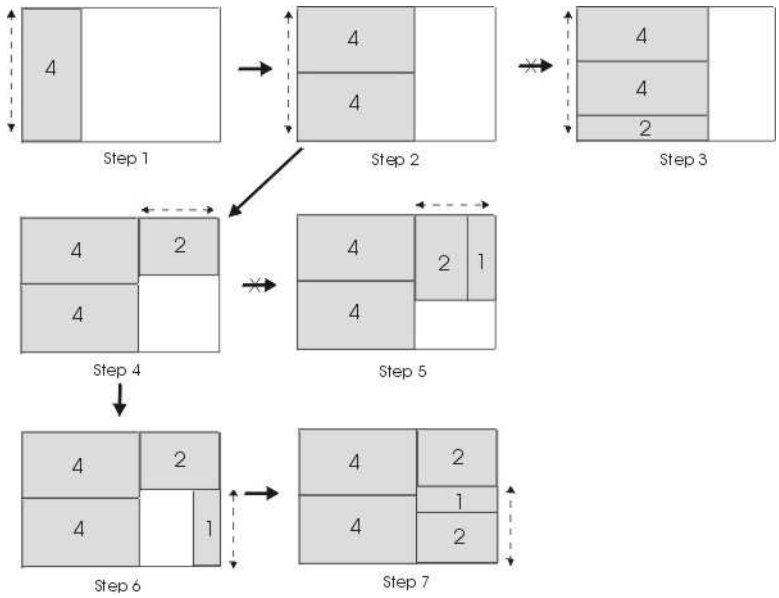


Figure 7. An example of partitioning using EncCon's algorithm.

Local region partitioning

We first define the local region of the root-vertex to be the entire rectangular display area. The root-vertex is placed at the centre of this rectangle. The partitioning starts from the root-vertex and ends when all the leaf-vertices are reached. Suppose that the rectangular local region $R(v)$ for vertex v is defined, and the position of v is at the centre of $R(v)$. We then need to calculate the local regions $\{R(v_{l+1}), R(v_{l+2}), \ldots, R(v_{l+k})\}$ for all the children $\{v_{l+1}, v_{l+2}, \ldots, v_{l+k}\}$ of vertex v. The partitioning ensures that the area of each rectangle $R(v_{l+i})$ *is* proportional to the weight $w(v_{l+i})$ of the vertex v_{l+i}.

The division of $R(v)$ into sub-regions $\{ R(v_{l+1}), R(v_{l+2}),\ldots, R(v_{l+k})\}$ is processed as below:

Suppose that the vertex μ is the parent of vertex v. We first find a side from rectangle *R(v)*, called *initial side*, where the vector $\overrightarrow{\mu v}$ cuts the rectangle *R(v)* or the side which is closest to μ if $\overrightarrow{\mu v}$ does not cut the rectangle. If v is the root then the *initial side* is defaulted as the bottom side of *R(v)*. We start the partitioning on the opposite side of the *initial side*, and the partitioning is then applied respectively to each of four sides (east-south-west-north) of *R(v)* in a circular manner at the clock-wise direction. This procedure is further described as below:

```
procedure rectangle-partition(array children, rectangle div-rectangle)
begin
    rectangle remaining-rectangle := firstside-partition(children, div-rectangle);
    if remaining-rectangle != null then
        remaining-rectangle := secondside-partition(get_remaining_chidlren(), remaining-rectangle);
        if remaining-rectangle != null then
            remaining-rectangle := thirdside-partition(get_remaining_chidlren(), remaining-rectangle);
            if remaining-rectangle != null then
                remaining-rectangle := fourthside-partition(get_remaining_chidlren(), remaining-rectangle);
            fi
        fi
    fi
    return remaining-rectangle;
end
```

On each side, the partitioning creates and fills in m rectangles ($m \leq k$) with the same width (or height). Thus, these sub-regions will form a large rectangle and leave a rectangular non-partitioned area (see Figure 8). Then, this non-partitioned rectangle (or also called *remaining rectangle*) is used for the next division on the next side. The number m is determined by the size of the *remaining rectangle* as well as the smallest ratio λ_{k+i} $(1 \leq i \leq m)$ of all ratios $\{\lambda_{k+1}, \lambda_{k+2},\ldots,\lambda_{k+m}\}$, where we have:

$$\lambda_{k+i} = \frac{width\_of\_R(v_{k+i})}{height\_of\_R(v_{k+i})} \quad (2)$$

Suppose that the partitioning is on one side of the *remaining rectangle*. We first check the numbers of children that can be added into this side. We want to maximize the number *m*, but also ensure that every sub-rectangle on the side is roughly square, which must satisfy with the following equation:

$$\frac{1}{\rho} < \lambda_{k+i} < \rho \quad (3)$$

Where ρ is a constant. We set ρ = 1.5 in our implementations that ensures the quality of a squared-like space partitioning, and the robustness of the layout algorithm.

Suppose that there are *m* children $\{v_{k+1}, v_{k+2}, \ldots, v_{k+m}\}$ of vertex *v* needing to be added into a side of the *remaining rectangle*, with respectively local regions $\{R(v_{k+1}), R(v_{k+2}), \ldots, R(v_{k+m})\}$. Suppose that the widths of sub-rectangles $\{R(v_{k+1}), R(v_{k+2}), \ldots, R(v_{k+m})\}$ are same as the direction of the partitioned side; the length of this side is l_1 and the length of the other side is l_2. Then the width and height of the rectangle $R(v_{k+i})$ are calculated by the following formulas:

$$width\_of\_R(v_{k+i}) = l_1 \frac{w(v_{k+i})}{\sum_{j=1}^{m} w(v_{k+j})} \quad (4)$$

$$height\_of\_R(v_{k+i}) = l_2 \frac{\sum_{j=1}^{m} w(v_{k+j})}{RW} \quad (5)$$

Where $w(v_{k+j})$ is the weight of the vertex v_{k+j}. *RW* is the weight of the *remaining rectangle*. *RW* is initially defined as the total weight of all the children $\{v_{k+1}, v_{k+2}, \ldots, v_{k+m}\}$. The value of *RW* after this partitioning is:

$$RW = RW - \sum_{j=1}^{m} w(v_{k+j}) \quad (6)$$

The above formulas ensure that the area of each child is proportional to the weight of the child. The partitioning happens around 4 sides of the rectangle *R(v)*. The Figure 8 illustrates the partitioning on the left side of a rectangle.

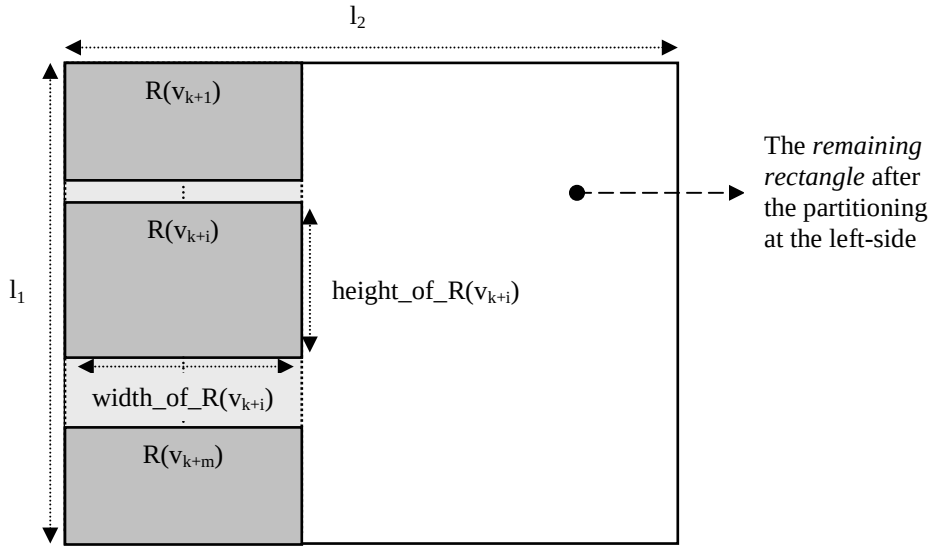


Figure 8. The partitioning on the left side of the rectangle.

However, this area division might not be completed because of the ratio of height and width of a particular rectangle does not satisfy the equation 3. There are two approaches to solve this problem.

Approach 1: the initial weight *RW* of divided rectangle *R(v)* is increased slightly and all the steps of the partitioning around 4 sides of the rectangle are implemented again. This procedure repeats until all the sub-rectangles are fit in the rectangle *R(v)*. This approach ensures all the sub-rectangles are placed circularly around four sides of the *R(v)*. This technique improves the aesthetical niceness of overall

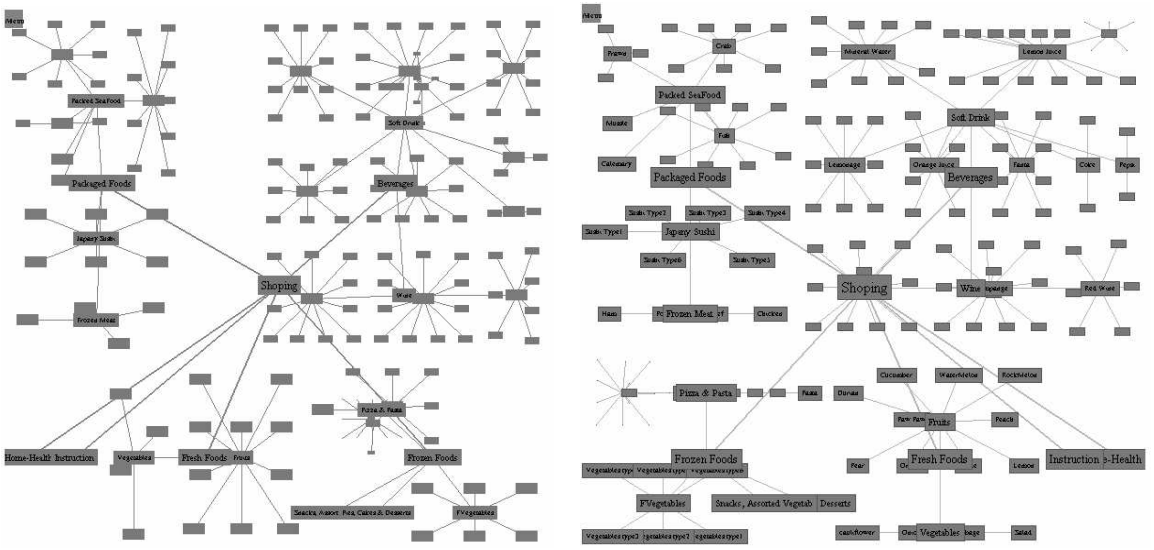
enclosure layout. However, this approach does not entirely utilise the display space. Figure 9a shows the layout of a data set using this approach. This procedure is formally defined as below:

```
procedure partition(array children, rectangle div-rectangle)
begin
   rectangle remaining-rectangle = rectangle-partition(children, div-rectangle);
   if remaining-rectangle != null then
         div-rectangle.weight := C*div-rectangle.weight;  /* C is a constant to increase the weight*/
         partition(children, div-rectangle);
   fi
end
```

Approach 2: we continue to partition the *remaining rectangle* until all sub-rectangles are fit into the rectangle *R(v).* In this case, the equation 3 is not applied to the procedure of partitioning to the last node on the side. This approach ensures 100% space efficiency but it might also reduce the aesthetically niceness of the overall enclosure layout in comparison with the previous approach. The algorithm works best with the list of vertices in ascendant order of weight. Thus, the list of vertices $\{v_1, v_2, \ldots, v_n\}$ is sorted in increasing order of weights before we calculate the sub-regions for these vertices. The Figure 9b shows the layout of the same graph as presented in Figure 9a, produced by the second approach. This procedure is formally defined as below:

```
procedure partition(array children, rectangle div-rectangle)
begin
   rectangle remaining-rectangle = rectangle-partition(children, div-rectangle);
   if remaining-rectangle != null then
         partition(get_remaining_chidlren(), remaining-rectangle);
   fi
end
```

From our experiments that include those from Figures 12 to 15, and the Application 1 and Application 2, it has been found that the Approach 2 often produces better layouts than Approach 1 does. The layouts used in other examples of this paper are generated by the second approach. Figures 12, 13, 14 and 15 in this paper are examples of the visualization using *EncCon* layout algorithm on various data sets.



(a) (b)

Figure 9. The example layouts of a medium size data set with approximately 170 nodes. The layout in picture (a) is generated by Approach 1 and the layout in picture (b) is generated by Approach 2.

The Comparison of different partitioning algorithms for placing node-link diagrams

The *Squarified Tree-Maps* [19] also aims to produce square-like rectangles and it produces similar space partitioning outcomes as *EncCon* does. Therefore, the partitioning outcomes produced by *Squarified Tree-Maps*[19] also satisfy the space partitioning requirements for *EncCon* visualization. However, technically these two approaches are different. The *EncCon* partitioning algorithm places the local regions around four sides (east-south-west-north) of the parent rectangle in a circular manner (see Figure 10a), while most of other space-filling algorithms including the *Squarified Tree-Maps* partition rectangles in a vertical-horizontal manner as illustrated in Figure 10b. The second partitioning approach in *EncCon* also repeats the

above process inside the *remaining rectangle* until all local regions are calculated. Our partitioning ensures the efficiency of space utilization as most of the space-filling approaches do. This process aims to enhance a good distribution of nodes and edges of a given node-link diagram in its rectangular local region.

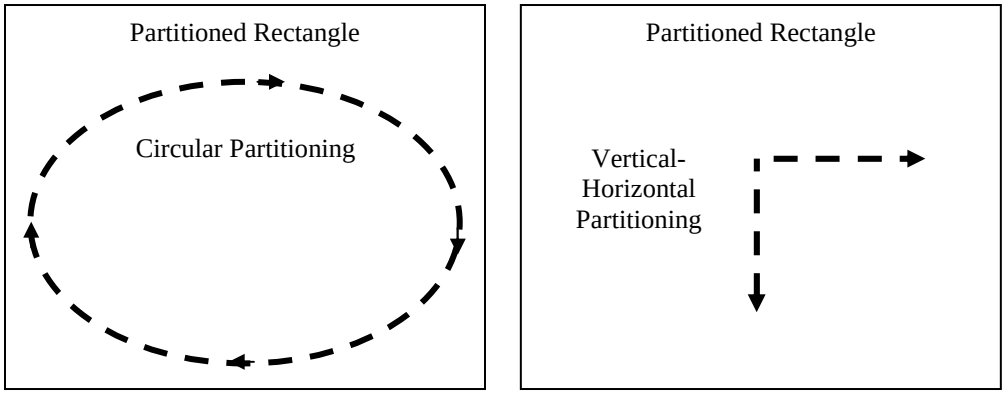
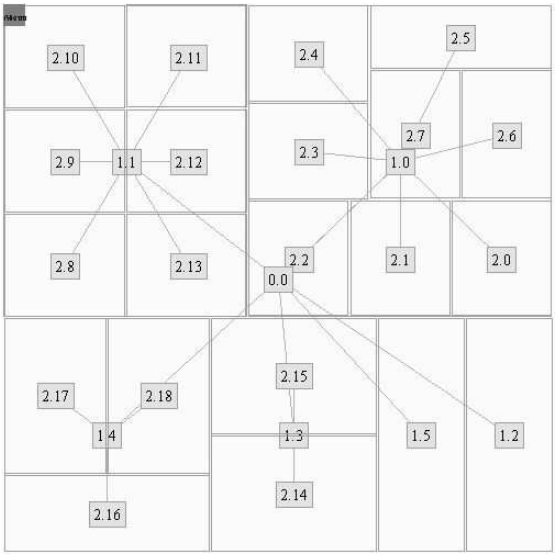


Figure 10. The diagram (a) shows the partitioning direction in *EncCon*. The diagram (b) shows the partitioning direction in *Squarified Tree-Maps*.

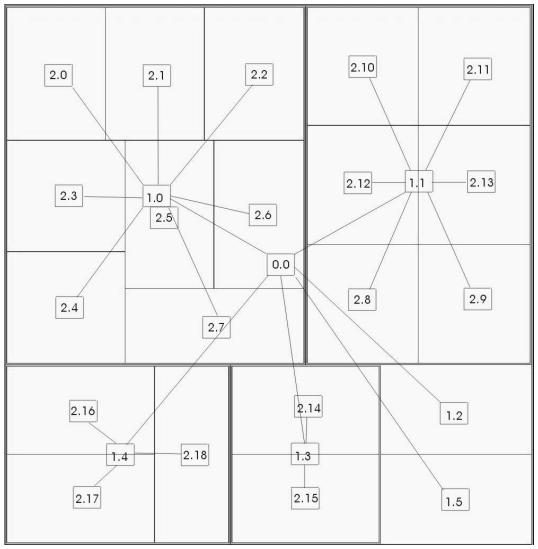
Therefore, we believe that our partitioning algorithms can improve the aesthetical appear of the layout of any given graph. In addition, our algorithm also checks the 'squared' aspect of rectangular local regions by using a simplified process (see equations 2 and 3).

Figure 11 shows an example of space partitioning using four different partitioning algorithms, including the one we used in *EncCon* and another one used in *Squarified Tree-Maps*. We then use our vertex positioning rules to place a small node-link diagram in each of these images so that we can observe the quality of these layouts generated by different partitioning algorithms. Clearly, for small data sets the differences between these algorithms, in terms of layout quality, computational time and space utilization, are not significant. We can see from these images in Figure 11 that the output layouts are similar, except for the layout generated by the original tree-maps algorithm as shown in Figure 11d. By using this 'slice and dice' tree-maps for the partitioning, the hierarchical node-link diagram is heavily overlapped and hard to be seen. From this example, we see that both *Squarified Tree-Maps* and *EncCon* partitioning can produce relatively good layouts for small node-link diagrams. However, the real advantage of *EncCon* visualization is to

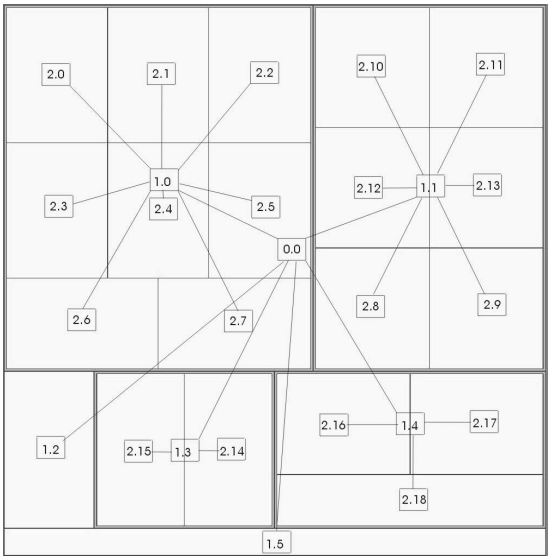
present large data sets with the high density layouts, where vertices of the graph are well distributed and the space utilization is optimised (see Figures 14 and 15). In the next section, we will use several large data sets to conduct a formal evaluation of these square-like area partitioning algorithms, including *Squarified Tree-Maps* and *EncCon* partitioning, to against the *Aesthetics Rules* defined in graph drawing. This will enable us to compare these algorithms in terms of producing high quality layouts.



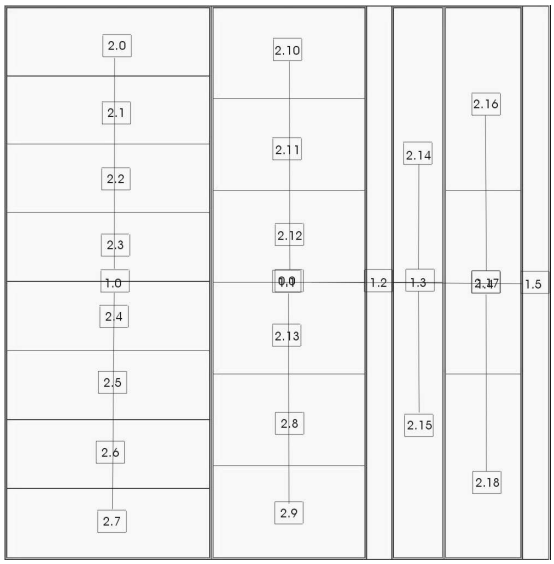
(a) EncCon partitining



(b) Squarified Tree-Maps

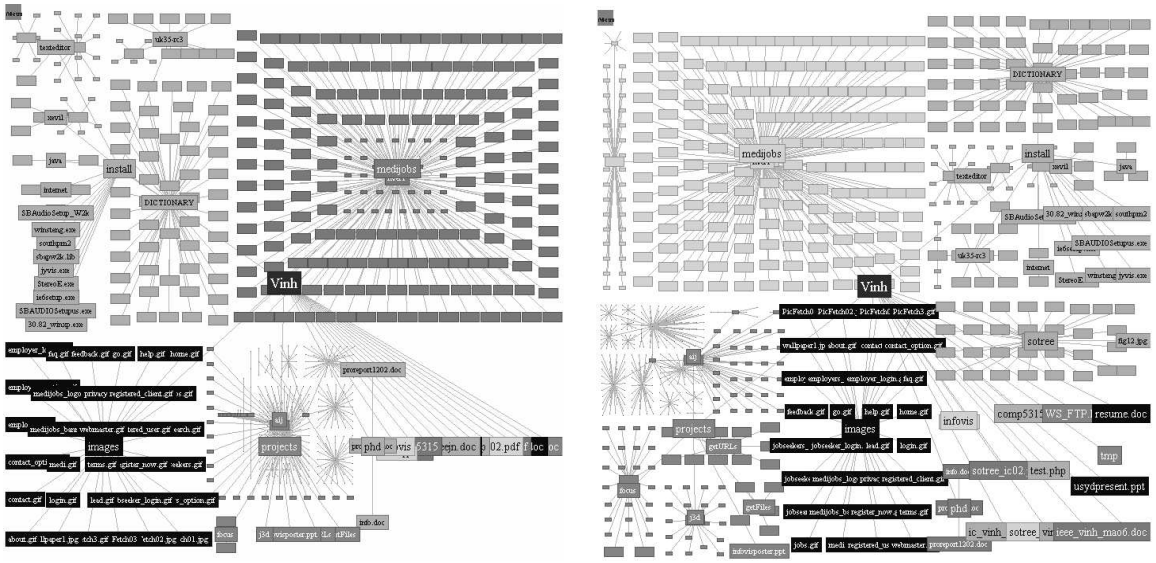


(c) Strip Tree-Maps



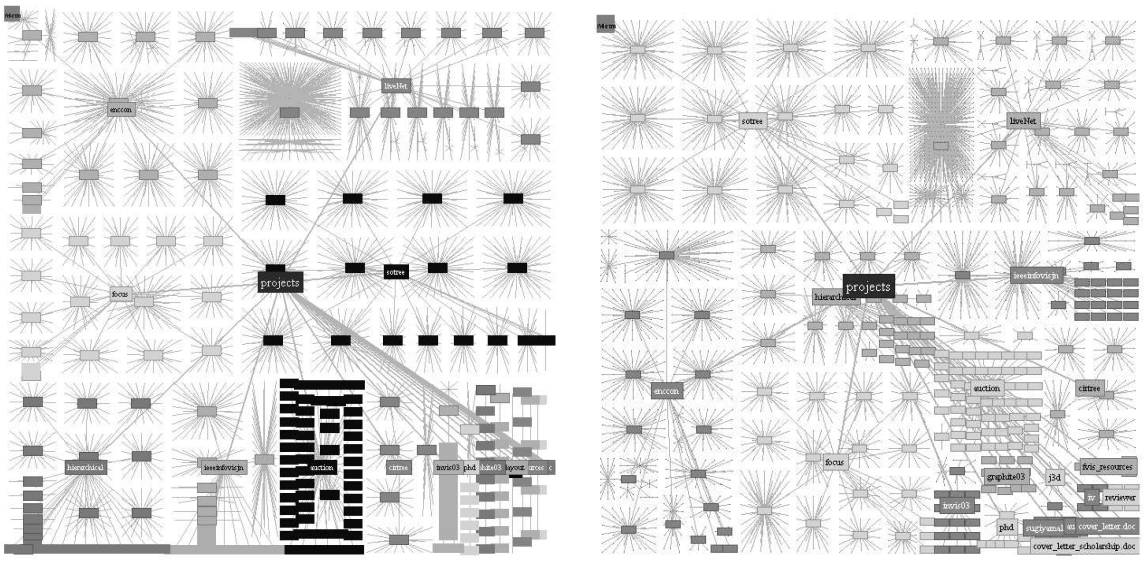
(d) Original Tree-maps

Figure 11. The layouts of the 1st experimental dataset. Image (a) shows a layout using *EncCon*'s partitioning algorithm. Image (b) shows a layout using *Squarified Tree-Maps*' partitioning algorithm. Image (c) shows a layout using *Strip Tree-Maps*' partitioning algorithm. Image (d) shows a layout using the original *Slice and Dice Tree-Maps*' partitioning algorithm.



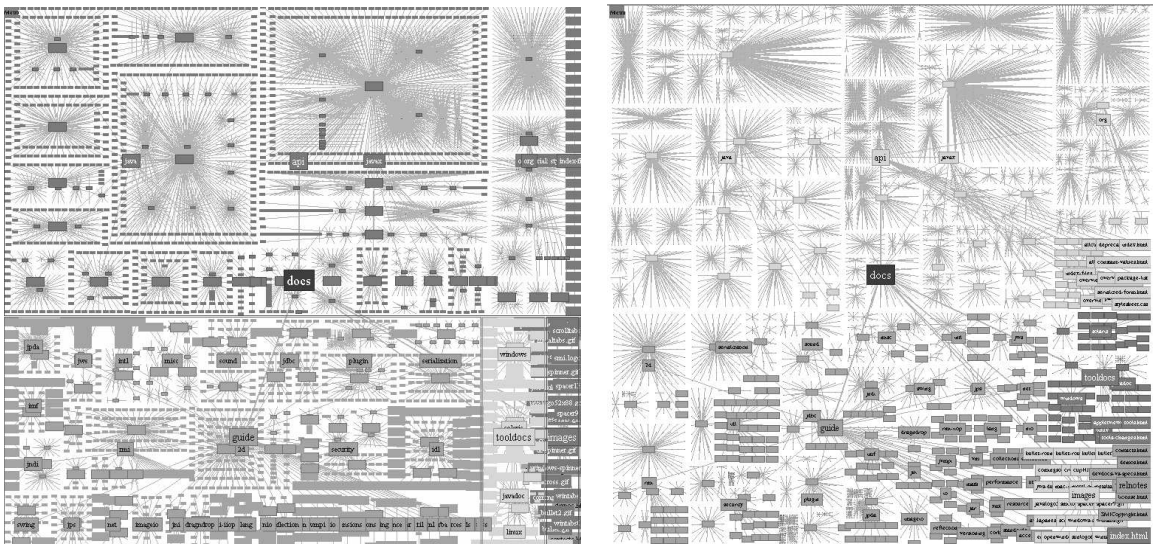
(a) (b)

Figure 12. The layouts of the 2nd experimental dataset: a *Unix* file directory with approximately 900 sub-directories and files that are generated by (a) *EncCon*'s partitioning algorithm and (b) *Squarified Tree-Maps'* partitioning algorithm.



(a) (b)

Figure 13. The layouts of the 3rd experimental dataset: a file system with approximately 3,660 sub-directories and files that are generated by (a) *EncCon*'s partitioning algorithm and (b) *Squarified Tree-Maps'* partitioning algorithm.



(a) (b)

Figure 14. The layouts of the 4th experimental dataset: the entire Java SDK, v.1.4.1 Documentation with approximately 9,500 sub-directories and files that are generated by (a) *EncCon*'s partitioning algorithm and (b) *Squarified Tree-Maps*' partitioning algorithm.

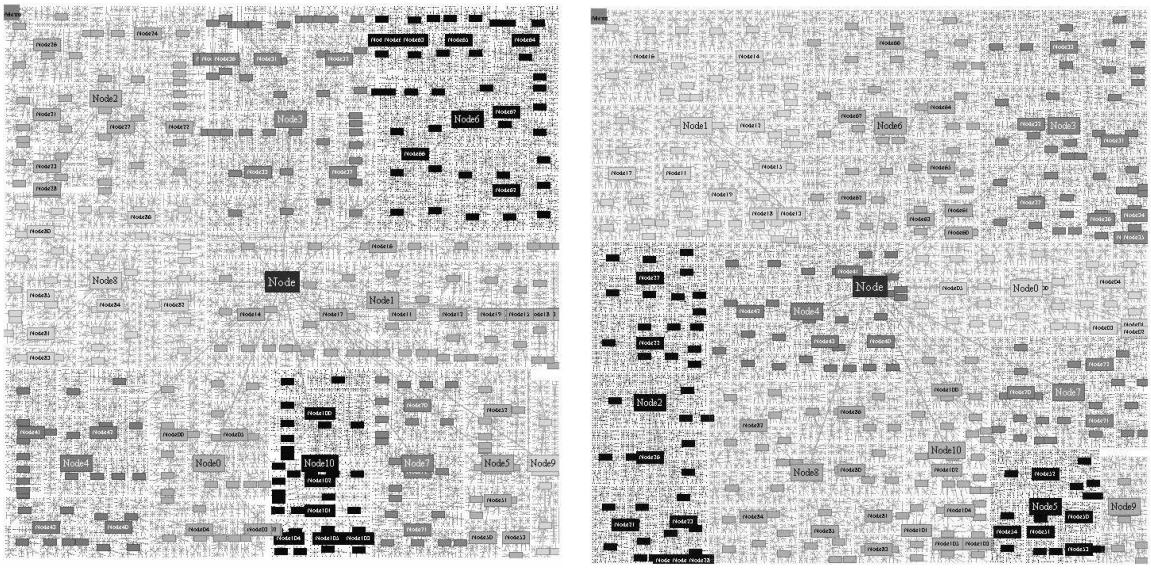


Figure 15. The layouts of the 5th experimental dataset: a very large data with approximately 50,000 nodes that are generated by (a) *EncCon*'s partitioning algorithm and (b) *Squarified Tree-Maps'* partitioning algorithm.

An experimental evaluation of square-like partitioning algorithms

From the comparison of four partitioning algorithms shown in Figure 11, we see that the square-like partitioning algorithms, which include *Squarified Tree-Maps* and *EncCon Partitioning*, can produce better layouts for the placement of small node-link diagrams. Other partitioning algorithms, such as *Strip Tree-Maps* and *Original Tree-Maps*, are not as good in this kind of visualization. Therefore, in this section we only evaluate *Squarified Tree-Maps'* and *EncCon's* partitioning algorithms, and then compare the outcomes of the evaluation against the aesthetics rules defined in graph drawing[42]. This experimental evaluation was carried out with five data sets consisting 26, 900, 3660, 9500 and 50,000 nodes. The layouts of these data sets are presented respectively from Figure 11 to Figure 15. The aesthetics rules we are using for the evaluation include *edge crossings*, *angular resolution*, *total edge length*, and *uniform edge length*. We now evaluate the partitioning algorithms against these aesthetics rules one by one:

Edge Crossings

Edge crossings make it difficult to trace paths and to interpret the relationships. Therefore, all edge crossings should be removed or the number of edge crossings should be minimised. To reduce the complexity of evaluation, we only consider the edge crossings in the top three levels of the hierarchy. This is because that those small graphical edges in the lower levels of the hierarchy can only show the global "density" information of the data sets, rather than particular detailed relationships. They are too small to be seen, and crossings among these small edges are not significant, in terms of interpreting particular relationships. Therefore, we can ignore counting these edge crossings in the lower levels of the hierarchy. The following table shows the actual number of edge crossings occurred in five different layouts generated by *EncCon* and *Squarified Tree-Maps* algorithms.

	Squarified Tree-Maps	EncCon
No. of Crossings in Data Set 1	0	0
No. of Crossings in Data Set 2	32	8
No. of Crossings in Data Set 3	812	276
No. of Crossings in Data Set 4	44	58
No. of Crossings in Data Set 5	8	12

Table 1. Numbers of the Edge Crossings.

We can see from the above table that *EncCon*'s partitioning algorithm produces fewer edge crossings for data sets 2 and 3, while *Squarified Tree-Maps'* partitioning algorithm produces slightly fewer crossings for data sets 4 and 5.

Angular Resolution

The *Angular Resolution* aesthetic rule measures the *Average Angular Variance* of all angles formed by edges of a non-leaf node and its child vertices. Suppose that vertex v_i has k children $\{v_{l+1}, v_{l+2}, \ldots, v_{l+k}\}$ and the angle formed by edges $\{v_iv_{l+1}, v_iv_{l+2}, \ldots, v_iv_{l+k}\}$ are respectively $\{\theta_{l+1}, \theta_{l+2}, \ldots, \theta_{l+k},\}$. The average angle α_i of vertex v_i is calculated by the following formula:

$$\alpha_i = \frac{1}{k}\sum_{j=1}^{k}\theta_{l+j} \quad (7)$$

Then, the *Average Angular Variance* AV_i (in percentage scale) of all the angles $\{\theta_{l+1}, \theta_{l+2}, \ldots, \theta_{l+k},\}$ at vertex v_i is calculated by:

$$AV_i = \frac{100\%\sum_{j=1}^{k} | \theta_{l+j} - \alpha_i |}{k\alpha_i} \quad (8)$$

Ideally, if all angles at vertex v_i are equal then $AV_i = 0$. Finally the overall *Average Angular Variance AV* of the entire tree *T*, which consists of n vertices $\{v_1, v_2, \ldots, v_n\}$ is calculated by the following formula:

$$AV = \frac{1}{n}\sum_{i=1}^{n} AV_i \quad (9)$$

We aim to minimise the value of *AV* for the layout of given trees. The following table shows the Angular Resolution in five different layouts generated by the *EncCon* and *Squarified Tree-Maps* algorithms.

	Squarified Tree-Maps	EncCon
AV Value in Data Set 1	11.08	15
AV Value in Data Set 2	42.97	32.42
AV Value in Data Set 3	43.44	43.35
AV Value in Data Set 4	48.73	45.28

AV Value in Data Set 5	30.71	25.88

Table 2. Angular Resolution.

We can see from the above table that the *EncCon*'s partitioning algorithm produces a better *Angular Resolution* in layouts 2, 3, 4 and 5, while *Squarified Tree-Maps'* partitioning algorithm produces a slightly better *Angular Resolution* in layout 1 of a small data set.

Total Edge Length

To satisfy the *Total Edge Length* aesthetic of graph drawing, we should minimize the sum of the lengths of all edges of the graph. This aesthetic is usually measured through the value of *Average Edge Length AL*. In other words, the average of all edge lengths in optimised graph layouts should be as small as possible.

Suppose that $G = \{V, E\}$ is a graph, where the edge set $E = \{e_1, e_2, \ldots, e_n\}$. The length l_{e_i} of each edge e_i linking between two vertices $v_a(x_a, y_a)$ and $v_b(x_b, y_b)$ can be easily calculated by the following formula:

$$l_{e_i} = \sqrt{(x_a - x_b)^2 + (y_a - y_b)^2} \quad (10)$$

Then the average edge length *AL* of all edges in graph *G* therefore can be calculated by the following formula:

$$AL = \frac{1}{n}\sum_{j=1}^{n} l_{e_j} \quad (11)$$

This aesthetic rule, in conventional graph drawing, is defined based on the assumption that all edges and vertices of a given graph are assigned the same value for their graphical attributes. For example, under this assumption all vertices will use the same style of icons (nodes) with the same size and same background for

their visual presentation, and all edges will use the same graphical line-type with the same thickness and the same brightness for their visual presentation.

However, in *Enclosure+Connection* visualization, the above assumption no longer holds. In *EncCon*, vertices and edges at different levels of the hierarchy are associated with different graphical attributes or different values of the same attributes. Fortunately, all vertices and edges in the same level of the hierarchy are assigned the same graphical attributes and the same value of these graphical attributes.

We believe that it's unfair to count all edges of different levels with different visual effects for the calculation of this single measurement. Therefore, we modify this aesthetic by calculating the AL_0, AL_1, and AL_2 of all edges at different levels, respectively levels 0, 1, and 2, of the hierarchy separately. The following table shows the *Average Edge Length* of five different layouts at the top three levels of the hierarchy, generated by *Squarified Tree-Maps* and *EncCon* partitioning algorithms.

	Level 0 – AL_0		Level 1 – AL_1		Level 2 – AL_2	
	Squarified	EncCon	Squarified	EncCon	Squarified	EncCon
Data Set 1	265.06	263.9	117.25	116.2	0	0
Data Set 2	309.11	300.06	97.6	99.52	105.83	112.08
Data Set 3	339.91	339.86	76.09	87.82	29.59	31.26
Data Set 4	326.91	307.95	155.97	168.44	38.52	48.41
Data Set 5	268.73	279	92.3	92.55	41.29	45.47

Table 3. Average Edge Length (the size of display is: 700x700 pixies).

Uniform Edge Length

Like the *Angular Resolution* aesthetic we described above, the *Uniform Edge Length* aesthetic is for the measurement of the average length variance of all edges of a given graph (or a sub-graph). We aim to

minimise the variance of the lengths of the edges. An optimised graph layout should have all edge lengths as uniform as possible.

Suppose that $G = \{V, E\}$ is a graph, where the edge set $E = \{e_1, e_2, \ldots, e_n\}$ with the corresponding edge lengths of $\{l_{e1}, l_{e2}, \ldots, l_{en}\}$. The length l_{ei} of each edge e_i can be easily calculated by formula 10, and the average edge length AL of all edges in graph G can be calculated by formula 11. Thus, the average length variance ALV (in percentage scale) of all edges in G can be calculated by the following formula:

$$ALV \frac{100\% \sum_{j=1}^{n} | l_{e_j} - AL |}{n * AL} \quad (12)$$

We also calculate three values ALV_0, ALV_1, and ALV_2 for three different levels of the hierarchy separately because of the above reason. The following table shows the *Average Length Variance* of five different layouts at the top three levels of the hierarchy, generated by *Squarified Tree-Maps'* and *EncCon's* partitioning algorithms.

	Level 0 – ALV_0		Level 1 – ALV_1		Level 2 – ALV_2	
	Squarified	EncCon	Squarified	EncCon	Squarified	EncCon
Data Set 1	16.68	16.33	29.78	32.14	0	0
Data Set 2	20.21	16.04	42.27	35.63	46.72	38.92
Data Set 3	24.79	18.56	55.13	40.46	37.57	38.05
Data Set 4	31.27	27.64	66.15	51.14	61.61	58.86
Data Set 5	25.7	23.05	38.78	32.7	40.75	42.61

Table 4. Average Length Variance (the size of display is: 700x700 pixies).

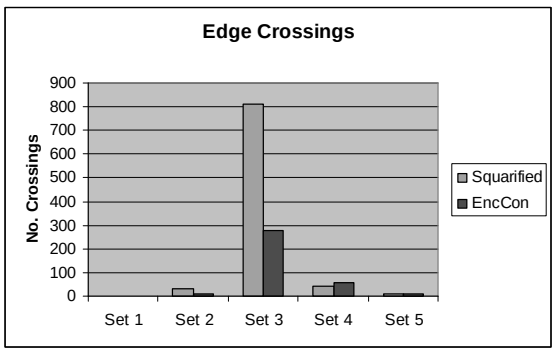
Results of the evaluation

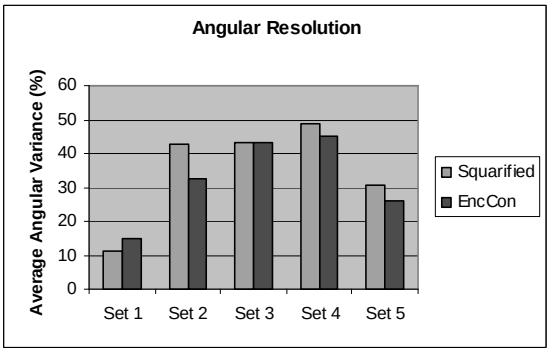
The results of the evaluation based the above criteria are summarized in Figure 16. This figure shows a comparison of the *edge crossings*, *angular resolution*, *total edge length*, and *uniform edge length* between two series of layouts, generated by *Squarified Tree-Maps* partitioning and *EncCon*'s circular partitioning. Five experiments were carried out with five different datasets, covering small, medium, moderately large, large and very large hierarchical data examples (see layouts from Figure 11 to Figure 15).

Overall, from the result of comparisons, we can see that for medium and moderately large datasets *EncCon*'s partitioning performs significantly better than *Squarified Tree-Maps* in terms of minimising edge crossings. The layout of a medium (or moderately large) size graph generated by *EncCon*'s partitioning algorithm contains only approximately 25% of the edge crossings occurred in the layout of the same graph generated by *Squarified Tree-Maps.*

We can also see that the *EncCon* partitioning produces a better *Angular Resolution* in layouts 2, 3, 4 and 5, while *Squarified Tree-Maps* partitioning produces a slightly better Angular Resolution in layout 1 of a small data set.

There is no significant difference between these two algorithms in producing *Average Edge Length.* However, *EncCon* partitioning does significantly reduce the *Average Length Variance* in the layout of moderately large or large graphs.





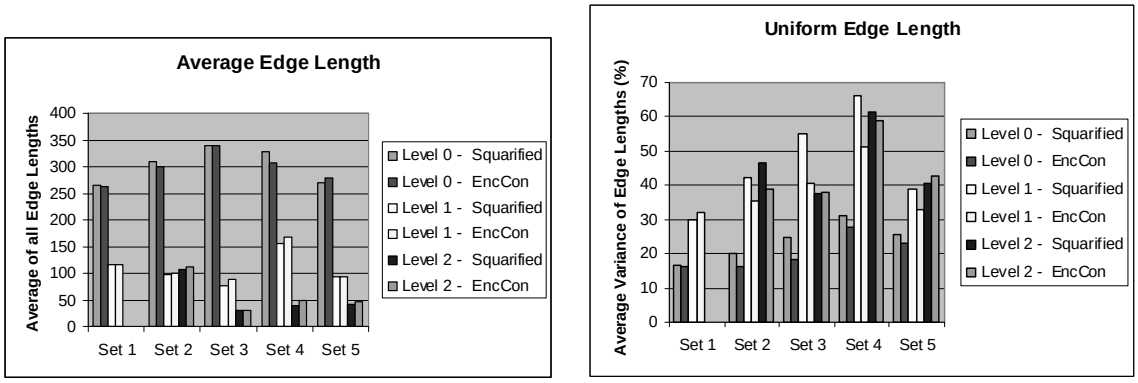


Figure 16. A summary of the experimental evaluation results that were generated with five data sets consisting 26, 900, 3660, 9500 and 50,000 nodes.

Layering Display and View Navigation

Although there have been many proposed innovative focus+context techniques to facilitate the viewing and navigation of medium size information spaces, the exploration of large information spaces remains a challenging task in the design of interactive visualization. Even if the advances in graph drawing research have enabled the efficient calculation of the corresponding geometrical spaces of large graphs and their placements on the screen (see Figures 14 and 15), the retrieval of actual data items through the large visualization is still impossible unless it provides effective navigation mechanisms. For example (as shown in Figure 14), we can use EncCon layout algorithm to display the overall structure of the entire Java SDK Documentation version 1.4.1 in one screen, however, how could we get this visualization to cooperate with the actual information retrieval activities? How could we navigate from this large visualization (with 9,500 nodes) to find a detailed description of a particular file, sub-directory or Java command? Clearly, the key to the successful design of large interactive visualization is navigation. Without providing efficient navigation mechanisms that cooperate with large displays, the visualization technique is useless in the field of information retrieval which aims to assist users to retrieve particular data items they want.

Most existing focus+context techniques require the division of display areas for the display of both the *global view* (or the *context view*) and the *detail view* (or the *focus view*), such as *Information Slices* [43] and

Bifocal Tree [44]. Thus, the available display space left for displaying the global structure (as well as the focus view) is getting smaller (see Figure 3). Other techniques apply multiple windows to display the *global context* and the *focus view* of information [45]. However, the use of separate windows breaks the context (or nature connection) between two views. This costs viewers extra cognitive efforts to link two views into one context.

To be able to efficiently handle the viewing/navigation of large data sets and address the above problem, *EncCon* uses a new *zooming+layering* concept to achieve the focus+context viewing of large hierarchies. This technique is different from the traditional *enlarge+embedded* concept which are used by other focus+context techniques. Technically, it employs a semi-transparency graphical technique to achieve the display of two layers in the virtual z-coordination at the same physical screen. This allows both the *context view* and the *detail view* to be drawn at two separate layers. These layers are then displayed in an overlapped manner at the same physical screen space. The *EncCon* always keeps one view in highlighted and another in de-emphasized state.

The *focus view* is displayed in a full screen with more details for users to see. The system allows the viewer to interactively shift the highlighting by bring the view to the front/back between the *global view* and the *detail view*. This improves the utility of the display space while still providing both the focus and the context information. The viewing technique is independent to the layout algorithm, and thus, it can be applied to other layout algorithms.

The navigation in *EncCon* is achieved interactively by semantic zooming, updating views and swapping views between layers. All these transactions are accommodated by animation to preserve the user's mental map of views.

The transparency techniques have been used in a number of three-dimensional visualization systems, such as the *Information Cube* [39], *Cone-Tree* [15] and the *Spiral Calendar* [40], etc, to solve the overlapping problem in three-dimensional viewing. However, these systems did not use the transparency technique for

navigating abstract graphs in two dimensional spaces. Lieberman [41] also presented a good technique for navigating geometry-related information spaces. However, his research focus is on the navigation of a spatial metaphor in which the data presentation is based on the graphical rendering techniques. We adapted his transparency idea and applied it into the navigation of graph visualization, in which a different visual metaphor "the abstract graph" is used for representing data.

In our visualization, the *global view* is overlapped with the *detail view* and they are displayed in two separate graphical layers of the same physical screen with different visibility values. As the default mode of the display, the *focus view* is drawn in full screen size on layer 1 of the display while the *context view* is drawn in a reduced size at the centre of layer 2. The purpose of this arrangement is to reduce the distraction between two views. The distractions may occur when there are too many overlaps among nodes between two *views*. At the default mode, we assign a *visible* value to layer 1 and a *semi-transparency* value to layer 2. The size of *global view* can be interactively adjusted to suit the user's preference. In our application, the default size of the *global view* is a half of the entire display area.

There are two modes of the display: 'default' and 'context'. In the 'default' mode, we assume that the user's attention is on the content of a particular detailed sub-structure from the entire information structure. However, it is quite possible that a user moves his attention from a detailed sub-structure to the content of the global structure during the navigation. Therefore, we defined a 'context' display mode that shifts the visibility values between two display layers. Practically, we re-assign a *visible* value to layer 2 (the global view) allowing users to see the content of the global structure and turn off the *visible* to *semi-transparency* for layer 1 (the detail view). In other words, in the 'context' mode, the display is reversed so that the *global view* is brought from the back to the front and highlighted and the *detail view* is sent from the front to the back and displayed in a semi-transparent manner. These two views can be shifted interactively by using left mouse-click on the background of each layer. The shifting between views is accommodated by fade in/out animation to preserve the user's mental map of views.

The background of the *context view* is painted with slightly darkened colour in comparison with the *detail view*'s background. This helps the *context view* to stand out from the *detail view*. The selected sub-hierarchy is also highlighted in the *context view* by using a different background colour as well as selected node. This property helps to improve the clarity of the display. The *context view* in this visualization can be either the view of the entire hierarchy or the view of a sub-hierarchy which is previously displayed as a *detail view* before a user selects a particular subset from this context.

In our visualization, the selection of a focus visual node *v* (a sub-hierarchy or an object) is the main mode of interactions taken by users during the navigation. This interaction can be applied to any visible node *v* at the *detail view* or *context view* in order to utilize a semantic zooming technique[22] to enlarge the display of local region *R(v)* into the full screen and bring it to layer 1 as a new *detail view*, which overrides the previous display of layer 1. The system can also return the *context view* back to the *detail view* by a right mouse clicking. The context view then will be enlarged through semantic zooming and can be brought from layer 2 to layer 1 for full screen display.

Application examples

We have applied our layout algorithm and navigation technique to several applications including a visual collaborative system for shared workspaces and a visual product catalog for online auctions. Both Figures 17 and 18 are the screenshots of these two applications. In this section, we will describe the applications of EncCon visualization in shared collaborative environment and the e-commerce.

Application 1: visual display and navigation of shared collaborative workspaces

We have developed a new interactive visualization that appears as an additional window embedded in a collaborative system called *LiveNet* [46]. This visual component can be used for viewing, navigating, editing and manipulating knowledge-based learning information. It employs our *EncCon* technique to handle large scale collaborative information and it aims to provide a better assistance to users of the

LiveNet for visual manipulation and navigation of collaborative objects stored in a relational database. We use a rich set of graphical attributes, such as transparency (visibility), variety of icons, pop-up window, shapes, colours, etc to achieve the display of multiple types of the relationships simultaneously in the same visualization. This visualization component of the *LiveNet* provides user with not only a two-dimensional graphical interface for direct data manipulation, but also a new layering navigation mechanism allowing users to navigate across different contexts of the relational structure. In other words, we allow users to display, view and navigate multiple graphs though the layering visualization concept. This enhances the understanding of complex relational information in a real world situation that the user wants to see. For instance Figure 17 shows an example of such visualization used in shared collaborative workspaces corresponding to a particular user. We can see from Figure 17 that a variety of the graphical icons are used to present different types of objects that are consistent with the original textual interface. The colours of the edge represent different types of the relationships, such as *the classification, the accessing, the participating*, and *the sharing* relationships.

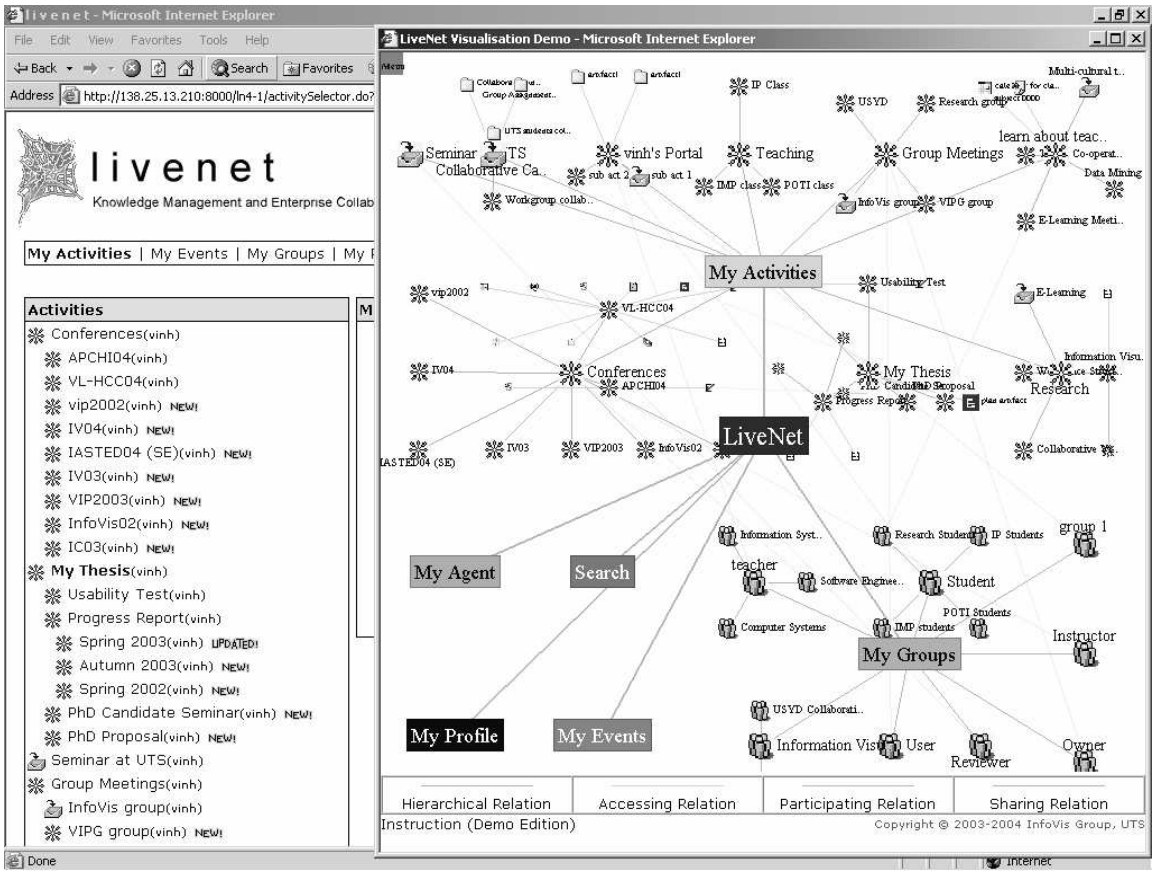


Figure 17. A screenshot of the visual interface for a particular user in *LiveNet*.

Application 2: A visual online auction site

As the second application, we have applied our *EncCon* technique to the online purchasing. We created a prototype of a visual product catalogue for navigating the large-scale auction items for an online auction store, which is similar to the current auction sites such as eBay. The *EncCon* visualization can display the entire product hierarchy and then uses the layering focus+context technique allowing users to navigate through the item catalogue to find out particular auction items they are interested. The system also provides mechanisms to identify special items, so that the user can quickly move to a particular sub-tree of the item hierarchy. We have developed a prototype of such an online auction store to demonstrate the applicability of our visualization technique for browsing large scale product catalogues. Figure 18 shows an example of an online auction site that uses EncCon interactive visualization as its visual interface.

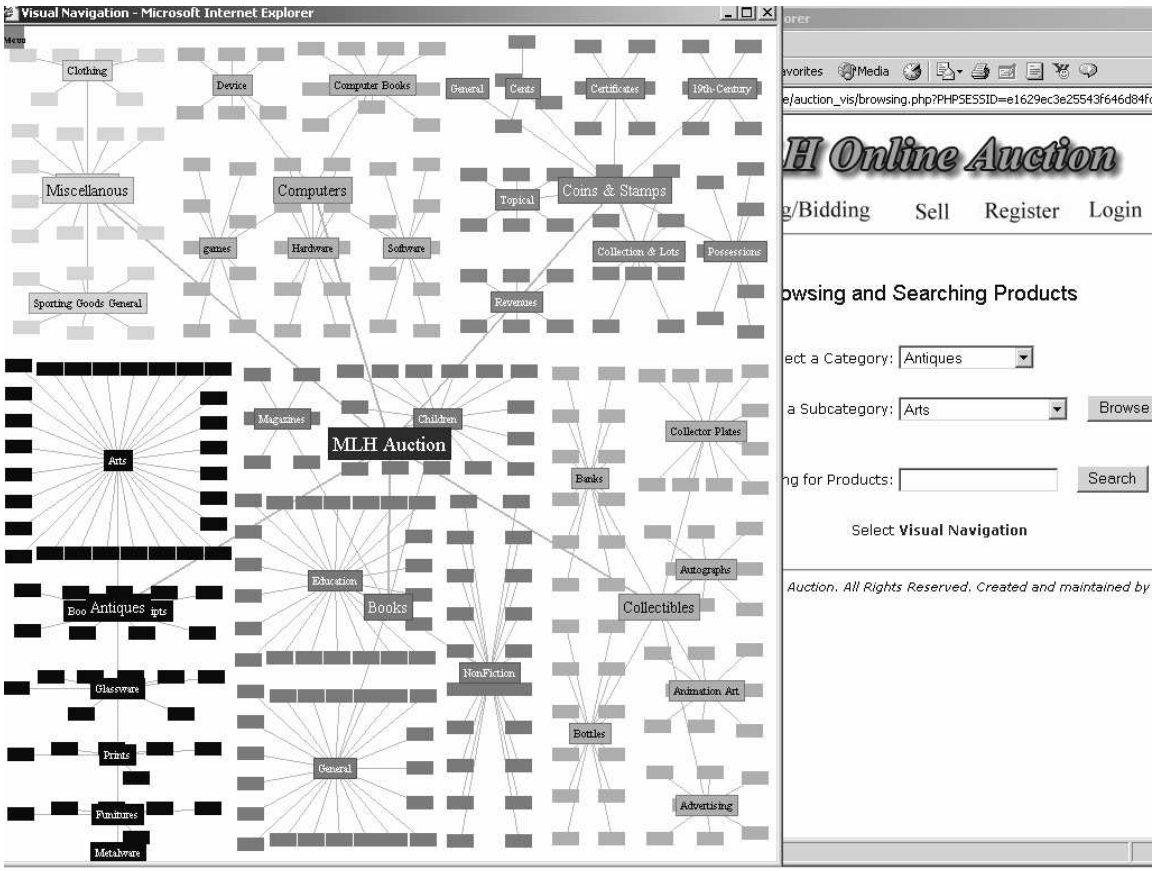


Figure 18. A screenshot of using EncCon interactive visualization as a visual user interface for online auction.

Conclusion

We have presented our *EncCon* approach for visualizing and navigating large hierarchies. We believe that there are three key issues currently raised in the design of effective interactive visualization for manipulating large data sets. They are 1) space utilization 2) navigation mechanisms and 3) minimization of human cognitive process (a clear presentation).

Space utilization becomes significantly important when visualizing large graphs with hundreds or thousands of nodes (and edges) because of the limitation of available screen pixels for display. We don't

want to waste screen pixels for displaying only the background, and we want to utilize up to 100% of the available pixels to display the large context structure (or large hierarchical structure). EncCon addresses this issue by using a rectangular space-filling method for recursively positioning of trees in the display space. The space-filling method used in *EncCon* is similar to *Squarified Tree-Maps* [19]. In comparison with the *SO-Tree* [24] which uses a polygonal space-filling method for the tree positioning, the rectangular space-filling method is generally more straightforward for viewers to perceive hierarchical relationships. Furthermore, the rectangular space-filling algorithm used in *EncCon* is relatively simple and requires less computational time than the polygonal space-filling algorithm.

Navigation of large hierarchies is the second key issue raised in the design of interactive visualization. Even if the advances in graph drawing research have enabled the efficient geometrical positioning of trees and their displays, a visualization of large trees would be useless still, in terms of assisting users to retrieve particular data items, unless it provides efficient navigation mechanisms in collaboration with the display. *EncCon* uses a new *zooming+layering* concept to achieve the focus+context viewing of large hierarchies, rather than the traditional *enlarge+embedded* approach which is used by most of the existing focus+context techniques. Technically, it employs a semi-transparent graphical technique to achieve the concurrent display of two separate layers of information on the same physical screen. This alternative approach enlarges the display areas for displaying *context view* and *detailed view* and allows more information to be presented in both views. The navigation in *EncCon* is achieved interactively by semantic zooming, updating views and swapping views between layers. All these transactions are accommodated by animation to preserve the user's mental map of the views.

The third issue we attempted to address is to minimize the human cognitive process involved in identifying the relational structure of large graphs. The large graphs usually contain hundreds or thousands of relationships, and therefore a clear presentation of such large numbers of relationships is essential for easy understanding and avoidance of confusion in viewing such complex relational structures. To be able to enhance the perception of large relational structures for human understanding, EncCon still uses a node-link diagram to show the relationships explicitly in addition to the space-filling approach used for

addressing the space utilization issue. Furthermore, to reduce the human cognitive cost, we adopt a rectangular space-filling approach, rather than a polyclonal space-filling approach for the positioning of graphs.

Because it addresses all three issues presented above, hence we believe that *EncCon* is an effective approach to creating interactive visualization of large tree structured relational data. However, there are still some problems that need to be solved. For example, we need to solve the problem of overlapping among nodes between layers even if it is not very significant since layer 2 is displayed in a semi-transparent manner.

In the future, we will investigate optimised graphical techniques and layout algorithms to minimise the effect of our layered display in human cognition processes. Although the layering navigation technique is independent of the layout, it works better with optimised layouts of the node-link diagram. We believe that by adopting optimised layout algorithms, the overlap among nodes between the *context view* and the *detail view* will become insignificant.

References

1 Ware C. *Information Visualization: Perception For Design*. Morgan Kaufmann Series in Interactive Technologies, Morgan Kaufmann: San Francisco, 2000; 438pp.

2 Chen C, *Information Visualization and Virtual Environments*. Springer-Verlag: London, 1999; 223pp.

3 Reingold EM and Tilford JS. Tidier Drawing of Trees. *IEEE Transactions on Software Engineering* 1981; **7**(2): 223-228.

4 Johnson B and Shneiderman B. Tree-Maps: A Space-Filling Approach to the Visualization of Hierarchical Information Structures. *the 1991 IEEE Visualization* 1991 (Piscataway, NJ, USA), IEEE Computer Society; 284-291.

5 Shiloach Y. Arrangements of Planar Graphs on the Planar Lattices. *Ph.D Thesis*, Weizmann Institute of Science, Rehovot, Israel, 1976.

6 Eades P. Drawing Free Trees. *Bulleting of the Institute of Combinatorics and its Applications* 1992; 10-36.

7 Melancon G and Herman I, *Circular Drawings of Rooted Trees, Technical Report of the Centre for Mathematics and Computer Sciences 1998, INS-R9817,* Amsterdam, The Netherlands.

8 Chi EH, et al. Visualizing the Evolution of Web Ecologies. *CHI'98 Conference on Human Factors in Computing Systems* 1998 (Los Angeles, California, USA), ACM Press; 400-407, 644-645.

9 Wills GJ. NicheWorks - Interactive Visualization of Very Large Graphs. *Journal of Computational and Graphical Statistics* 1999; **8**(2): 190-212.

10 Teoh ST and Ma K-L. RINGS: A Technique for Visualizing Large Hierarchies. *Graph Drawing* 2002 (Irvine, California, USA), Springer; 268-275.

11 Hendley RJ, et al. Narcissus: Visualizing information. *Information Visualization '95 Symposium* 1995 (Atlanta, GA, USA), IEEE; 90-96.

12 Plaisant C, Grosjean J, and Bederson BB. SpaceTree: Supporting Exploration in Large Node Link Tree, Design Evolution and Empirical Evaluation. *IEEE Symposium on Information Visualization (InfoVis 2002)* 2002 (Boston, MA, USA), IEEE; 57-64.

13 Lamping J and Rao R. The Hyperbolic Browser: A Focus + Context Technique for Visualizing Large Hierarchies. *Journal of Visual Languages and Computing* 1996; **7**(1): 33-55.

14 Munzner T. Exploring Large Graphs in 3D Hyperbolic Space. *IEEE Comp. Graphics & Applications* 1997; **18**(4): 18-23.

15 Robertson GG, Mackinlay JD, and Card SK. Cone Tree: Animated 3D Visualizations of Hierarchical Information. *ACM SIGCHI conference on Human Factors in Computing Systems '91* 1991 (New York, USA), ACM Press; 189-194.

16 Kleiberg E, van de Wetering H, and van Wijk JJ. Botanical Visualization of Huge Hierarchies. *IEEE Symposium on Information Visualization (InfoVis'01)* 2001 (San Diego, CA, USA), IEEE Computer Society; 87-94.

17 Cheswick B, Burch H, and Branigan S. Mapping and Visualizing the Internet. *2000 USENIX Annual Technical Conference* 2000 (San Diego, CA, USA), USENIX; 1-12.

18 van Wijk JJ and van de Wetering H. Cushion Treemaps: Visualization of Hierarchical Information. *IEEE Symposium on Information Visualization (InfoVis'99)* 1999 (San Francisco, California, USA), IEEE Computer Society; 73-78.

19 Bruls M, Huizing K, and van Wijk JJ. Squarified Treemaps. *joint Eurographics and IEEE TCVG Symposium on Visualization* 2000 (Vienna, Austria), Springer; 33-42.

20 Stasko J and Zhang E. Focus+Context Display and Navigation Techniques for Enhancing Radial, Space-Filling Hierarchy Visualizations. *IEEE Symposium on Information Vizualization (InfoVis'00)* 2000 (Salt Lake City, Utah, USA), IEEE; 57-65.

21 Yang J, Ward MO, and Rundensteiner EA. InterRing: An Interactive Tool for Visually Navigating and Manipulating Hierarchical Structures. *IEEE Symposium on Information Visualization (InfoVis'02)* 2002 (Boston, Massachusetts, USA), IEEE Computer Society; 77-84.

22 Herman I, Melancon G, and Marshall MS. Graph Visualization in Information Visualization: a Survey. *IEEE Transactions on Visualization and Computer Graphics* 2000; **6**: 24-44.

23 Bederson BB, Shneiderman B, and Wattenberg M. Ordered and Quantum Treemaps: Making Effective Use of 2D Space to Display Hierarchies. *ACM Transactions on Graphics (TOG)* 2002; **21**(4): 833-854.

24 Nguyen QV and Huang ML. Space-Optimized Tree: a Connection+Enclosure Approach for the Visualization of Large Hierarchies. *Information Visualization Journal (Palgrave Macmillan)* 2003; **2**(1): 3-15.

25 Song H, Curran EP, and Sterritt R. Multiple Foci Visualisation of Large Hierarchies with FlexTree. *Information Visualization Journal (Palgrave Macmillan)* 2004; **3**(1): 19-35.

26 Furnas GW. Generalized Fisheye Views. *SIGCHI '86 Conference on Human Factors in Computing Systems* 1986 (Boston, Massachussetts), ACM Press; 15-22.

27 Sarkar M and Brown MH. Graphical Fisheye Views. *Communications of the ACM (CACM)* 1994; **37**(12): 73-84.

28 Kadmon N and Shlomi E. A Polyfocal Projection for Statistical Surfaces. *the Cartographic Journal* 1978; **15**(1): 36-41.

29 Spence R and Apperley MD. A Bifocal Display Technique for Data Presentation. *of Eurographics' 82* 1982; 27-43.

30 Mackinlay JD, Robertson GG, and Card SK. The Perspective Wall: Detail and Context Smoothly Integrated. *ACM Computer-Human Interaction '91 Conference on Human Factors in Computing Systems* 1991 (New York, USA), ACM Press; 173-179.

31 Jog NK and Shneiderman B. Starfield Visualization with Interactive Smooth Zooming. *IFIP 2.6 Visual Database Systems* 1995 (Lausanne, Switzerland), Chapman & Hall; 3-14.

32 Perlin K and Fox D. Pad: An Alternative Approach to the Computer Interface. *20th International Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH '93)* 1993 (New York, USA), ACM Press; 57-64.

33 Bederson BB, et al. Pad++: A Zoomable Graphical Sketchpad For Exploring Alternate Interface Physics. *Journal of Visual Languages and Computing* 1996; **7**(1): 3-32.

34 Bederson BB, Grosjean J, and Meyer J. Toolkit Design for Interactive Structured Graphics. *Transactions on Software Engineering* 2004; **30**(8): 535-546.

35 Huang ML, Eades P, and Cohen RF. On-line Animated Visualization of Huge Graphs Using a Modified Spring Algorithm. *Journal of Visual Languages and Computing* 1998; **9**(6): 623-645.

36 North SC. Incremental Layout in DynaDAG. *Graph Drawing (GD '95)* 1996 (Passau, Germany), Springer; 409-418.

37 Brandes U and Wagner D. A Bayesian Paradigm for Dynamic Graph Layout. *The Symposium on Graph Drawing (GD'97)* 1997 (Rome, Italy), IEEE Computer Society; 236-247.

38 Huang ML, Eades P, and Cohen RF. WebOFDAV - Navigating and Visualizing the Web On-Line with Animated Context Swapping. *WWW 7 / Computer Networks* 1998; **30**(1-7): 638-642.

39 Rekimoto J and Green M. The information cube: Using transparency in 3D information visualization. *Third Annual Workshop on Information Technologies & Systems (WITS'93)* 1993; 125-132.

40 Mackinlay JD, Robertson GG, and DeLine R. Developing Calendar Visualizers for the Information Visualizer. *ACM Symposium on User Interface Software and Technology 1994 (UIST '94)* 1994 (Marina del Rey, CA, USA), ACM; 109-118.

41 Lieberman H. Powers of Ten Thousand: Navigating in Large Information Spaces. *ACM Symposium on User Interface Software and Technology* 1994 (Marina del Rey, CA, USA), ACM; 15-16.

42 Di Battista G, Eades P, Tamassia R, and Tollis IG. *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice Hall: New Jersey, 1999; 397pp.

43 Andrews K and Heidegger H. Information Slices: Visualising and Exploring Large Hierarchies using Cascading, Semi-Circular Discs. *Late-Breaking Hot Topics - IEEE Symposium on Information Visualization (InfoVis'98)* 1998 (Research Triangle Park, North Carolina, USA), IEEE; 9-12.

44 Cava RA, Luzzardi PRG, and Freitas CMDS. The Bifocal Tree: a Technique for the Visualization of Hierarchical Information Structures. *Workshop on Human Factors in Computer Systems (IHC2002)* 2002 (Fortaleza, Brazil).

45 Robert JC. On Encouraging Multiple Views for Visualisation. *International Conference on Information Visualisation (IV 1998)* 1998 (London, UK), IEEE Computer Society; 8-14.

46 Hawryszkiewycz I. Workspace Networks For Knowledge Sharing. *Fifth Australian World Wide Web Conference (AusWeb99)* 1999 (Sydney, Australia); 219-227.