Maintaining the Mental Map for Circular Drawings

Michael Kaufmann[1] and Roland Wiese[1,2]

[1] Wilhelm-Schickard-Institut für Informatik, Universität Tübingen, 72076 Tübingen
`mk@informatik.uni-tuebingen.de`
[2] yWorks GmbH, Sand 13, 72076 Tübingen
`Roland.Wiese@yworks.de`

Abstract. In this paper we present new ideas to extend the framework for circular drawing of networks by Six and Tollis [15] by some new concepts which makes the framework suitable for user interaction. The original approach displays each biconnected component in a circular way, and the blocktree of the graph as a tree drawn radially [9]. We introduce the concept of hicircular drawings, a hierarchical extension of the mentioned framework replacing the circles of single vertices by circles of circular or star-like structures. This concept is inspired by the works of Brandenburg on graph clustering, and the recursive concepts of series-parallel graphs, PQ- resp. SPQR-trees.

1 Introduction

Next to orthogonal layout, Sugiyama-style layout and springembedder-style layout, the circular layout style has gained increasing popularity.

The original circular layout style assumes that the graph is partitioned into clusters, the nodes of each cluster are placed on the circumference of an embedding circle, and each edge is drawn with a straight line [16]. It has found more and more applications in network design and development, especially in the area of computer networks and telecommunication, but also in social networking [12].

The layout style has been introduced by Madden et. al [11,7] to the graph drawing community, the method used is a comparably simple heuristic. Six and Tollis [15,16] adapted the style but developed a scientific framework for the approach. The main idea is to display each biconnected component in a circular way, and the blocktree, i.e. the tree of biconnected components, of the graph as a radial tree of such circles. Other works on clustering include the approaches by Brandenburg [2,10], where a certain kind of subgraphs (cliques) are collapsed to nodes of a graph (cycle, tree) on a higher level of abstraction. An overview on clustering methods as well as on drawing clusters can be found in [4].

On the layout side, the basic concept of compound graphs [18] has been proposed to handle clusters. Compound graphs and also clustered graphs have two different representations, an adjacency representation and also an inclusion representation. The latter one is either done by an additional view of the inclusion tree, or a representation of the cluster by nested rectangles. See also [8] for a

© Springer-Verlag Berlin Heidelberg 2002

more advanced approach. Such a twofold representation is not very flexible, since it distracts the users attention, if it is not carefully used. Therefore we try to display clusteral structures, and also want to show the way a graph develops in a dynamic setting. While in the area of computer networks the graphs are mostly static, we recently met applications from the area of product management, where the structure is dynamically growing.

The following figure shows the need of an interactive extension of the concept of Six and Tollis: First the graph consists of a chain of quadrangles, which are displayed nicely by the layout algorithm “Circular with Radial”. When we add just a single edge connecting the two outermost quadrangles, all the components collaps into one. Circular with Radial places the vertices on a circle, performs a crossing minimization heuristics but cannot avoid at least $n/4$ crossings, one from each of the quadrangles. Unfortunately, the angles that arise are very tiny such that the previous structure can hardly be recognized. An alternative layout for this graph is given in the next figure, it shows the complete ring of components as it arose from the construction. Since one can assume that the user has this structural view of the closed chain of quadrangles in mind, his mental map is somewhat maintained. Note that there is already extensive work been done on dynamic graph drawing and various concepts of mental maps. For an overview and related references see [3].

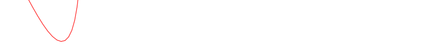

Fig. 1. The left figure shows the chain-like layout as constructed by the Six/Tollis' algorithm. Adding a new edge from the lowest to the topmost node leads to the middle layout, destroying the mental map completely. A better alternative is shown to the right, where the mental map of the structure, namely a circle of smaller circles, is nicely maintained.

Regarding the example, the ring of quadrangles can be viewed as one big component with some subcomponents, which finally leads to a hierarchical concept. In section 3, we will give the ideas for a new more general approach, and explain the necessary operations that lead to hopefully suitable layouts even in a dynamic setting where vertices and edges can be inserted and deleted. We will keep the description on a rather sketchy level.

First, we review the framework for circular drawings by Six and Tollis.

2 The Framework of Six and Tollis and Its Limitations

The framework of Six and Tollis can be summarized as follows: The graph $G = (V, E)$ is decomposed into the biconnected components which form the resulting block tree. For the articulation nodes v which are not adjacent to a bridge between two components, v is assigned to one of the components. The nodes of each single component are placed on a circle such that the distance between neighboring nodes is fixed. This determines the circumference of each component, which is represented as a single supernode of large size in the following. Next an adaption of the algorithm for radial tree layout [9] taking into account the supernodes, which usually have different sizes, gives the final layout of the whole structure. In radial tree layouts, the non-root nodes are placed in concentric circles around the root. Note that some rotations are also necessary in order to obtain a nice final drawing and that the maintenance of a reasonable connection between two adjacent components is not trivial, but can be solved. Further note that the circular drawing of each single component can be enhanced by various techniques for crossing reduction. For more details on the crossing reduction step as well as on the framework see [14,15].

We have already seen in the introduction, that the concept of Six and Tollis is static, that and features that support user interactivity are not incorporated. The whole scheme works in only two major stages, with a strong focus on the biconnectivity structure, since the superstructure of the blocktree is drawn as a radial tree and the single biconnected components as circles. Furthermore, the drawing of single biconnected components is quite limited, the placement of the nodes on a circle hides structural properties like symmetries, on the other hand, the contingent property of outerplanarity can be displayed [14].

We try to develop a more structured layout of biconnected components especially if it consists of several finer structures, without loosing the benefits of the concept of Six and Tollis. Before we discuss the setting for interactivity, we introduce a slightly more complicated variant for static circular layout, but which provides more flexibility than the previous approach. Later on, we will extend this new approach to support user interactiveness.

3 The New Approach

The main idea is to extend the radial-circular framework by the ability of maintaining the mental map. The layout of the different parts will either stay unchanged, or just stretched or squeezed. The embedding will definitely not be changed. Only essential structural changes should be displayed.

3.1 Our Static Variant of the Approach of Six and Tollis

As before, the idea is to take the block tree of biconnected components and layout the root component R in a circular way. Then *recursively* layout the subtrees T adjacent to the root R. The layouts of the subtrees are approximately represented by round supernodes of a size according of the size of the layout. To find appropriate positions for these supernodes, the available space around the layout of R is partitioned by straight lines starting from the center of the layout of R. These lines define certain sectors, the so-called wedges [15]. The size of the wedges is chosen appropriate to the size of the supernodes, while the position of the wedge relative to the layout of R is chosen such that afterwards the layout of T, here represented by a supernode, can be attached directly to its articulation point in R. This recursive concept gives a lot of flexibility as will be shown later.

Note that R can also be a single node, i.e. if the input graph is a tree. Complete binary trees give also a good motivation of our variant, since the constructed layout is the well-known H-tree layout [13] which is linear with respect to the area. The original radial-tree construction covers an area of $O(n^2)$ in this case.

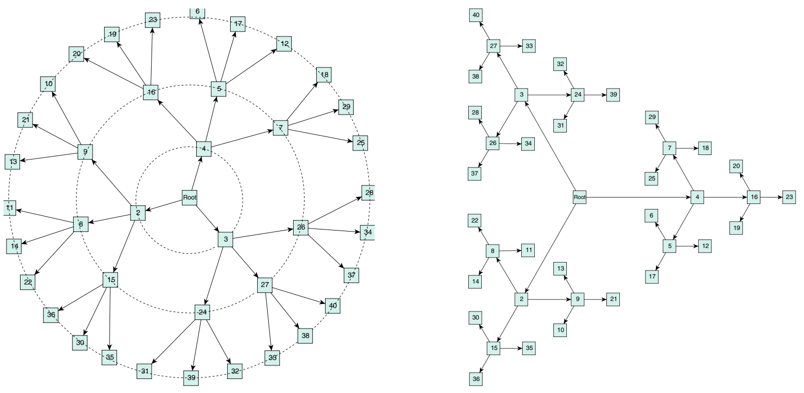

Fig. 2. A complete ternary tree laid out according to the radial-circular algorithm [15] and according to our alternative method where the subtrees are drawn recursively.

So, compared to the algorithms of Six and Tollis, the construction works very similar and also allows a lot of variations like the application of crossing minimization schemes. The decomposition into biconnected components form the block tree, the biconnected components are mapped to circles. The main difference lies in the interpretation of the (block) tree structure, which we now consider to be a recursive structure. For the subtrees of the root, we draw independent layouts of the same layout style which are then connected to the articulation points of the root components.

3.2 Detailed Hierarchical Structures

Next, we present the new concepts for refined structures. Intuitively, we allow to replace the nodes in the scheme above by supernodes that again represent tree-like or circular structures. We simulate the tree-like structures by the simpler concept of *stars*. A star is a graph with a center component and some leg components adjacent to the center, but not connected otherwise.

Each supernode represents an associated underlying subgraph and has a round shape of a size according to the layout size of the associated subgraph. We have two types of supernodes, namely *star nodes* and *circle nodes*. The underlying structure of a star node is a star, consisting of a circle node as center component and possibly a further (super)node at each leg of the star. A circle node represents a set of (super)nodes arranged in a circular way.

This way, the hierarchical structure of the input graph G can be represented by a tree H consisting of star nodes, circle nodes and normal nodes at the leaves of the tree. We call H the *hicircular tree*. A convention of H is that the center component of each star node will always be the leftmost child of the corresponding star node in the tree.

The elements of the smallest edge subset in E that can be represented by the same hicircular tree as the whole graph G are called *structural edges*. All other edges do not contribute to the hierarchical structure, they are *nonstructural*. In Figure 3, we give a graph and two possible representations by hicircular trees, depending on the interpretation of the structure. In this example, all the edges are structural.

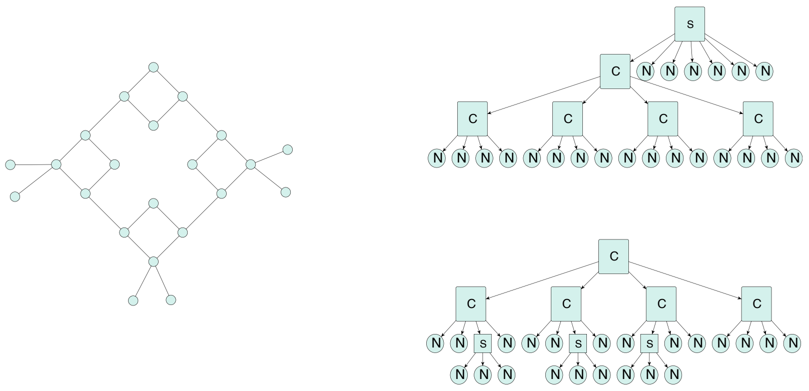

Fig. 3. A graph and two alternatives for its hicircular tree. Circular nodes are denoted by C, star nodes by S and normal nodes by N. In the upper tree, the global structure is a star with a big circular component in the center and six little legs, while the lower structure is a circle, and the legs are on a lower hierarchy level.

Note that similar descriptions of hierarchies have been used in various other concepts like series-parallel graphs [13], PQ-trees [1] or SPQR-trees [6].

For the block tree of biconnected components as in the approach of Six and Tollis, the hicircular tree consists of a tree of star nodes, each with a circle node as the leftmost child (the center) and more star nodes or normal nodes as the other children. All children of circle nodes are normal nodes, since circle nodes just represent biconnected components, which are not differentiated any further.

From a given hicircular tree, it is natural to derive the corresponding *hicircular layout.* The layout technique for the underlying structure of a star node is the same as described in the last subsection. For a circle node we arrange the underlying (super)nodes in a circular way, similar as described by Six and Tollis, but having nodes of different sizes. The details are omitted.

3.3 The Dynamic Case

Based on our concepts developed so far, we use the hicircular tree for the actual graph G to represent its hierarchical structure.

Our objective is to preserve the tree-like and cyclic structures, that we kept as the mental map of the user over time if possible. To achieve this goal we want to maintain the hierarchical structures as much as possible, keep the original layout as recognizable as possible, and still clearly distinguish between tree and cycle structures.

We will now discuss the elementary update operations; more complex operations can be formulated by a concatenation of these elementary operations:

1. Append a new node v to another node w by a new edge $e = (v, w)$ We consider the normal node w in the hicircular tree. w lies either on a circle, or it is the center of a star, or a normal node. In the two latter cases, v is appended as a leg to w. If w is a circle node, we change w's status to 'star node', such that it becomes the center of a star and gets the single circle node v as a leg.

The actual layout is changed as follows: We determine the position of w, place v next to w and connect both nodes with edge e without changing the embedding besides the new edge. Then we run the layout algorithm again for the structural edges of G. This gives a layout where v is inserted as a 'leg' of w. The remaining layout does not change considerably. The actual wedges around w are squeezed a bit, and some space for the wedge for v has been created. v has been placed as close as possible (minimal distance) to w within this wedge. Then the nonstructural edges are added as straight lines. Note that the complete relayout of the graph is not really necessary but it keeps the scheme simple. Alternatively, the layouts within the new smaller wedges can be squeezed by a affine transformation of the node positions.

2. Add a new edge $e = (v, w)$. We consider the position of the single nodes v and w in the hicircular tree H, and determine the lowest common ancestor a in H. If a is a circle node, v and w lie already on a common cycle, which is displayed by supernode a. Hence we decide that e has no effect on the structure of G, e is nonstructural. The drawing for the old graph will not change at all, only the new edge will be inserted, possibly by a straight line.

If a is a star node, the new edge joins two leaf nodes within H, such that a new node cycle $v = v_r, \ldots, v_0 = a = w_0, \ldots, w_s = w$ is formed, where $v_i (0 < i < r)$ are nodes on the path from a to v and $w_j (0 < j < s)$ are nodes on the path from a to w. We now remove all edges from H that lie on that newly formed circle and add a new circle node c to H. Then we relink all nodes on the circle such

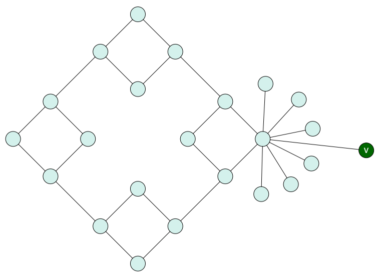

Fig. 4. Appending the new black node to a center of a star node S means a squeezing of the other wedges of the subgraphs of S.

that they become children of c. Finally we remove the edge that connects a with its parent node p and add the edge (p, c) instead. This transformation yields a well-formed hicircular tree again. If a had more than two legs as it is the case in Figure 5, they remain the legs of their centers as before.

From the new hicircular tree, a corresponding new layout can be computed. Compared to the previous layout, the new layout might considerably change, although the planar embedding of the structural edges remains the same except of the new edge. The new edge is embedded according to the previous tree structure, it might only cross some nonstructural edges. The supernodes on the new cycle will move such that a circular structure arise. The adjacent wedges have to be squeezed or even stretched depending on the embedding of the legs of the star nodes, that are involved in the operation. Note that some of the subgraphs associated to the legs of the stars nodes of the new circle will be located inside of the circle, others on the outside. Local adjustments help in this case again.

3. Delete a single node v by removing the connecting edge (v, w). This is exactly the reverse operation of operation **1.** Eventually, we have to change the state of w from 'star node' to 'circle node'. Furthermore, we have the freedom to widen the wedges around w, since the wedge for v could be removed. There is only a slight effect on the drawing of G.

4. Delete an edge $e = (v, w)$. We assume that v, w lie in the same biconnected component. Otherwise, the operation is easy, since the layout is split into two components. We omit this case here. As in operation **2.** we consider the lowest common ancestor a of v and w and determine, if the removal of e breaks a structural cycle C (a cycle which is displayed by a circle node). Otherwise, e is just removed from the drawing, while everything else remains unchanged. If cycle C is broken, the corresponding supernode a is split into two legs that contain either v or w but not both. The supernodes of the legs are again modeled as sequences of star nodes, similar to the original construction for the block tree. The effect is just reverse to that of operation **2.**, cf. Figure 6.

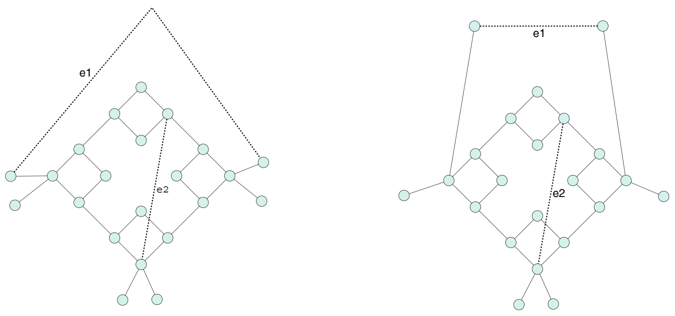

Fig. 5. Inserting two edges to the graph from Figure 3. Edge $e2$ does not change the structure, so it is just drawn straightline across the drawing. Edge $e1$ closes a cycle between two legs of a star. Note that this is only the case when our interpretation of the structure and its representing hicircular tree is the same as in the upper alternative of Figure 3. In this case, a new cycle has been established and we change the structure of the drawing accordingly.

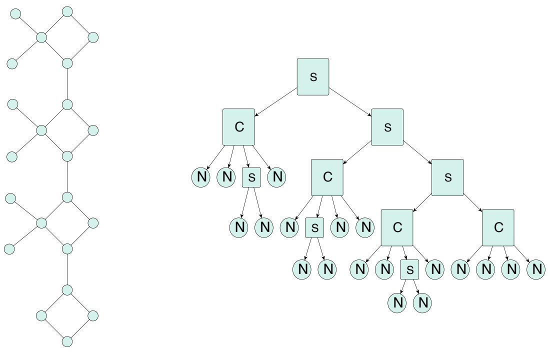

Fig. 6. The deletion of an edge from the outermost circle (Figure 3) shows a drastic effect of an edge deletion on the drawing and the hicircular tree

4 Extensions

Our scheme that we presented here is efficient and it contains a lot of flexibility. We will comment in this section on some variations:

- Assume that we have inserted already many nonstructural edges, such that the more detailed structure cannot be seen anymore just because of the large number of edges. Then the scheme allows to collapse the refinement of the structure and then to display only the coarse structures, just the way as shown in Figure 7.
- If a cycle is broken by the deletion of a structural edge, and there are additional nonstructural edges at present, they might become structural. The scheme now allows to just break the cycle and leave all the other edges as they are, or to remove the nonstructural edges at once and insert them

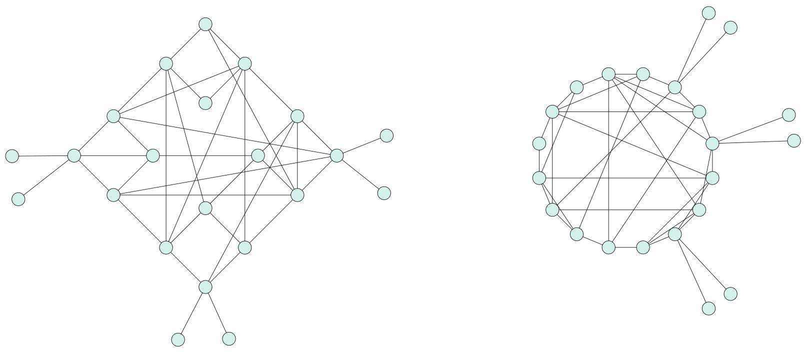
Fig. 7. The center component on the left-hand side with the finer resolution contains too many nonstructural edges such that the refined structural resolution is of no help anymore. Hence the user can decide to collapse a hierarchy level and draw the whole component on one circle as shown in the right figure.

again in the same order as before. This way, one of the nonstructural edges might become structural and the cycle or part of it might remain unbroken. This clearly is an expensive way of restructuring. There might be ways for improvements.
- If a nonstructural edge is inserted, we have originally proposed to draw it straightline across the whole structure. Very often, there are much better alternatives like splines, and/or a clever rerouting of such an edge helps, especially if the drawing is not too crowded yet and edge overlappings can (largely) be avoided.

5 Concluding Discussion

We have presented ideas for an extension of the scheme for circular layouts proposed by Six and Tollis to support the display of fine-granular structures and maintain the mental map while dynamically changing the graph. We kept the description on a high level, our goal was to present the main concept, obviously, a lot of details have been omitted. The tool we used to represent our hierarchical structure, was inspired by series-parallel graphs, PQ-trees or SPQR-trees. The basic components are trees, which again are represented by stars, and cycles, which are represented by circles. The update operations on the graph are presented as elementary operations 'insert/delete single nodes/edges'. More complex operations can be simulated by splitting them. At several steps of our descriptions, we have advised a complete recomputation of the layout. This is mainly due a simple description of the scheme, more efficient alternatives can be thought.

A major property of the approach is that the drawing reflects the way the graphs have evolved. So, the mental map of the user who is focused on tree-like and cyclic structures is maintained. But the drawings of the static graphs without having the history of operations, or the hicircular tree, might not so easy to be understood, since the layout heavily depends on the order of the operations. This effect is not particular due to our algorithm but because of the nature of dynamic algorithms that always try to reflect the evoluation of a structure over time [3].

As already been shown in Figure 2, the hicircular tree is not unique. Hence later operations like *Inserting an edge* might have different effects, as it has been shown in figure F. Fortunately, it can be brought into a standard form by some additional rules, which would support the upper alternative from Figure 2. One way would be to compute the blocktree of biconnected components, and represent this by a tree of C-nodes. Each biconnected component can eventually partitioned by identifying a (structural) edge whose removal would provides a chain-like structure of larger components $C_1, ..., C_t$. $C_1, ..., C_t$ again form the substructures of the cycle, represented by the C-node from above. On the other hand, the hicircular layout is unique for a given hicircular tree.

Note that the scheme supports tree- and circular structures, which are planar per se, and the structural edges are always drawn pairwise crossing-free, but it is also robust enough to work for non-planar graphs as indicated in the last section. Each operation can be executed in linear time, but it seems that advanced data structures similar to [6] should enable more efficient implementations, possibly with logarithmic running times per operation. The basics of the proposed scheme have already been integrated in the yFiles library [19].

In the future, we will consider the practicability of the proposed scheme, asking if it is robust enough to be used in practical applications, and which of the variants are worthwhile to be used. We consider this proposal as a first step in the right direction, since there is certainly demand for dynamic graph drawing. And for the circular layout style, nothing has been published yet in this direction.

References

1. Booth, K.S. and Lueker G.S., *Testing for the consecutive ones property, intervals graphs and graph planarity testing using PQ-tree algorithms*, J. Comput. System Sci.13, pp. 335-379, 1976.
2. Brandenburg F., *Graph Clustering 1: Cycles of Cliques*, Proc. GD'97, LNCS 1353, Springer, pp. 158-168, 1998.
3. Branke J., *Dynamic Graph Drawing*, in Kaufmann/Wagner (Eds.) Drawing Graphs: Methods and Models, LNCS 2025, Springer, pp. 228-246, 2001.
4. Brockenauer R. and S. Cornelsen, *Drawing Clusters and Hierarchies*, in Kaufmann/Wagner (Eds.) Drawing Graphs: Methods and Models, LNCS 2025, Springer, pp. 193-227, 2001.
5. Di Battista, G., Eades P., Tamassia R. and I. G. Tollis, *Graph Drawing: Algorithms for the Visualization of Graphs*, Prentice Hall, 1999.
6. Di Battista G., and Tamassia R., *On-line graph algorithms with SPQR-trees*, Proc. ICALP 1990, LNCS 442, Springer, pp. 598-611, 1990.
7. Dogrusöz U., B. Madden and P. Madden, *Circular Layout in the Graph Layout Toolkit*, Proc. GD'96, LNCS 1190, Springer, pp. 92-100, 1997.

8. Duncan Ch., Goodrich M., and Kobourov St., *Planarity-Preserving Clustering and Embedding of Large Graphs*, Proc. GD'99, LNCS 1731, Springer, pp. 186-196, 2000.
9. Eades P., *Drawing Free Trees*, Bulletin of the Institute for Combinatorics and its Applications, 5, pp. 10-36, 1992.
10. Edachery, J., Sen A., Brandenburg F., *Graph Clustering Using Distance-k Cliques*, Proc. GD'99, LNCS 1731, Springer, pp. 98-106, 2000. .
11. Kar G., B. Madden and R. Gilbert, *Heuristic Layout Algorithms for Network Presentation Services*, IEEE Network, 11, pp. 29-36, 1988.
12. Krebs V., Visualizing Human Networks, *Release 1.0: Esther Dyson's Monthly Report*, pp. 1-25, 1996.
13. Lengauer T., Combinatorial Algorithms for Integrated Circuit Layout, Wiley-Teubner, 1990.
14. Six, J.M. and I.G. Tollis, *Circular Drawings of Biconnected Graphs*, Proc. Alenex'99, LNCS 1619, Springer, pp. 57-73, 1999.
15. Six, J.M. and I.G. Tollis, *A Framework for Circular Drawings of Networks*, Proc. GD'99, LNCS 1731, Springer, pp. 107-116, 2000.
16. Six, J.M., Ph.D. Thesis, University of Texas at Dallas, 2000.
17. Sugiyama K., S. Tagawa and M. Toda. *Methods for Visual Understanding of Hierarchical System Structures*, IEEE Transactions on Systems, Man and Cybernetics, 11(2), pp. 109-125, 1981.
18. Sugiyama K., K. Misue. *Visualisation of structural information: Automatic drawing of compound digraphs*, IEEE Transactions on Systems, Man and Cybernetics, 21(4), pp. 876-892, 1991.
19. The yFiles library: http://www-pr.informatik.uni-tuebingen.de/yfiles/ and http://www.yworks.de