A Framework for User-Grouped Circular Drawings

Janet M. Six[1] and Ioannis (Yanni) G. Tollis*[2]

[1] Lone Star Interface Design, P.O. Box 1993, Wylie, TX 75098
jsix@lonestarinterfacedesign.com
[2] Department of Computer Science, University of Crete
GR 714 09 Heraklion, Greece and
Institute of Computer Science, Foundation for Research and Technology
Hellas-FORTH, P.O. Box 1385, 71110 Heraklion, Crete, Greece
tollis@csd.uoc.gr

Abstract. In this paper we introduce a framework for producing circular drawings in which the groupings are user-defined. These types of drawings can be used in applications for telecommunications, computer networks, social network analysis, project management, and more. This fast approach produces drawings in which the user-defined groupings are highly visible, each group is laid out with a low number of edge crossings, and the number of crossings between intra-group and inter-group edges is low.

1 Introduction

A *circular graph drawing* is a visualization in which

- the graph is partitioned into groups,
- the nodes of each group are placed onto a unique embedding circle, and
- each edge is drawn with a straight line.

Circular graph drawing has received increasing attention in the literature [2, 4,13,14,20,21,22,24,25]. Kar, Madden, and Gilbert presented a circular drawing technique for networks in [13]. This approach partitions the graph into groups, places the groups onto embedding circles, and then sets the final coordinates of the nodes. As discussed in [4], an advanced version of this technique is included in Tom Sawyer Software's *Graph Layout Toolkit* (www.tomsawyersoftware.com).

Tollis and Xia present several linear time algorithms for the visualization of survivable telecommunication networks in [25].

Citing a need for graph abstraction and reduction of today's large information structures, Brandenburg describes an approach to draw a path (or cycle) of cliques in [2].

InFlow [16] is a tool to visualize social networks. This tool produces diagrams and statistical summaries to pinpoint the strengths and weaknesses within an organization (www.inflow.net).

* On leave from The University of Texas at Dallas.

G. Liotta (Ed.): GD 2003, LNCS 2912, pp. 135–146, 2004.

We presented a linear time algorithm for producing circular drawings of biconnected graphs on a single embedding circle in [20,21]. This technique was extended to place a nonbiconnected graph on a single embedding circle in [20, 22]. A framework for producing circular drawings of nonbiconnected graphs on multiple circles was presented in [20,24]. These techniques require $O(m)$ time and produce drawings with a low number of edge crossings. More details about these techniques will be dicussed in Section 2.

Kaufmann and Wiese extended our circular approach [20,21,22,24] in [14]. In this approach, the blocktree structures of nonbiconnected graphs are interpreted in a different manner which allows finer structures to be shown within the visualization of a biconnected component. The authors also give an interactive version of this technique. An advanced version of this approach is included in yWorks' *yFiles Library* (`www.yworks.com`).

All of these techniques are very useful for applications in telecommunications[15], computer networks [20], social network analysis [16], project management [16], and more. However, with the exception of the Graph Layout Toolkit (GLT) technique [4,13], these techniques do not allow the user to define which nodes should be grouped together on an embedding circle. And in the GLT technique, the layouts of the user defined groups are themselves placed on a single embedding circle. For some graph structures, this may not be ideal. In this paper, we present a circular drawing algorithm which allows the user to define the node groups, draws each group of nodes efficiently and effectively, and visualizes the superstructure well. We call this approach *user-grouped circular drawing*.

An example of an application in which user-grouped circular drawing would be useful is a computer network management system in which the user needs to know the current state of the network. It would be very helpful to allow the user to group the computers by department, floor, usage rates, or other criteria. See Figure 1. This graph drawing could also represent a telecommunications network, social network, or even the elements of a large software project. There are, of course, many other applications which would benefit from user-grouped circular drawing.

The remainder of this paper is organized as follows: in Section 2, we review our previous circular techniques. In Section 3, we introduce a framework for user-grouped circular drawing. In Section 4, we discuss a force-directed approach in which node placement is restricted to the perimeter of circles. In Section 5, we present an algorithm for user-grouped circular drawing. In Section 6, we discuss conclusions and future work.

2 Review of Our Previous Circular Techniques

As mentioned in the previous section, we have presented multiple efficient circular graph drawing techniques which produce visualizations with a low number of edge crossings [20,21,22,24]. In this section, we give a brief review of these techniques.

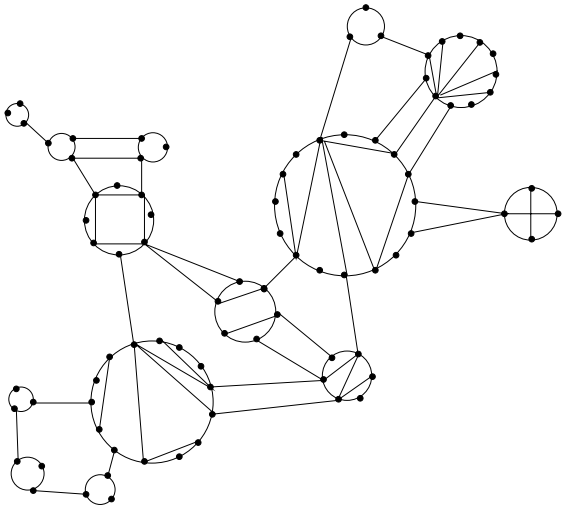

Fig. 1. A user-grouped circular drawing.

2.1 Circular Drawings of Biconnected Graphs

In [20,21], we introduced a linear time technique to produce circular graph drawings of biconnected graphs on a single embedding circle. First, it is important to note the difficulty of this problem. Of course, minimizing the number of crossings in a drawing is the well-known NP-Complete crossing number problem [10]. The more restricted problem of finding a minimum crossing embedding such that all the nodes are placed onto the circumference of a circle and all edges are represented with straight lines is also NP-Complete as proven in [17]. The authors show the NP-Completeness by giving a polynomial time transformation from the NP-Complete *Modified Optimal Linear Arrangement* problem.

In order to produce circular drawings with fewer crossings than previous techniques, we presented the algorithm *CIRCULAR* which tends to place edges toward the outside of the embedding circle. Also, nodes are placed near their neighbors.

This technique visits the nodes in a wave-like fashion, looking for *pair edges* (edges incident to two nodes which share at least one neighbor) which are then removed. Sometimes, *triangulation edges* are added to aid this process. It is the selective edge removal which causes many edges to be placed toward the periphery of the embedding circle in the visualizations produced by CIRCULAR. Subsequent to the edge removal, CIRCULAR proceeds to perform a Depth-First Search (DFS) search on the reduced graph. The longest path of the resulting DFS tree is placed on the embedding circle and the remaining nodes are nicely merged into this ordering.

The worst-case time requirement of CIRCULAR is $O(m)$, where m is the number of edges. An important property of this technique is the guarantee that

if a zero-crossing drawing exists for a given biconnected graph, CIRCULAR will find it. Such graphs must be outerplanar. In fact, CIRCULAR was inspired by the algorithm for recognizing outerplanar graphs presented in [18]. For drawings which do contain crossings, a postprocessing method which further reduces the number of crossings can be applied. See [20,21] for such a method.

Extensive experiments compared CIRCULAR and Tom Sawyer Software's GLT. CIRCULAR drawings had 15% fewer crossings. This improvement increased to 30% with the crossing-reduction postprocessing step. Sample drawings from the experimental study are shown in Figure 2.

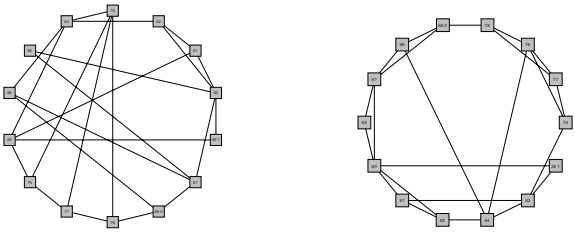

Fig. 2. The drawing on the left is produced by the GLT. The drawing on the right is of the same graph and is produced by CIRCULAR with crossing-reduction postprocessing. The drawing on the right has 75% fewer crossings than the GLT drawing.

2.2 Circular Drawings of Nonbiconnected Graphs on a Single Embedding Circle

In [20,22] we presented the algorithm *CIRCULAR-Nonbiconnected* for producing circular drawings of nonbiconnected graphs on a single embedding circle. Given a nonbiconnected graph G, we can decompose G into biconnected components. In CIRCULAR-Nonbiconnected, we layout the resulting block-cutpoint tree on a circle and then layout each biconnected component with a variant of CIRCULAR.

First, we consider how to attain a circular drawing of a tree. A DFS produces a numbering that we can use to order the nodes around the embedding circle in a crossing-free manner. From this result, we know how to order the biconnected components around the embedding circle. Next, we need to consider articulation points which are not adjacent to a bridge (*strict articulation points*). Strict articulation points appear in multiple biconnected components. In which biconnected component should a strict articulation point appear in the circular drawing? Multiple approaches to this issue are discussed in [20,22]. Due to space restrictions, we do not discuss these solutions here. A third issue to consider is how to transform the layout of each biconnected component to fit onto an arc of the embedding circle. This transformation is called *breaking*. The resulting breaks occur at an articulation point within the biconnected component.

The worst-case time requirement for CIRCULAR-Nonbiconnected is $O(m)$ if we use CIRCULAR to layout each biconnected component. The resulting drawings have the property that the nodes of each biconnected component (with the exception of some strict articulation points) appear consecutively. Furthermore, the order of the biconnected components on the embedding circle are placed according to a layout of the accompanying block-cutpoint tree. Therefore, the biconnectivity structure of a graph is displayed even though all of the nodes appear on a single circle. An example drawing is shown in Figure 3.

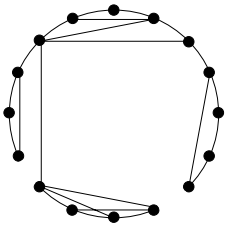

Fig. 3. An example drawing produced by CIRCULAR - Nonbiconnected.

2.3 Circular Drawings of Nonbiconnected Graphs on Multiple Embedding Circles

In [20,24] we presented Algorithm *CIRCULAR-withRadial* which produces circular drawings of nonbiconnected graphs on multiple embedding circles. As in CIRCULAR-Nonbiconnected, we first decompose the given graph G into biconnected components. Then we layout the block-cutpoint tree with a variant of the radial layout technique [1,5,9]. Then each biconnected component is laid out with a variant of CIRCULAR. Many details are omitted here. The worst-case time requirement is $O(m)$.

CIRCULAR-withRadial is a very useful technique and extension of this work to include interactive schemes has been presented by Kaufmann and Wiese in [14]. CIRCULAR-withRadial groups given graphs by their biconnected components. This grouping helps to provide a logical, beneficial view of nonbiconnected graphs, however in some cases it would be helpful to allow the user to define which nodes should be placed together on an embedding circle. We present such an algorithm in Section 5.

3 A Framework for User-Grouped Circular Drawing

The problem of producing circular drawings of graphs grouped by biconnectivity is quite different from the problem of drawing a graph whose grouping is user-defined. In the latter case, there is no known structure of either the groups or

the relationship between the groups. Therefore, we must use a general method for producing this type of visualization. The four goals of a user-grouped circular drawing technique should be:

1. the user-defined groupings are highly visible,
2. each group is laid out with a low number of edge crossings,
3. the number of crossings between intra-group and inter-group edges is low, and
4. the layout technique is fast.

We know from previous work in clustered graph drawing [6,7,8,12] that the relationship between groups is often not very complex. We take advantage of this expectation in this framework. Define the *superstructure* G_s of a given graph $G = (V, E, P)$, where P is the node group partition, as follows: the nodes in G_s represent the elements of P. For each edge $e \in E$ which is incident to nodes in two different node groups, place an edge between nodes representing the respective groups in G_s. The type of structure which we expect G_s to have should be visualized well with a force-directed [3,5] technique, therefore we will layout the superstructure G_s with this approach. Since G_s will likely not be a very complicated graph, it should not take much time to achieve a good drawing with a force-directed technique.

The node groups themselves will be either biconnected or not and since CIRCULAR and CIRCULAR-Nonbiconnected can layout biconnected and nonbiconnected graphs on a single embedding circle in linear time and have been shown to perform well in practice, we also will use those techniques here.

We have now addressed how to achieve goals 1 and 2 with good speed. However, in order to produce good user-grouped circular graph drawings, we must successfully merge these two techniques so that we can simultaneously reach goals 1,2, and 3. And, of course, we need a fast technique in order to achieve goal 4. Attaining goal 3 is very important to the quality of drawings produced by a user-grouped circular drawing technique. As shown in [19], a drawing with fewer crossings is more readable. It is especially important to reduce the number of intra-group and inter-group edge crossings as those can particularly cause confusion while interpreting a drawing. See Figure 4. How can we achieve this low number of crossings? We must place nodes which are adjacent to nodes in other groups (called *outnodes* in [4,13]) close to the placement of those other nodes. A force-directed approach is a good way to attain this goal since it would encourage outnodes to be closer to their neighbors. Traditional force-directed approaches [3,5] will not work here though, because we need to constrain the placement of nodes to circles. In Section 4, we present a force-directed approach in which the nodes are restricted to appear on circular tracks. With the use of this technique we will reach goal 3. As will be discussed, we can do this in a reasonable amount of time.

As with most force-directed techniques, the initial placement of nodes has a very significant impact on the final drawing [3,5]. Therefore it is important to have a good initial placement. This is why we should layout the superstructure and each node group first. At the completion of those steps, we should have the

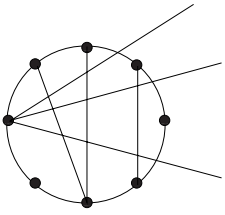

Fig. 4. Example of intra-group and inter-group edge crossings.

almost-final drawing. It will then be a matter of fine-tuning the drawing with our new circular-track force-directed technique. And as shown in [23], once you have an almost-final drawing, it does not take much time for a force-directed technique to converge.

4 Circular-Track Force-Directed

In order to adapt the force-directed paradigm for circular drawing, we need a way to guarantee that the nodes of a group appear on the circumference of an embedding circle, *the circular track*. The nodes are restricted to appear on the circular track, but are allowed to jump over each other and appear in any order. See Figure 5. And as in the force-directed approach, we want to minimize the potential energy in the spring system which is modelling the graph. In this section, we describe how this circular-track adaptation can be achieved.

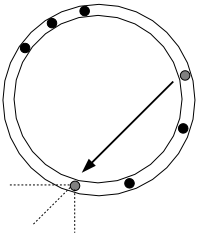

Fig. 5. Circular-track force-directed technique.

First, we need to look at node coordinates in a different way. Node i belongs to group α and is located at position (x_i, y_i). Given that the center of the embedding circle on which α is located is at (x_α, y_α) and the radius of that circle is r_α, we can restate the coordinates of i in the following way:

$$x_i = x_\alpha + r_\alpha * cos(\theta_i) \tag{1}$$

$$y_i = y_\alpha + r_\alpha * sin(\theta_i) \tag{2}$$

Remember that Hooke's Law [11] gives us the following equation for the potential energy V in a spring system:

$$V = \sum_{ij} k_{ij}[(x_i - x_j)^2 + (y_i - y_j)^2] \tag{3}$$

where k_{ij} is the spring constant for the spring between nodes i and j. Equation (3) can be rewritten using (1) and (2):

$$V = \textstyle\sum_{(i,j)\in E} k_{ij}\ [((x_\alpha + r_\alpha * cos(\theta_i)) - (x_\beta + r_\beta * cos(\theta_j)))^2 + ((y_\alpha + r_\alpha * sin(\theta_i)) - (y_\beta + r_\beta * sin(\theta_j)))^2] \tag{4}$$

where node j belongs to group β, (x_β, y_β) is the center and r_β is the radius of the embedding circle on which β appears. Following through on the minus signs, we rewrite:

$$V = \textstyle\sum_{(i,j)\in E} k_{ij}\ [(x_\alpha + r_\alpha * cos(\theta_i) - x_\beta - r_\beta * cos(\theta_j))^2 + (y_\alpha + r_\alpha * sin(\theta_i) - y_\beta - r_\beta * sin(\theta_j))^2] \tag{5}$$

We can find a minimal energy solution on variables x, y and θ. It is interesting to note that if i and j are on the same circle, then x_α and x_β are equivalent as are y_α and y_β. And, of course, $r_\alpha = r_\beta$. Now we rewrite equation (5):

$$V = \sum_{(i,j)\in E} k_{ij}[r_\alpha(cos(\theta_i) - cos(\theta_j))^2 + r_\alpha(sin(\theta_i) - sin(\theta_j))^2)] \tag{6}$$

We can calculate r_α from the number of nodes in α, so that means that finding the minimum V is now a one-dimensional problem based on finding the right set of θs. When we combine (5) or (6) with equations for magnetic repulsion to prevent node occlusion, we have a force-directed equation for which the nodes of a group lie on the circumference of a circle. Now we extend equation (5) to include repulsive forces.

$$\rho_{ij} = [(x_\alpha + r_\alpha * cos(\theta_i) - x_\beta - r_\beta * cos(\theta_j))^2 + (y_\alpha + r_\alpha * sin(\theta_i) - y_\beta - r_\beta * sin(\theta_j))^2] \tag{7}$$

$$V = \sum_{(i,j)\in E} k_{ij}\rho_{ij} + \sum_{(i,j)\in V\times V} g_{ij}\frac{1}{\rho_{ij}} \tag{8}$$

where g_{ij} is the repulsive constant between nodes i and j. The force on node i by node j is:

$$F_i = \frac{V(\theta_i + \epsilon, \theta_j) - V(\theta_i - \epsilon, \theta_j)}{2\epsilon} \tag{9}$$

where ϵ is a very small constant.

Another important consideration is the set of spring constants used in the above equations. It is not necessary for the spring constant to be the same for each pair of nodes. It is also possible for these constants to change during different phases of execution.

5 A Technique for Creating User-Grouped Circular Drawings

Now that we have a force-directed technique in which the nodes are placed on circular tracks, we need to show how we will successfully merge the force-directed approach and circular drawing techniques of [20,21,22,24]. We now present a technique for creating user-grouped circular drawings.

Algorithm 1 *CIRCULARwithFORCES*

Input: A graph $G = (V, E, P)$.
Output: User-grouped circular drawing of G Γ.

1. Determine the superstructure G_s of G.
2. Layout G_s with a basic force-directed technique.
3. For each group p_i in P
   a) If the subgraph induced by p_i, G_i, is biconnected layout G_i with *CIRCULAR*.
   b) Else layout G_i with *CIRCULAR-Nonbiconnected*.
4. Place the layout of each group p_i at the respective location found in Step 2.
5. For each group p_i
   a) rotate the layout circle and keep the position which has the lowest local potential energy.
   b) reverse the order of the nodes around the embedding circle and repeat Step 5a.
   c) if the result of Step 5a had a lower local potential energy than that of Step 5b revert to the result of Step 5a.
6. Apply a force-directed technique using the equations of Section 4 to G.

Going back to the four goals discussed in Section 3, we will attain goal 1 by using a basic force-directed technique to layout the superstructure. We will attain goal 2 by laying out each group with either CIRCULAR or CIRCULAR-Nonbiconnected. Attaining goal 3 means successfully merging the results of the force-directed and circular techniques.

Once we have the layout of the superstructure and each group, we place the layout of each group at the respective location found during the layout of the superstucture. Now we have an almost-final layout: it is a matter of rotating the layouts of the groups and maybe adjusting the order of nodes around the embedding circle. Since we know that CIRCULAR and CIRCULAR-Nonbiconnected produce good visualizations, we should change these layouts as little as possible. So first, we will fine-tune the almost-final drawing by rotating each layout and keeping the rotation which has the least local potential energy. We rotate

each embedding circle through n_α positions, where n_α is the number of nodes in the group α. With respect to determing local potential energy, we need to determine the lengths of inter-group edges which are incident to the nodes of α. The rotation of choice should minimize the lengths of those edges. In other words, we choose the rotation in which as many nodes as possible are close to their other-group neighbors. Since for each embedding circle we try n_α positions and examine the length of α's incident inter-group edges at each position, then the rotation step will take $O(n * m_{inter-group})$ time for the entire graph, where $m_{inter-group}$ is the number of inter-group edges. As discussed in Section 3, we expect $m_{inter-group} \ll m$. Then we will "flip" each layout and again rotate. We keep the rotation which has the least local potential energy. After these steps, it is still possible that some nodes will be badly placed with respect to their relationships with nodes in other groups. In other words, those placements cause intra-group and inter-group edges to cross. In order to address this problem, we will apply the force-directed technique described in Section 4. The result of this step will be the reduction of intra-group and inter-group edge crossings since nodes will be pulled to the side of the embedding circle which is closer to their other-group relatives.

Because Algorithm 1 makes use of a force-directed technique, the worst-case time requirement is unknown. However, in practice, we expect the time requirement to be $O(n^2)$ for the following reasons: Step 1 requires $O(m)$ time. Step 2 will be on a small graph and should not require much time to reach convergence. Step 3 requires $O(m)$ time. Step 4 requires $O(n)$ time. Step 5 require $O(n * m_{inter-group})$ time. Since Step 6 is a force-directed technique it could take $O(n^3)$ time in practice, however the result of the previous steps will be an almost-final layout and thus should not need much time to converge. It was evidenced in [23] that when a force-directed technique is applied to an almost-final layout, it does not take much more time for convergence to occur. Therefore, in practice we expect this step to require $O(n^2)$ time. Thus, we have attained goal 4 from Section 3.

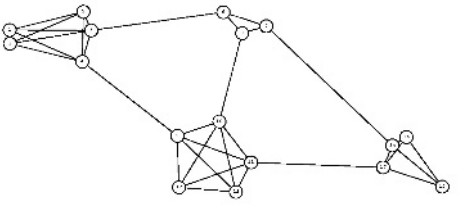

Fig. 6. Sample user-grouped circular drawing from our preliminary implementation.

We have done a preliminary implementation of CIRCULARwithFORCES in TCL/TK. In this implementation, all nodes and embedding circles are given an

arbitrary initial placement. Then the force-directed equations of Section 4 are applied to the graph with the placement of group embedding circles frozen. See Figure 6 for a sample drawing.

An interesting behavior we noticed is that the drawing with minimal energy is not necessarily the best circular drawing. In circular drawing, a major goal is to reduce edge crossings. However, it is well known [3] that reducing crossings sometimes means the compromise of other aesthetics, especially area. And area is related to minimum energy in spring systems. We propose adding springs from each node to its initial placement on the plane with the spring constants for these springs being high. This should keep these nodes from gravitating towards each other too much and causing extra crossings. We also suggest creating dummy nodes which are placed in the center of each embedding circle and attaching strong springs from them to every node in their respective group.

6 Conclusions and Future Work

In this paper, we have presented a framework for creating circular graph drawings in which the grouping is defined by the user. This framework includes the successful merging of the force-directed and circular graph drawing paradigms. We introduced the algorithm CIRCULARwithFORCES that produces drawings in which the user-defined groupings are highly visible, each group is laid out with a low number of edge crossings, and the number of crossings between intra-group and inter-group edges is low. This layout technique is also fast.

In the future we would like to extend this approach to include layouts in which nodes are placed on the periphery of shapes other than circles or on curves. We would also like to develop a technique which uses this approach in three dimensions.

Acknowledgements. We would like to thank Dr. Andrew Urquhart for his assistance with our circular-track force-directed equations.

References

1. M. A. Bernard, On the Automated Drawing of Graphs, *Proc. 3rd Caribbean Conf. on Combinatorics and Computing*, pp. 43–55, 1994.
2. F. Brandenburg, Graph Clustering 1: Cycles of Cliques, *Proc. GD '97, LNCS 1353*, Springer-Verlag, pp. 158–168, 1997.
3. G. Di Battista, P. Eades, R. Tamassia and I. G. Tollis, *Graph Drawing: Algorithms for the Visualization of Graphs*, Prentice-Hall, 1999.
4. U. Doğrusöz, B. Madden and P. Madden, Circular Layout in the Graph Layout Toolkit, *Proc. GD '96, LNCS 1190*, Springer-Verlag, pp. 92–100, 1997.
5. P. Eades, A Heuristic for Graph Drawing, *Congr. Numer.*, 42, pp. 149–160, 1984.
6. P. Eades, Q. Feng and X. Lin, Straight-Line Drawing Algorithms for Hierarchical Graphs and Clustered Graphs, *Proc. GD '96, LNCS 1190*, Springer-Verlag, pp. 113–128, 1997.

7. P. Eades and Q. W. Feng, Multilevel Visualization of Clustered Graphs, *Proc. GD '96, LNCS 1190*, Springer-Verlag, pp. 101–112, 1997.
8. P. Eades, Q. W. Feng and H. Nagamochi, Drawing Clustered Graphs on an Orthogonal Grid, *Jrnl. of Graph Algorithms and Applications*, pp. 3–29, 1999.
9. C. Esposito, Graph Graphics: Theory and Practice, *Comput. Math. Appl.*, 15(4), pp. 247–53, 1988.
10. M. Garey and D. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, 1979.
11. D. Halliday and R. Resnick, *Fundamentals of Physics: 3rd Edition Extended*, Wiley, New York, NY, 1988.
12. M. L. Huang and P. Eades, A Fully Animated Interactive System for Clustering and Navigating Huge Graphs, *Proc. of GD '98, LNCS 1547*, Springer-Verlag, pp. 107–116, 1998.
13. G. Kar, B. Madden and R. Gilbert, Heuristic Layout Algorithms for Network Presentation Services, *IEEE Network*, 11, pp. 29–36, 1988.
14. M. Kaufmann and R. Wiese, Maintaining the Mental Map for Circular Drawings, *Proc. of GD 2002, LNCS 2528*, Springer-Verlag, pp. 12–22, 2002.
15. A. Kershenbaum, *Telecommunications Network Design Algorithms*, McGraw-Hill, 1993.
16. V. Krebs, Visualizing Human Networks, *Release 1.0: Esther Dyson's Monthly Report*, pp. 1–25, February 12, 1996.
17. S. Masuda, T. Kashiwabara, K. Nakajima and T. Fujisawa, On the NP-Completeness of a Computer Network Layout Problem, *Proc. IEEE 1987 International Symposium on Circuits and Systems, Philadelphia, PA*, pp.292–295, 1987.
18. S. Mitchell, Linear Algorithms to Recognize Outerplanar and Maximal Outerplanar Graphs, *Information Processing Letters*, 9(5), pp. 229–232, 1979.
19. H. Purchase, Which Aesthetic has the Greatest Effect on Human Understanding, *Proc. of GD '97, LNCS 1353*, Springer-Verlag, pp. 248–261, 1997.
20. J. M. Six (Urquhart), *Vistool: A Tool For Visualizing Graphs*, Ph.D. Thesis, The University of Texas at Dallas, 2000.
21. J. M. Six and I. G. Tollis, Circular Drawings of Biconnected Graphs, *Proc. of ALENEX '99, LNCS 1619*, Springer-Verlag, pp. 57–73, 1999.
22. J. M. Six and I. G. Tollis, Circular Drawings of Telecommunication Networks, *Advances in Informatics, Selected Papers from HCI '99*, D. I. Fotiadis and S. D. Nikolopoulos, Eds., World Scientific, pp. 313–323, 2000.
23. J. M. Six and I. G. Tollis, Effective Graph Visualization Via Node Grouping, *Software Visualization: From Theory to Practice, The Kluwer Intl. Series in Engineering and Computer Science Vol. 734*, K. Zhang Ed., Kluwer Academic Publishers, 2003.
24. J. M. Six and I. G. Tollis, A Framework for Circular Drawings of Networks, *Proc. of GD '99, LNCS 1731*, Springer-Verlag, pp. 107–116, 1999.
25. I. G. Tollis and C. Xia, Drawing Telecommunication Networks, *Proc. GD '94, LNCS 894*, Springer-Verlag, pp. 206-217, 1994.