Acta Univ. Sapientiae, Informatica, **1**, 2 (2009) 135–148

On confluent drawings: visualizing graphs using an ortho-confluent system

Irina Honciuc
Al. I. Cuza University of Iaşi
Computer Science Faculty
email: irina.honciuc@info.uaic.ro

Cornelius Croitoru
Al. I. Cuza University of Iaşi
Computer Science Faculty
email: croitoru@info.uaic.ro

Abstract. Confluent drawing is a technique that allows visualizing non-planar graphs in a crossing-free manner. Its central idea is very simple: subsets of graph's edges are merged into confluence points and drawn as smooth curved lines, similar to train tracks. This approach eliminates edge crossings and offers an aesthetically pleasant representation for the initial graph. This article presents the ortho-confluence technique, which introduces the idea of local orthogonal system relative to a graph's node. The concept of ortho-confluence was successfully implemented in our application named ConfluentViz, and the results are presented in the final part of this article.

1 Introduction

In 2003, Dickerson, Eppstein, Goodrich and Meng introduced confluent drawing, and with it a heuristic that is able to generate confluent drawings for some graphs [4]. The problem of finding a proper representation without edge crossings for a non planar graph is not very straight forward. But there are heuristics that determine whether a non-planar graph can be efficiently drawn in a confluent way.

A particular approach in confluent drawings is ortho-confluence. This represents a way of drawing a graph in a confluent manner, such that some edges

AMS 2000 subject classifications: 68R10
CR Categories and Descriptors: G.2.2. [Graph Theory]: Graph algorithms
Key words and phrases: graph theory, graph visualization, confluence, ortho-confluence

of the initial graph are merged with a horizontal or vertical track corresponding to a local cartesian coordinate system that has its origin in another node called parent. The parent of a node in a confluent representation is, in general, the parent node given by the BFS order. Depending on the graph structure, a parent node in the confluent representation can also be a node that has a very large degree or a node that is connected with other nodes by a long edge. This article offers a close look on ortho-confluence technique as it is successfully implemented in the ConfluentViz application. This is a graph editor tool that enables the user to create and manage ortho-confluent representations for various graph categories: trees, forests, graphs containing circuits, graphs containing large cliques (bicliques).

The first sections of this paper present the context of the topic and the existing work including some important results. The following sections are related to the ortho-confluence concept. Furthermore, the implementation results and issues are explained in detail. At the end of our paper, we present some conclusions and further work.

2 Motivation and problem description

Graph visualization is a vast area of research [1, 11, 12, 13, 14]. Developing an intelligent tool that produces visual representations for different graph categories is a challenging process; it implies finding a suitable algorithm that takes a graph as an input and outputs an equivalent representation. This final representation must satisfy some aesthetic criteria in order to be relevant. It is highly desired to have less crossing edges, a good positioning for the vertices and edges, optimal angles, all these on a minimum of drawing area. Perhaps the most important criteria is edge crossings minimization, because crossed edges make the relations in the representation difficult to identify. The ideal output would be the one with no edge crossings at all.

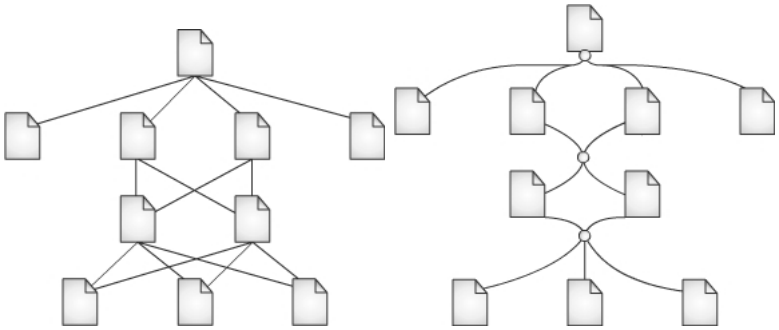
Graphs that can be represented in a standard way on a plane surface with no edge crossings are called planar graphs [15]. There are efficient algorithms that produce representations with no edge crossings for planar graphs [1]. Unfortunately, most of the graphs that appear in real life models are not planar. Thus, most of these graphs cannot be represented in a standard way without edge crossings. There are algorithms for minimizing edge crossings in non planar graphs, but the general problem of representing a non-planar graph in a standard way that minimizes edge crossings is NP-hard [7].

However, representing non planar graphs nicely is a common problem in

many domains. For example, in software visualization there are diagrams for representing application architecture, class diagrams, method calls, data flow processes, object interactions. In these diagrams the components, the entities or the objects are drawn as simple shapes: circles, squares, triangles, etc.

An important advantage of confluent representations is that in such diagrams we can easily identify source and destination nodes for the edges that share a common portion. These common structures could indicate in a method-call diagram separate methods that can be joined together for efficiency. Similarly, structures in which many sources communicate all with many destinations could indicate the need for refactoring or could offer new perspectives for changing the software design.

The navigation rules in a web application are represented in Fig. 1 in a standard way and in a confluent manner.



(a) in a standard way (b) in a confluent way

Figure 1: Two representations of a web site navigation rules

Other applications for graph visualization also include different airline maps, subway maps, social networks, genealogy. We want to obtain this kind of representations automatically. Thus, we need efficient algorithms to generate software diagrams or maps that preserve the relations in the model and at the same time the output is pleasant for the human eye.

3 Existing work

M. Dickenson, D. Eppstein, M. Goodrich, and J. Meng introduced [4] the concept of confluent representations as a way of visualizing non planar diagrams in a planar way and presented algorithms that output confluent representations for both directed and undirected graphs, mainly for graphs that appear

frequently in software visualizations.

The concept is quite simple: some edges are merged together forming "tracks" so that their intersections become overlapping paths. The resulting graphs are easier to comprehend, yet keeping a high degree of connectivity information. Some airline companies already use these confluent representations for displaying route maps. Also, similar diagrams are present in surface topology.

It is well-known that every non-planar graph contains a subgraph homeomorphic to the complete graph of five vertices, K_5, or the complete bipartite graph between two sets of three nodes, $K_{3,3}$ [15]. On the other hand, every K_n or $K_{n,m}$ admits a confluent representation as it is indicated in Fig. 2.

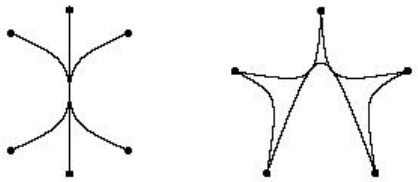


Figure 2: Confluent representations for $K_{3,3}$ and K_5

A curve is locally-monotone if it contains no self intersections and no sharp turns. Confluent representations are a way of drawing graphs on a plane surface by merging edges into paths that are unions of locally-monotone curves. An undirected graph G is confluent if and only if there exists a drawing A such that:

- There is a one-to-one mapping between the vertices in G and A, so that, for each vertex $v \in V(G)$, there is a corresponding vertex $v' \in A$, which has a unique point placement in the plane. In other words, there is a bijective function between the vertices in G and A.
- There is an edge $(v_i, v_j) \in E(G)$ if and only if there is a locally-monotone curve e' connecting v_i' and v_j' in A.
- A is planar. That is, while locally-monotone curves in A can share overlapping portions, no two of them can cross.

In a confluent representation A, a confluence point is defined as the point in plane, where two or more locally-monotone curves are merged together.

Directed confluent representations are defined similarly, except that in such drawings the locally-monotone curves are directed and the tracks formed by union curves must be oriented consistently.

There are two important visual elements that are used in confluent representations: *traffic circles* and *switches*. A switch is a common point for two or more curves or a point in which these curves change direction. A traffic circle could be defined as a confluent representation of a clique so that all the locally-monotone curves share a common portion with a circular track. This way, the clique property is preserved, that is any node is reachable from any other node (Fig. 3). Traffic circles partially solve the crossing edges problem, offering a simplified view of the representation and also a suggestive way of representing multiple connections between nodes. For example, many important cities reduced the traffic problems by eliminating cross intersections, replacing them with traffic circles.

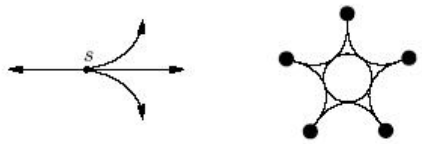


Figure 3: A switch and a traffic circle (representing K_5)

Although testing the planarity of a graph can be done in linear time, the problem of deciding whether a graph has a confluent representation is quite difficult. The main idea of the algorithm that outputs a confluent representation for a graph G is to find all the clique and biclique subgraphs of G and replace them with traffic circles. This algorithm applies especially on sparse graphs. It has been shown that the time complexity of this algorithm is $O(n)$.

Another important result is that there are large classes of non-planar graphs that can be drawn in a planar way using the confluent approach. These classes are:

- interval graphs;
- complements of trees;
- cographs;
- complements of n-cycles.

For example, a complement of a tree or an interval graph admits a confluent representation even though they are non planar graphs. The proof for each confluence theorem for the graph classes above is done especially by construction. Still, it has been demonstrated that there are some graphs that cannot

be drawn in a confluent way [4]. Among these we have the 4-dimensional cube, a certain subgraph of the Petersen graph and the Petersen graph itself. The subgraph obtained by eliminating a node from the Petersen graph is the smallest non-confluent graph we know. Also, if we divide each edge of a graph by adding a new node, the resulted graph is non-confluent. Similarly, by adding a node on each edge of a non-planar graph and connecting it to both endpoints of that edge, the result is also non-confluent. In general, all the chordal graphs are not confluent.

In 2005, David Eppstein, Michael Goodrich and Jeremy Yu Meng introduced delta-confluent drawings [6]. Delta-confluent graphs are a generalization of tree-confluent graphs. These classes of graphs and distance-hereditary graphs, a well-known class of graphs, coincide. The idea of tree-confluent graphs was published by Hui, Schaefer and Štefankovič [10]. A graph is tree-confluent if and only if it is represented by a train track system which is topologically a tree. It is also shown in their paper that the class of tree-confluent graphs is equivalent to the class of chordal bipartite class.

- A Δ-junction is a structure where three paths are united in three distinct points. Each of these points is called a junction port.
- A Λ-junction is a structure where two of the three paths in a Δ-junction are disconnected. The two paths that are disconnected are called *tails* and the remaining one is called *head*.

Figure 4: A Δ-junction (left) and a Λ-junction (right)

A Δ-confluent drawing is a confluent drawing in which each junction is either a Δ-junction or a Λ-junction and if we replace every junction in the drawing with a new vertex, the result is a tree.

4 Ortho-confluence

We can define ortho-confluence similarly to confluent representations. We say that a graph G is ortho-confluent if and only if there is a representation A

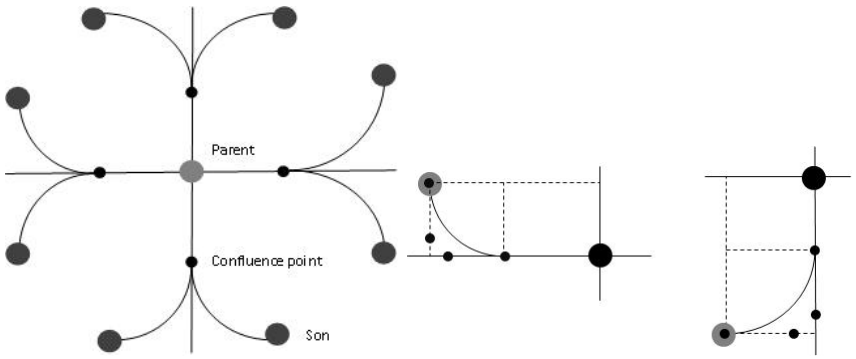
such that:

- There is a one-to-one mapping among the vertices in G and A, so that, for each vertex $v \in V(G)$, there is a corresponding vertex $v' \in A$, which has a unique point placement in the plane. In other words, there is a bijective function between the vertices in G and A.
- There is an edge $(v_i, v_j) \in E(G)$ if and only if there is a locally-monotone curve e' connecting v_i' and v_j' in A.
- A is planar. That is, while locally-monotone curves in A can share overlapping portions, no two of them can cross.
- The confluence points from any subset of curves in A must be positioned on either a vertical or a horizontal axe.

The graph classes that can be drawn both simply confluently and ortho-confluently are those in which the confluence points can be positioned on a grid. The distance between grid lines is the smallest distance between confluence points for the represented graph. Graph classes such as n-cycles complements, path complements, tree complements and interval graphs can all be represented ortho-confluently. Also, Δ-confluent graphs can be represented in an ortho-confluent manner. In general, any non-planar graph that admits a confluent representation can also be drawn ortho-confluently by applying some minor modifications:

- Traffic circles, Δ-junctions, Λ-junctions and other predefined confluence structures should be treated as nodes on a grid when representing them.
- The tangents in the endpoints of each smooth curve should form a 90 degrees angle.

However, there are some important elements that define an ortho-confluent representation. We can consider nodes in an ortho-confluent representation as being parent nodes and son nodes. The parent nodes are traversed by two tracks: a vertical track and a horizontal track that together form a local coordinate system. These tracks separate the drawing surface in four distinct quadrangles. Depending on the positioning of the son node in one of these four quadrates, there are 8 cases in which the son node can be confluently connected with its parent (Fig. 5(a)). In order to represent the tracks we used Bézier curves like in Fig. 5(b) (the control points are chosen at three quarters from the son-track distance).



(a) confluently attaching son to parent

(b) horizontal confluence (left) and vertical confluence (right)

Figure 5: The ortho-confluent system

Having a graph that does not contain large cliques, the algorithm that outputs an ortho-confluent representation has the following steps:

1. Find a parent node. This node is one of the two endpoints of the longest edge in the graph.
2. Apply a BFS on the graph and maintain an order list (the node's order given by the BFS) and a parent list (containing for each node his parent in BFS).
3. Confluently attach each node to his parent, in the order given at step 2.
4. Maintain two lists – processed and confluencePoints – that contain, for each node in the graph, whether it was included in the ortho-confluent representation and its confluence point with the vertical or the horizontal track of the parent.

5 Implementation issues

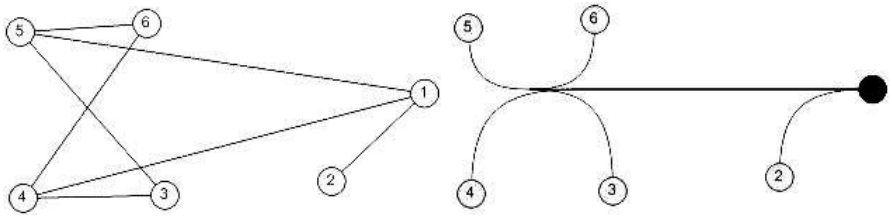
At the moment, there is no commercial graph editor to offer the possibility of representing graphs in a confluent layout. The application we developed in order to illustrate ortho-confluence representation is called ConfluentViz. The main purpose of this tool is to enable users to edit graphs and also to generate aesthetically pleasant representations for different graph classes: trees, graphs that contain circuits, graphs that contain large cliques and bicliques.

The application is developed in C# programming language and uses a free, open source system for designing diagrams and 2D user interface applications – Piccolo.NET. This has monolithic [3] class architecture: it primarily uses compile-time inheritance to extend functionality instead of using run-time composition to extend functionality. It offers the possibility of designing applications that require Zoomable User Interface (ZUI) and animations. It is developed for the .NET framework 2.0 and it is based on the classes and methods collection in GDI+ (Graphics Device Interface) for representing geometric shapes in .NET. We extended this system obtaining a graph editor with confluent layout creation support. The main functionalities of ConfluentViz are:

- graph editing;
- ortho-confluent representation;
- XML storage for graphs;
- obtaining JPEG or PostScript images from the actual representations.

The graph classes that can be represented confluently using ConfluentViz are: trees, forests, graphs containing circuits, graphs containing large cliques or bicliques. Moreover, this application offers the possibility to create a confluent layout automatically, after editing the graphs, or in an assisted manner.

An example of an ortho-confluent representation for an ordinary graph is presented below in Fig. 6. The second graph is a valid confluent representation of the first one, because the connections between nodes (represented by straight lines in the first case and smooth curves in the second case) are preserved. We can see that we can reach nodes 6 and 3 from node 5, by traversing the horizontal track that connects parent node with son nodes. The same is for nodes 4, 6 and 3. Similarly, node 5 cannot be reached from node 4 because the track that connects them is not a smooth one. All the curves are locally monotone, that is they do not have sharp turn backs or crossings.



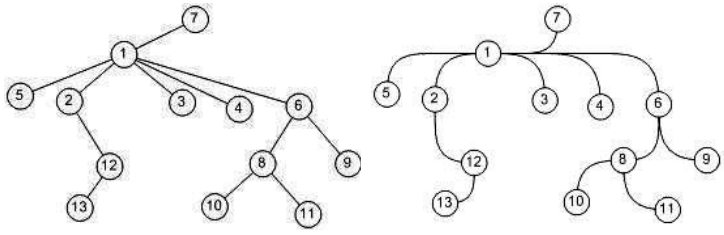
(a) the standard representation (b) the ortho-confluent representation

Figure 6: A graph represented in a standard way and in an ortho-confluent way

Algorithm 1 offers an ortho-confluent representation for trees and forests and it also uses BFS. We can mention that trees are planar graphs, thus they admit a confluent representation. The complexity of the algorithm is $O(n)$, where n is the number of nodes in the graph, and this is because we use BSF and adjacency lists. An example of an ortho-confluent representation for a tree is given in Fig. 7. We obtained good results using this algorithm especially on trees that have large degree nodes: the incident edges are better distinguished when they are represented as Bézier curves and merged together in confluence points.

Algorithm 1 ConfluentComponentTree(g)

```
1.  rootIndex <- MaxDegreeNode(g) //keep the max. degree vertex
2.  bfs <- BreadthFirstSearch(g, rootIndex)
3.  orderBFS<- bfs.order // keep the order of the nodes in BFS
4.  parentsBFS<-bfs.parents // keep the parent of each node in
    the BFS
5.  foreach i=0,VerticesCount
6.      parentIndex <- orderBFS[i]
7.      node <- editor.nodeLayer[orderBFS[i]];
8.      if (parentIndex >= 0) then
9.            parent <- editor.nodeLayer[parentIndex];
10.           Merge(parent, node);
11.     endif
12. end foreach
```



(a) the standard representation (b) the ortho-confluent representation

Figure 7: A tree represented in a standard way and in an ortho-confluent way

Cliques are complete subgraphs of a graph. In order to obtain a confluent representation for cliques we have to:

1. place each vertex of the clique on the support circle, so that we obtain a regular polygon,

2. in the middle of this support circle represent a traffic circle,

3. determine the confluence points coordinates (the intersection of the lines that unite the middle of each polygon edge and the center of the support circle with the traffic circle),

4. draw the tracks (connect each node with the closest confluence point determined at step 3).

In Fig. 8(b) we can see a traffic circle in the middle that replaces a part of the crossing edges. This traffic circle in not an actual node in the graph, it is a structure that has a visual role, facilitating the confluent representation. Bicliques are other structures that can be represented confluently. Similar to clique's case, their usual representation can have many edge crossings that make the relation in the drawing hard to identify. In order to obtain a confluent representation for bicliques we have to follow the next set of steps:

1. identify the 2 partition sets of the biclique subgraph using BFS,

2. determine the largest partition set and the node that has the highest y coordinate in this partition,

3. align the centers of the nodes in the largest partition vertically,

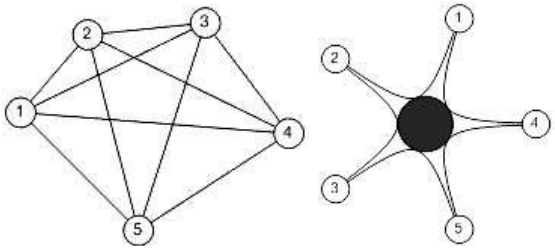
4. determine the middle nodes of the two partitions,

5. align the middle nodes horizontally,

6. align the nodes vertically in the smallest partition,

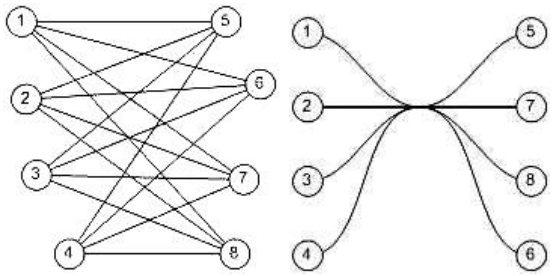
7. connect each vertex in the two partition sets with the middle of the line that unites the nodes determined at step 4.

An example of a biclique that was represented in a confluent manner using the above set of steps is given in Fig. 9.



(a) the standard representation (b) the ortho-confluent representation

Figure 8: K_5 represented in a standard way and in an ortho-confluent way



(a) the standard representation (b) the ortho-confluent representation

Figure 9: $K_{4,4}$ represented in a standard way and in an ortho-confluent way

Having an algorithm that outputs a confluent representation for structures like cliques and bicliques, we can easily obtain a confluent representation for a non-planar graph that contains large cliques or bicliques.

6 Further work

At the moment, ConfluentViz application does not use a planarity test. This would be a nice feature to have if we want to implement a general algorithm that outputs a confluent representation for any non-planar graph, similarly to HeuristicDrawUndirected algorithm presented previously.

Related to ortho-confluence, we saw that this is a particular type of conflu-

ent representation that introduces the notion of local orthogonal coordinate system. It would be interesting to determine which type of confluent representation produces better results for different graph classes. Moreover, having a very large graph, with a very complicated structure, we could use together different confluent techniques to produce a confluent representation.

There are also other types of graphs on which we can successfully apply confluent techniques. For example, directed hypergraphs [8] are a generalization for directed graphs and they can model binary relations among the subsets of a given set. These types of relations are common in different areas in Computer Science such as: data base systems, expert systems, parallel programming, scheduling, routing in dynamic networks, data mining and bioinformatics. The edges of the directed graph are called hyperarcs and they connect distinct subsets of nodes. A solution for visualizing the hypergraphs could be confluent representation. In this case, the arcs that form a hyperarc are merged together in a confluence point (grouping origin and destination of a hyperarc).

7 Conclusion

In this article, we presented a new method of visualizing different graph categories called confluent representation. This can be very useful in software visualization, topology, airline maps and subway maps or in designing site navigation rules. We introduced ortho-confluence and we identified some efficient algorithms that output aesthetic drawings for different classes of graphs. The results obtained with ConfluentViz application satisfy the main aesthetic criteria for graph visualization.

References

[1] G. Di Battista, P. Eades, R. Tamassia, I. G. Tollis, *Graph drawing: algorithms for the visualization of graphs*, Prentice Hall, 1998. ⇒ 136

[2] B. B. Bederson, *Piccolo.NET: a scalable structured graphics toolkit*, Microsoft Faculty Summit, August 2, 2004.

[3] B. B. Bederson, J. Grosjean, J. Meyer, Toolkit Design for Interactive Structured Graphics, *IEEE Transactions on Software Engineering*, **30**, 8 (2004) 535–546. ⇒ 143

[4] M. T. Dickerson, D. Eppstein, M. T. Goodrich, J. Y. Meng, Confluent drawings: visualizing non-planar diagrams in a planar way, *Proc. 11th Int. Symp. Graph Drawing (GD 2003)*, *Lecture Notes in Computer Science*, **2912**, 2003, pp. 1–12. ⇒ 135, 137, 140

[5] D. Eppstein, M. T. Goodrich, J. Y. Meng, Confluent layered drawings, *Proc. 12th Int. Symp. Graph Drawing (GD 2004)*, *Lecture Notes in Computer Science* (ed. J. Pach), **3383**, 2004, pp. 184–194.

[6] D. Eppstein, M. T. Goodrich, J. Y. Meng, Delta-confluent drawings, *Proc. 13th Int. Symp. Graph Drawing (GD 2005)*, *Lecture Notes in Computer Science* (eds. P. Healy, N. S. Nikolov), **3843**, 2006, pp. 165–176. ⇒ 140

[7] M. R. Garey, D. S. Johnson, Crossing number is *NP*-complete. *SIAM J. Algebraic Discrete Methods*, **4**, 3 (1983) 312–316. ⇒ 136

[8] A. L. P. Guedes, L. Markenzon, *Directed Hypergraph Planarity*, Technical Report RT-DINF 003/2001, Departamento de Informtica – UFPR, December 2001. ⇒ 147

[9] M. Hirsch, H. Meijer, D. Rappaport, Biclique edge cover graphs and confluent drawings, *Proc. 14th Int. Symp. Graph Drawing (GD 2006)*, *Lecture Notes in Computer Science*, **4372**, 2007, pp. 405–416. ⇒

[10] P. Hui, M. Schaefer, D. Štefankovič, Train tracks and confluent drawings, *Proc. 12th Int. Symp. Graph Drawing (GD 2004)*, *Lecture Notes in Computer Science* (ed. J. Pach), **3383**, 2004, pp. 318–328. ⇒ 140

[11] M. Jünger, P. Mutzel, *Graph drawing software*, Springer-Verlag, 2003. ⇒ 136

[12] M. Kaufmann, D. Wagner, *Drawing graphs – methods and models*, *Lecture Notes in Computer Science Tutorial*, **2025**, Springer-Verlag, 2001. ⇒ 136

[13] T. Nishizeki, S. Rahman, *Planar graph drawing*, World Scientific, 2004. ⇒ 136

[14] K. Sugiyama, *Graph drawing and applications for software and knowledge engineers*, World Scientific, 2002. ⇒ 136

[15] W. T. Tutte, C. St. J. A. Nash-Williams, *Graph theory*, Cambridge University Press, 2001. ⇒ 136, 138

Received: April 4, 2009