Fast Detection and Display of Symmetry in Trees

Joseph Manning AND Mikhail J. Atallah

DEPARTMENT OF COMPUTER SCIENCES
PURDUE UNIVERSITY
WEST LAFAYETTE, IN 47907

Abstract: The automatic construction of good drawings of abstract graphs is a problem of practical importance. Displaying symmetry appears as one of the main criteria for achieving goodness. An expression is obtained for the maximum number of axial symmetries of a tree which can be simultaneously displayed in a single drawing, and an algorithm is presented for constructing such a maximally-symmetric drawing. Similar results are also obtained for rotational symmetries in trees. The algorithms run in time which is linear in the size of the tree, and hence are optimal.

Summary

1. Drawing Graphs

   discusses the concept of producing good drawings of graphs and the criteria for goodness.

2. Displaying Symmetry in Graphs

   examines the problem of displaying axial and rotational symmetry in drawings of graphs.

3. Some Basic Properties of Axial and Rotational Symmetry

   reviews simple basic results on axial and rotational symmetry of any geometrical figure.

4. Symmetry and the Center of a Tree

   defines the center of a tree and demonstrates its importance in relationship to symmetry.

5. Axial Symmetry in Trees

   derives an expression for the maximum number of axial symmetries of a tree which can be simultaneously displayed in a single drawing, and presents a linear-time algorithm for constructing such a drawing.

6. Rotational Symmetry in Trees

   derives an expression for the maximum number of rotational symmetries of a tree which can be simultaneously displayed in a single drawing, and presents a linear-time algorithm for constructing such a drawing.

7. Remarks

   discusses the current computer implementation of the algorithms, drawing trees which have little symmetry, and related results for outerplanar graphs.

CONGRESSUS NUMERANTIUM **64**(1988), pp.159-169

1. Drawing Graphs

Mathematically, an abstract graph simply consists of two sets, V and $E \subseteq V \times V$, and as such is completely specified by an enumeration of these sets or by adjacency lists or an adjacency matrix [AHU 74]. However, such representations convey little structural information about the graph, and so graphs are instead often presented by means of drawings. Every given abstract graph may be drawn in various different ways, all equally "correct", but some certainly "better" than others in the sense that they clearly display important structural properties of the graph. For example, both drawings in Figure 1 represent the same abstract graph (the Herschel graph), but key properties such as planarity, biconnectivity, symmetry, diameter, and even bipartition are readily apparent from the second drawing but not from the first:

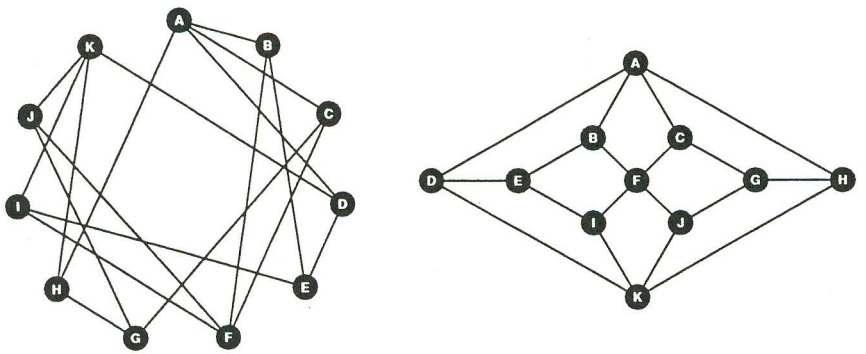

Figure 1 : *Two drawings of the same graph*

To formalize the notion of a "good" drawing of a given graph, there are several criteria which may be applied, including:

- USE STRAIGHT LINE SEGMENTS TO DRAW EDGES [Fár 48]
- DISPLAY AXIAL SYMMETRIES OF THE GRAPH [LNS 85]
- DISPLAY ROTATIONAL SYMMETRIES OF THE GRAPH [LNS 85]
- MINIMIZE THE NUMBER OF EDGE CROSSINGS ($\mathcal{NP}$-complete: [GaJ 79])
- MINIMIZE THE RATIO OF LONGEST TO SHORTEST EDGE LENGTHS
- MINIMIZE THE RATIO OF LARGEST TO SMALLEST FACE AREAS (planar graphs)
- DRAW INTERIOR FACE BOUNDARIES CONVEX (planar graphs) [CON 85]
- DRAW TRI-CONNECTED COMPONENTS CONVEX (planar graphs) [CON 85]
- SITUATE VERTICES AT INTEGER LATTICE POINTS [Shi 76, Woo 81, SuR 83]

Note that there may be conflicts between some of the above criteria, so in general it will not be possible to satisfy all of them in a single drawing. For example, the two

drawings of K_4 in Figure 2 demonstrate a conflict between displaying symmetries and minimizing edge crossings — the first shows 4 axial and 3 rotational symmetries but has an edge crossing, while the second has no edge crossing but then shows only 3 axial and 2 rotational symmetries:

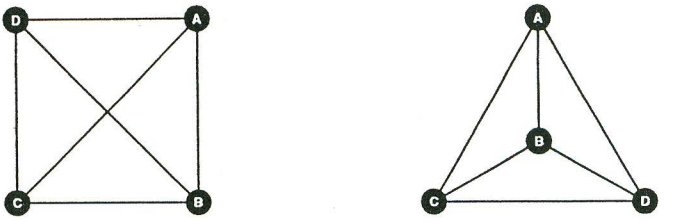

Figure 2 : *Symmetric and Planar drawings of* K_4

2. Displaying Symmetry in Graphs

From the results of practical experimentation, the best overall *general* criteria appear to be the display of axial symmetry and, to a somewhat lesser extent, the display of rotational symmetry. These criteria are now studied in more detail, with particular attention to their computational suitability. In addition, for simplicity all drawings will use straight line segments to draw edges; this ensures that a drawing of a graph is fully specified simply by the positions of its vertices, and also does not conflict with the display of either symmetry or planarity [Fár 48].

It should be emphasized at this point that axial and rotational symmetry are inherent properties of the *abstract graph*, independent of any particular *drawing*. In fact these properties can be formally defined in a purely algebraic manner in terms of automorphisms of the graph. However, for the present study it is more convenient to work with the equivalent geometric concept; an abstract graph is defined to have an axial (rotational) symmetry if there is *some* drawing of the graph which displays that axial (rotational) symmetry. Here, a (straight-edge) *drawing* of a graph is simply a set of distinct points in the plane, one point for each vertex, with straight line segments between points representing adjacent vertices. To avoid ambiguous representations, a line segment may not intersect a point unless the corresponding edge and vertex are incident, nor may two collinear segments overlap.

The detection of axial or rotational symmetry in arbitrary graphs appear to be computationally difficult problems. This is shown by the following propositions, which reduce the celebrated Graph Isomorphism problem, for which there is no known general efficient algorithm [CoG 70], to each of these symmetry detection problems:

PROPOSITION 1: *Determining if an arbitrary graph has at least one axial symmetry is computationally at least as hard as general graph isomorphism.*

PROOF: Let G_1 and G_2 be any two graphs whose isomorphism is to be tested. Form a new composite graph G from G_1 and G_2, as shown in Figure 3, by adding the triangle (u, v_1, v_2), joining v_1 and v_2 to each vertex of G_1 and G_2, respectively:

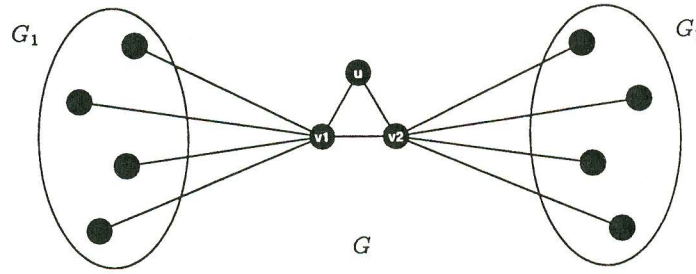

Figure 3 : *Axial Symmetry detection is as hard as Graph Isomorphism*

Then G can have only one possible axial symmetry, which must pass through u and interchange v_1 and v_2; such a symmetry exists iff G_1 and G_2 are isomorphic. □

PROPOSITION 2: *Determining if an arbitrary graph has at least one (non-trivial) rotational symmetry is computationally at least as hard as general graph isomorphism.*

PROOF: Let G_1 and G_2 be any two graphs whose isomorphism is to be tested. Form a new composite graph G from G_1 and G_2, as shown in Figure 4, by adding the path (v_1, u, v_2), joining v_1 and v_2 to each vertex of G_1 and G_2, respectively:

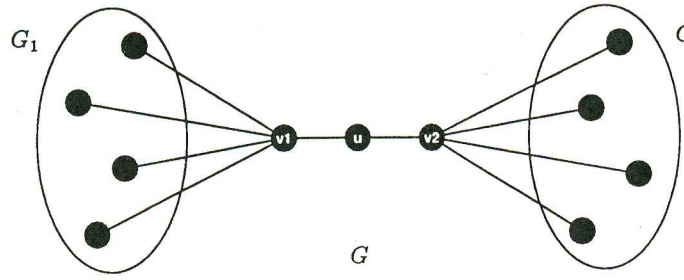

Figure 4 : *Rotational Symmetry detection is as hard as Graph Isomorphism*

Then G can have only one possible (non-trivial) rotational symmetry, which must be a 180° rotation about u and interchange v_1 and v_2; such a symmetry exists iff G_1 and G_2 are isomorphic. □

For *trees*, however, the present paper shows that both axial and rotational symmetries can be detected in a highly efficient manner. Expressions are obtained for the maximum number of axial symmetries and rotational symmetries which can be simultaneously displayed in (separate) single drawings of a given tree. Furthermore, algorithms are given which actually construct such maximally-symmetric drawings. These algorithms run in time which is linear in the number of vertices of the tree, and hence are optimal to within a constant factor.

The only other research on producing symmetric drawings of graphs appears to be that of Lipton, North, and Sandberg [LNS 85]. Results are obtained for the restricted class of *perfectly drawable* graphs, although the drawing algorithm may be computationally inefficient since it involves finding the graph's automorphism group.

3. Some Basic Properties of Axial and Rotational Symmetry

The following standard results hold for all finite geometrical figures in the plane, apart from figures composed solely of concentric annular regions — however, such exceptional cases will never arise in the present context:

AS-1: ALL AXES OF SYMMETRY INTERSECT IN A SINGLE COMMON POINT
this point is the centroid (center of gravity) of the figure.

AS-2: THE ANGLES BETWEEN ALL PAIRS OF ADJACENT AXES ARE THE SAME
if there are K axes, this common angle is $360°/2K$.

RS-1: THERE CAN BE ONLY ONE CENTER OF ROTATIONAL SYMMETRY
this point is again the centroid of the figure.

RS-2: ALL ROTATIONAL SYMMETRIES ARE MULTIPLES OF A SINGLE ROTATION THROUGH AN ANGLE θ, WHERE $K\theta = 360°$ FOR SOME INTEGER K
for a rotationally asymmetric figure, $\theta = 360°$.

4. Symmetry and the Center of a Tree

Properties *AS-1* and *RS-1* show the importance of the centroid of a geometrical figure when investigating its symmetry. Note in particular that the centroid remains fixed under all axial and rotational symmetries. Since the geometrical figures of interest here are drawings of trees, a useful graph-theoretic counterpart of the centroid is the concept of *center*.

Let the *distance* between any two vertices in a tree be the number of edges along the unique path joining them, and let a *leaf* be any vertex of degree 1.

DEFINITION: A *center* of a tree is any vertex such that the maximum distance between it and any leaf is minimized.

It is easily shown that every tree has either one center or two adjacent centers [Jor 69]; examples of each are shown in Figure 5:

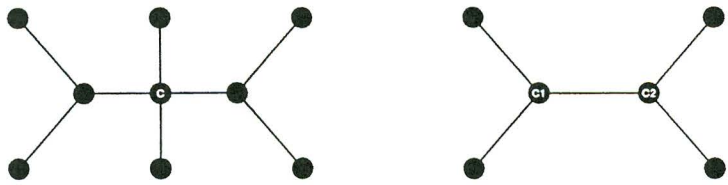

Figure 5 : *A tree with one center (C), and a tree with two centers (C_1, C_2)*

To determine the center(s) of a tree, simply delete all leaves from the tree and keep repeating this process until at the end of a complete phase, exactly one or two vertices remain — each such vertex is then a center of the original tree. By using a *queue* to store the current vertices of degree 1 at every stage, this algorithm is easily implemented to run in linear time [Mit 77].

Each symmetry of a tree must map centers onto centers, since these vertices are distinguished. Thus in a one-center tree, the center must always remain fixed, while in a two-center tree, the centers either remain fixed or are interchanged. Note however that every two-center tree can be reduced to a corresponding one-center tree by inserting a new vertex on the edge joining the two centers. Thus from now on, for simplicity it will be assumed that every tree has a single center; this center is at the intersection of all axial symmetries, is the origin of all rotational symmetries, and remains fixed under all such symmetries. Furthermore, to avoid a trivial special case, every tree will be assumed to have more than one vertex.

Removing the center and all its incident edges from a tree results in a collection of disjoint subtrees; these are called *c-trees*, and for each c-tree the unique vertex adjacent to the center is called its *root*. Under any symmetry of the original tree, every c-tree must clearly map onto an *isomorphic* c-tree, with roots mapping to roots.

Isomorphism of two rooted trees can be determined efficiently by encoding vertices as tuples of integers and testing for equality of the sets of tuples from both trees ([AHU 74], §3.2). Using a simple extension of this algorithm, the collection of c-trees may be partitioned into rooted isomorphism classes; this can be done in time which is linear in the size of the entire tree.

Let $S_1, \ldots, S_t$ denote these rooted isomorphism classes, let $N_1, \ldots, N_t$ denote the numbers of c-trees in the respective classes, and let $g = \gcd(N_1, \ldots, N_t)$.

5. Axial Symmetry in Trees

Let K_A denote the maximum number of axial symmetries of a given tree which can be simultaneously displayed in a single drawing. The main result of this section, Theorem 1, gives a computationally efficient expression for finding the value of K_A. The proof of this result is constructive in nature and leads directly to an algorithm for actually generating such a maximally-symmetric drawing. This algorithm runs in time which is linear in the size of the tree, and hence is optimal.

By property *AS-1*, all axes of symmetry in a drawing of a tree must pass through the center point. Let a *semi-axis* denote that part of an axis of symmetry emanating from the center in one direction only. Starting at an arbitrary semi-axis, consecutive semi-axes are termed *odd* and *even* respectively. Let a *wedge* denote that region of the plane properly between adjacent semi-axes; by property *AS-2* all wedges have the same angle. So in a drawing with K_A axes of symmetry, there are $2K_A$ semi-axes (K_A of them odd, K_A even), and $2K_A$ wedges, each with angle $360°/2K_A$.

An isomorphism class of c-trees is called an *A-class* if a c-tree from that class, together with the center of the entire tree and the edge joining the center to the root, has itself got an axial symmetry passing through both center and root. Thus in a drawing of the tree, (the root of) a c-tree can lie along a semi-axis only if it belongs to an A-class. To determine if a class S_i is in fact an A-class, take any c-tree in S_i, remove its root, and partition the resulting subtrees into rooted isomorphism classes; then S_i is an A-class iff at most *one* of these latter subclasses has odd cardinality and is itself an A-class, a fact which may be determined recursively. Again, this can all be done in time which is linear in the size of the c-tree. It also shows how a c-tree from an A-class may be laid along a semi-axis: its root is placed on the semi-axis, and an equal number of subtrees from each subclass are placed symmetrically on either side of the semi-axis, with a single subtree from the odd-sized subclass, if the latter exists, laid along the semi-axis by a recursive application of this method.

Let $M_i = N_i/g$ $(1 \leq i \leq t)$, and define the conditions (α) and (β) as shown; note that these two conditions can be evaluated in linear time:

$$(\alpha) \cdots\cdots\cdots \begin{cases} \textit{at most two of the } N_i\textit{'s are odd} \\ \quad \text{AND} \\ S_i \textit{ is an A-class whenever } N_i \textit{ is odd} \end{cases}$$

$$(\beta) \cdots\cdots\cdots \begin{cases} \textit{at most two of the } M_i\textit{'s are odd} \\ \quad \text{AND} \\ S_i \textit{ is an A-class whenever } M_i \textit{ is odd.} \end{cases}$$

The value of K_A can then be expressed as follows:

THEOREM 1:
$$K_A = \begin{cases} 0 & \text{if } (\alpha) \text{ fails} \\ g & \text{if } (\beta) \text{ holds} \\ g/2 & \text{otherwise.} \end{cases}$$

Proof: This result is an immediate consequence of Lemmas 1, 3, and 4, below. □

Suppose $K_A \neq 0$ and consider any drawing of the tree which displays all K_A axes. By symmetry under reflection in semi-axes, each class S_i must have the same number of c-trees in every wedge, along every odd semi-axis, and along every even semi-axis; let w_i, o_i, e_i denote these quantities, respectively. Furthermore, all odd semi-axes, and similarly all even semi-axes, must either be unoccupied, or else be occupied by c-trees from the *same* class. In particular, at most *two* classes can have c-trees lying along semi-axes, and these must be A-classes.

LEMMA 1: $K_A \neq 0 \iff (\alpha)$.

Proof:

$\Longrightarrow$: Consider any drawing of the tree which displays all K_A axes of symmetry. For those S_i's whose c-trees all lie in the $2K_A$ wedges, $N_i = 2K_A w_i$ which is even. Thus N_i can be odd only for those S_i's which have some c-trees along semi-axes. There can be at most two such S_i's and these must be A-classes.

$\Longleftarrow$: Construct a drawing of the tree by choosing any line in the plane, placing the center of the tree on this line, and laying one c-tree from each odd-sized A-class along distinct halves of the line. All remaining c-trees can be grouped into isomorphic pairs; arrange each such pair symmetrically on opposite sides of the line. This completes the drawing of the tree; the chosen line is an axis of symmetry, so $K_A \geq 1$. □

LEMMA 2: $K_A \neq 0 \implies K_A \mid g$.

Proof: For $K_A \neq 0$, in any drawing of the tree which displays K_A axes of symmetry:
$$N_i = 2K_A w_i + K_A o_i + K_A e_i = (2w_i + o_i + e_i) K_A.$$
Thus $K_A \mid N_i$ for each $1 \leq i \leq t$, giving $K_A \mid g$. □

LEMMA 3: $K_A = g \iff (\beta)$.

Proof:

$\Longrightarrow$: Consider any drawing of the tree which displays $K_A = g$ axes of symmetry. For those S_i's whose c-trees all lie in the $2g$ wedges, $N_i = 2g w_i$, giving $M_i = 2w_i$ which is even. Thus M_i can be odd only for those S_i's which have some c-trees along semi-axes. There can be at most two such S_i's and these must be A-classes.

$\Longleftarrow$: Construct a drawing of the tree by choosing g axes, subject to *AS-1* and *AS-2*, and place the center of the tree at their point of intersection. If only one M_i is odd, lay a c-tree from the corresponding S_i along each odd semi-axis; if two M_i's are odd, lay c-trees from the corresponding S_i's along odd and even semi-axes, respectively. In either case, let $\hat{N}_i$ denote the number of unplaced trees remaining in each class S_i, and let $\hat{M}_i = \hat{N}_i/g$; each $\hat{M}_i$ is even. Complete the drawing by placing $\hat{M}_i/2 = \hat{N}_i/2g$ c-trees from each class S_i in each of the $2g$ wedges, arranged symmetrically around semi-axes. This drawing shows that $K_A \geq g$, so by Lemma 2, $K_A = g$. □

LEMMA 4: (α) AND NOT (β) $\Longrightarrow$ $K_A = g/2$.

Proof: Note first that if g is odd, then N_i $(= g.M_i)$ is odd whenever M_i is odd, and so $(\alpha) \Longrightarrow (\beta)$. Thus if (α) AND NOT (β) holds, g must be even.

Construct a drawing of the tree by choosing $g/2$ axes, subject to *AS-1* and *AS-2*, and place the center of the tree at their point of intersection. Then place N_i/g c-trees from each class S_i in each of the g wedges, arranged symmetrically around semi-axes. This drawing shows that $K_A \geq g/2$, so by Lemmas 2 and 3, $K_A = g/2$. □

The proofs of Lemmas 3 and 4 provide an efficient algorithm for constructing a maximally-symmetric drawing of any given tree. An example is shown in Figure 6. Here, $(N_1, N_2, N_3) = (4, 6, 6)$ so $g = 2$ and $(M_1, M_2, M_3) = (2, 3, 3)$, and since both S_2 and S_3 are A-classes, condition (β) holds, giving $K_A = g = 2$:

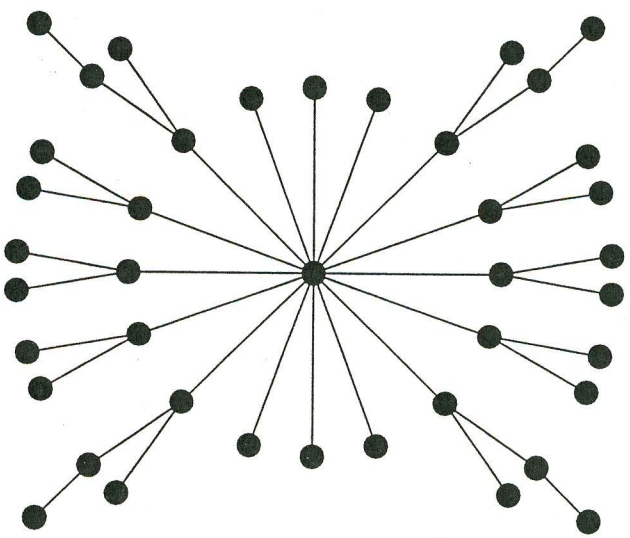

Figure 6 : *Drawing of a tree, showing vertical and horizontal Axial Symmetries*

6. Rotational Symmetry in Trees

Recall from property $\mathcal{RS}$-2 that all rotational symmetries are multiples of a single rotation through an angle θ, where $K\theta = 360°$ for some integer K. Let K_R denote the largest such K for any drawing of a given tree. Then K_R is in fact the maximum number of rotational symmetries of the tree which can be simultaneously displayed. The value of K_R is given by:

THEOREM 2: $K_R = g$.

Proof: Consider any drawing of the tree which displays all K_R rotational symmetries. Partition the plane into K_R similar conical-shaped *wedges*, having a common angle $\theta = 360°/K_R$ and a common apex at the location of the center of the tree, oriented so that the roots of all c-trees lie properly within wedges. By symmetry under a rotation through θ, each wedge must be identically occupied. Thus for each class S_i the N_i c-tree roots must be distributed equally among the K_R wedges, so $K_R \mid N_i$ for each $1 \leq i \leq t$, giving $K_R \mid g$.

Construct a drawing of the tree by again partitioning the plane into g similar conical-shaped wedges, having a common angle $\theta = 360°/g$ and a common apex. Place the center of the tree at this apex, and place N_i/g c-trees from each class S_i in each of the g wedges, arranged in the same configuration within each wedge. The resulting drawing is symmetric about the center under a rotation through θ. It follows that $K_R \geq 360°/\theta = g$, giving $K_R = g$. □

The proof of Theorem 2 provides an efficient algorithm for constructing a drawing of the tree which displays all K_R rotational symmetries. This algorithm runs in time which is linear in the size of the tree, and hence is optimal.

7. Remarks

The two symmetry algorithms have been successfully implemented on a computer. The programs take as input the adjacency lists of an arbitrary tree and produce as output a device-independent coordinate specification for a symmetric drawing. These coordinates can then be used to display the tree on either a high-resolution graphics terminal or a laser printer. As seen above, the theoretical running time of both algorithms is linear in the size of the tree; in actual practice, the response of the programs is almost instantaneous, for trees with up to one hundred vertices. The programs are written in the language *Modula-2*, and further details are available upon request from the first author.

Many trees possess very little symmetry. Nevertheless, the methods described here may still be applied to produce good drawings, by relaxing the definition of the classes S_i — instead of requiring that c-trees in the same class be strictly isomorphic, some weaker criteria may be used, such as c-trees having the same number of vertices, the same height, or the same degree at the root.

Recent work has extended the above results to the case of *outerplanar graphs*. Using the present algorithms as a foundation, and adding some substantially different techniques, [MaA 86] obtains linear-time algorithms for the detection and display of both axial and rotational symmetries in arbitrary outerplanar graphs.

Acknowledgements

Research supported by the Office of Naval Research (Grants N00014-84-K-0502, N00014-86-K-0689), and the National Science Foundation (Grant DCR-8451393), with matching funds from AT&T.

References

[AHU 74] AHO, A., HOPCROFT, J., AND ULLMAN, J. (1974) *The Design and Analysis of Computer Algorithms.* Addison-Wesley.

[CoG 70] CORNEIL, D., AND GOTLEIB, C. (1970) "An Efficient Algorithm for Graph Isomorphism". *J. ACM 17*, 1, pp 51 – 64.

[CON 85] CHIBA, N., ONOGUCHI, K., AND NISHIZEKI, T. (1985) "Drawing Plane Graphs Nicely". *Acta Informatica 22*, pp 187 – 201.

[Fár 48] FÁRY, I. (1948) "On Straight Line Representation of Planar Graphs". *Acta Sci. Math. Szeged, 11*, pp 229 – 233.

[GaJ 79] GAREY, M., AND JOHNSON, D. (1979) *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman.

[Jor 69] JORDAN, C. (1869) "Sur les Assemblages de Lignes". *J. Reine Angew. Math. 70*, pp 185 – 190.

[LNS 85] LIPTON, R., NORTH, S., AND SANDBERG, J. (1985) "A Method for Drawing Graphs". *Proc. ACM Symposium on Computational Geometry*, pp 153 – 160.

[MaA 86] MANNING, J., AND ATALLAH, M. (1986) "Fast Detection and Display of Symmetry in Outerplanar Graphs". Tech. Rep. CSD-TR-606, Dept. of Computer Sciences, Purdue Univ.

[Mit 77] MITCHELL, S. (1977) "Algorithms on Trees and Maximal Outerplanar Graphs: Design, Complexity Analysis, and Data Structures Study". Ph.D. Thesis, Univ. of Virginia.

[Shi 76] SHILOACH, Y. (1976) "Arrangements of Planar Graphs on the Planar Lattice". Weizmann Institute of Science, Rehovot, Israel.

[SuR 83] SUPOWIT, K., AND REINGOLD, E. (1983) "The Complexity of Drawing Trees Nicely". *Acta Informatica 18*, pp 377-392.

[Woo 81] WOODS, D. (1981) "Drawing Planar Graphs". Ph.D. Thesis, Stanford Univ.