3

Symmetric Graph Drawing

Peter Eades
University of Sydney

Seok-Hee Hong
University of Sydney

3.1 Introduction

Symmetry is one of the most important aesthetic criteria that clearly reveals the structure and properties of a graph. Graphs in textbooks on graph theory are normally drawn symmetrically. In some cases, a symmetric drawing may be preferred over a planar drawing. As an example, consider the two drawings of the same graph in Figure 3.1 (from [KK89]). The left drawing has five edge crossings, but eight symmetries (four rotations and four reflections). On the right is a planar drawing; it only has axial symmetry. Most people prefer the drawing on the left. As another example, the Petersen graph is normally drawn as in Figure 3.2. This drawing shows ten symmetries (five rotations and five reflections). In fact, it can be shown that a drawing of the Petersen graph can have at most ten symmetries, and Figure 3.2 is maximally symmetric.

Of course, every drawing has the trivial symmetry, the identity mapping on the plane. The aim of *symmetric graph drawing* is to draw a graph with nontrivial symmetry. More ambitiously, we aim to draw a graph with as much symmetry as possible.

Symmetries of a drawing of a graph G are clearly related to the automorphisms of G; intuitively, a symmetry of a drawing of G induces an automorphism of the graph. For example, in Figure 3.2, a rotational of the plane by $2\pi/5$ is a symmetry of the drawing and induces the automorphism $(0,1,2,3,4)(5,6,7,8,9)$. A reflection of the plane in a vertical axis induces the automorphism $(1,4)(6,9)(7,8)(2,3)$. The automorphism group of a graph G defines its "combinatorial symmetries." However, not every automorphism can be represented as a symmetry of a *drawing* of G. For example, the automorphism group of the Petersen graph has 120 elements, but, as mentioned above, a drawing can display only ten

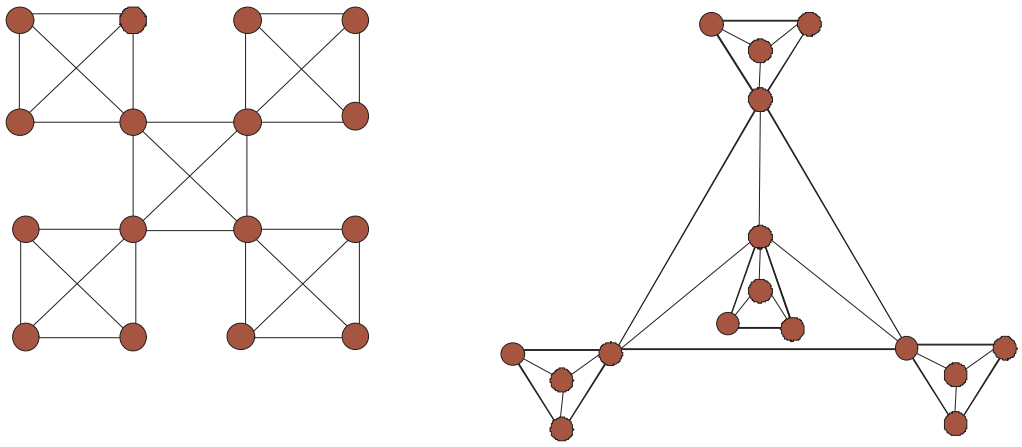

Figure 3.1 Two drawings of the same graph: a planar drawing with eight symmetries and five edge crossings, and a planar drawing with an axial symmetry [KK89].

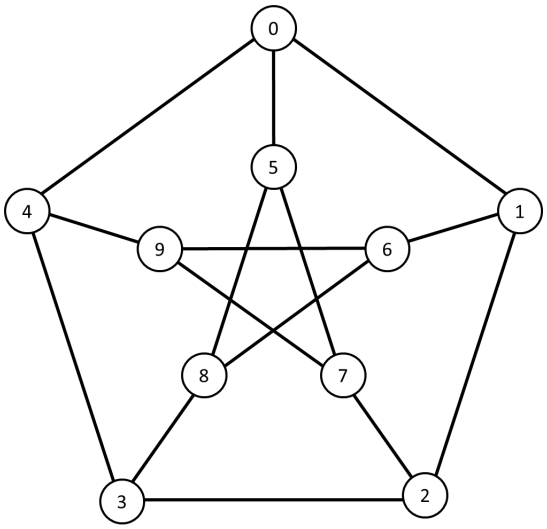

Figure 3.2 A drawing of the Petersen graph.

of these. Symmetric graph drawing involves determining those automorphisms of a graph G that can be represented as symmetries of a drawing of G.

This chapter describes a formal model for symmetric graph drawing in Section 3.2, and gives a characterization of subgroups of the automorphism group of a graph that can be displayed as symmetries of a drawing in Section 3.3. Most precise formulations of the symmetric graph drawing problem are NP-complete. Section 3.4 describes a proof of the NP-completeness of one such formulation and briefly reviews some heuristics for the general symmetric graph drawing problem. Of course, we want a drawing of a graph to satisfy other aesthetics as well as symmetry. In particular, it is useful to examine the problem of constructing a planar straight-line drawing of a planar graph, such that symmetry is maximized. Surprisingly, there is a linear-time algorithm for this problem; it is sketched in Section 3.5. The chapter concludes with a brief survey of some other approaches to symmetric graph drawing and some open problems.

3.2 Basic Concepts for Symmetric Graph Drawing

3.2.1 Drawing of a graph

A *graph* $G = (V, E)$ consists of a set V of vertices and a set E edges, that is, unordered pairs of vertices. Unless explicitly stated otherwise, we assume that the graph is *simple*, that is, it has no multiple edges and no self-loops.

A *drawing* D of a graph G consists of a point $D_V(u)$ in R^2 for every vertex $u \in V$, and a closed curve segment $D_E(u, v)$ in R^2 for every edge $(u, v) \in E$. The curve $D_E(u, v)$ has its endpoints at $D_V(u)$ and $D_V(v)$. Through most of this chapter, the curve $D_E(u, v)$ is a straight-line segment.

For an investigation of symmetric graph drawing, we must take a little care about the definition of the drawing of a graph. We allow two curves $D_E(u, v)$ and $D_E(u', v')$ to cross (share a point), but we have some non-degeneracy conditions as follows:[1]

ND1 The mapping D_V is injective. This excludes, for example, the ultra-symmetric case where all vertices are drawn at the origin.

ND2 A curve $D_E(u, v)$ must not contain a point $D_V(w)$ where $u \neq w \neq v$; in other words, an edge must not intersect with a vertex to which it is not incident.

ND3 Two curves must not overlap; that is, they must not share a curve of nonzero length. This excludes, for example, the axially symmetric case where all the vertices of the graph are drawn on the x axis.

ND4 If two curves share a point, then they must cross at this point; that is, they alternate in cyclic order around the crossing point.

Note that for straight-line drawings, ND2 implies both ND3 and ND4. Most of this chapter is concerned with straight-line drawings, and so discussions of degeneracy concentrate on ND1 and ND2.

3.2.2 Automorphisms of a graph

Basic concepts and terminology for permutation groups can be found in [Wie64].

An *isomorphism* from a graph $G_1 = (V_1, E_1)$ to a graph $G_2 = (V_2, E_2)$ is a one-one mapping β of V_1 onto V_2 that preserves adjacency, that is, $(u, v) \in E_1$ if and only $(\beta(u), \beta(v)) \in E_2$. An *automorphism* of a graph $G = (V, E)$ is an isomorphism of G onto itself, that is, a permutation of the vertex set that preserves adjacency. The *order* of an automorphism β is the smallest positive integer k such that β^k is the identity.

Any set of automorphisms of G that forms a group is called an *automorphism group* of G; the set of all automorphisms of G is denoted by $aut(G)$. The *size* of an automorphism group is the number of elements of the group.

We have defined an automorphism group A of a graph as a permutation group on the vertex set V of a graph $G = (V, E)$. It is easy to see that this defines a permutation group A' acting on the edge set E, and it is often convenient to regard A as acting on E. For

[1] The graph drawing literature is somewhat inconsistent about the precise details of the definition of a graph drawing. In some places, a drawing with these non-degeneracy conditions is called a *strict*, *clear*, and/or *proper*. In this chapter, however, we use the term "graph drawing" to includes these non-degeneracy conditions.

example, if $\beta \in A$ and $(u, v) \in E$, then we write "the edge $\beta(u, v)$" to denote the edge $(\beta(u), \beta(v))$.

A subset $B = \{\beta_1, \beta_2, \ldots, \beta_k\}$ of an automorphism group A *generates* A if every element of A can be written as a product of elements of B. We denote the group A generated by B by $\langle \beta_1, \beta_2, \ldots, \beta_k \rangle$. From the computational point of view, generators are important because they give a succinct way to represent an automorphism group. If we were to represent a permutation explicitly, then it may require $\Omega(n)$ space, where $n = |V|$. Thus, an explicit representation of an automorphism group of size k may take space $\Omega(kn)$. In many cases this is too large; for example, the space requirement may preclude a linear-time algorithm, merely because the representation of the output is super-linear. To avoid this problem, we usually represent a group by a set of generators; in general the set of generators is smaller than the group. Most of the groups discussed in this chapter are generated by one or two elements.

Many of the difficulties of symmetric graph drawing arise when vertices and/or edges are fixed by an automorphism. For this reason, we need a careful notion of "fix." Suppose that A is an automorphism group of $G = (V, E)$. The *stabilizer* of $u \in V$, denoted by $stab_A(u)$, is the set of automorphisms in A that fix u, that is,

$$stab_A(u) = \{\beta \in A \mid \beta(u) = u\}. \tag{3.1}$$

The definition can be extended to subsets of V: if $Y \subseteq V$, then

$$stab_A(Y) = \{\beta \in A \mid \forall y \in Y,\ \beta(y) \in Y\}. \tag{3.2}$$

Note that the stabilizer of a set fixes the set setwise.

For each automorphism β we denote $\{u \in V \mid \beta(u) = u\}$ by fix_β. The set of vertices that are fixed *elementwise* by every element of A is denoted by fix_A, that is,

$$fix_A = \{v \in V \mid \forall\ \beta \in A,\ \beta(v) = v\ \}. \tag{3.3}$$

Note that while $stab_A(Y)$ is a set of group elements, fix_A is a set of vertices. Further, the expression "fix the edge (u, v)" does not necessarily entail "fixing u and fixing v"; it could mean that u and v are swapped.

If $\beta \in A$ and $u \in V$, then the *orbit* of u under β, denoted by $orbit_\beta(u)$, is the set of images of u under $\langle \beta \rangle$, that is,

$$orbit_\beta(u) = \{\beta^i(u) \mid 0 \leq i < k\}, \tag{3.4}$$

where β has order k. We can extend this definition to groups: the *orbit* of u under A is

$$orbit_A(u) = \{\beta(u) \mid \beta \in A\}. \tag{3.5}$$

Note that the orbits partition V. The following theorem is fundamental in finite group theory.

Theorem 3.1 (Orbit-stabilizer theorem [Arm88]) *Suppose that A is a group acting on a set X and let $x \in X$. Then $|A| = |orbit_A(x)| \times |stab_A(x)|$.*

The following corollary is helpful in the following sections.

COROLLARY 3.1 Suppose that A is a group acting on a set X.

- If A has no fixed points, then $|orbit_A(x)| = |A|$ for every $x \in X$.
- If A has one fixed point $w \in X$, then $|orbit_A(x)| = |A|$ for every $x \neq w \in X$.

3.2.3 Symmetries of a graph drawing

Symmetry is an intuitive notion that can be formally defined in many different ways. In this chapter we will concentrate on a standard mathematical notion of symmetry; other notions are discussed in Section 3.6.

An *isometry* is a mapping of the plane onto itself that preserves distances. A *symmetry* of a drawing $D = (D_V, D_E)$ of a graph $G = (V, E)$ is an isometry σ of the plane that maps the drawing onto itself, that is:

- for every vertex $u \in V$, there is a vertex $v \in V$ such that $\sigma(D_V(u)) = D_V(v)$, and
- for every edge $(u, v) \in E$, there is an edge $(a, b) \in E$ such that $\sigma(D_E(u, v)) = D_E(a, b)$.

Note that if σ is a symmetry of a drawing $D = (D_V, D_E)$ of a graph $G = (V, E)$, then $\beta = D_V^{-1}\sigma D_V$ is an automorphism of G. We say that D *displays* β. Given an automorphism β, if there is a drawing which displays β, then we say that β is *geometric*.

An automorphism group A is *geometric* if every element of A is displayed in a single drawing; in this case the drawing *displays* A.

To define the intuitive notion of "maximally symmetric drawing" of a graph, we need to decide what it means for one drawing to display more symmetry than another. Here we take a simple view: that if D displays A and D' displays A', then D is more symmetric than D' if A has a larger size than A'. This means that searching for a maximally symmetric drawing entails searching for a maximum size geometric automorphism group.

3.3 Characterization of Geometric Automorphism Groups

Suppose that a drawing D of a graph $G = (V, E)$ displays the automorphism group A. Let A' denote the group of symmetries of D. It is useful to note the group-theoretic relationship between A and A'. If D contains three non-collinear points, then A is isomorphic to A' because a motion of three non-collinear points in the plane uniquely determines an isometry. If all the vertices of the drawing lie on a single line, it may be the case that $|A'| = 4$ while $|A| = 2$, because the rotation by π gives the same automorphism as a reflection in the line. However, this is a pathological case, because the only graphs that have drawings on a single line are sets of paths; in general, we assume that A is isomorphic to A'.

Next, we consider the simple question: Given an automorphism group A of a graph G, is there a drawing of G that displays A? The answer is straightforward, since a symmetry of a finite set of points in the plane is relatively straightforward. The following theorem is an extension of results of Lipton et al. [LNS85], Manning et al. [MA86, MA88, AM88], and Lin [Lin92] to handle degeneracies.

Theorem 3.2 *Suppose that A is an automorphism group of a graph G. Then:*

(a) A can be displayed as a reflection if and only if $|A| = 2$ and fix_A induces a set of disjoint paths.

(b) A can be displayed as a rotation if and only if all the following conditions hold

i. A has one generator ρ, and

ii. $|fix_A| \leq 1$, and

iii. if A fixes an edge, then $|fix_A| = 0$.

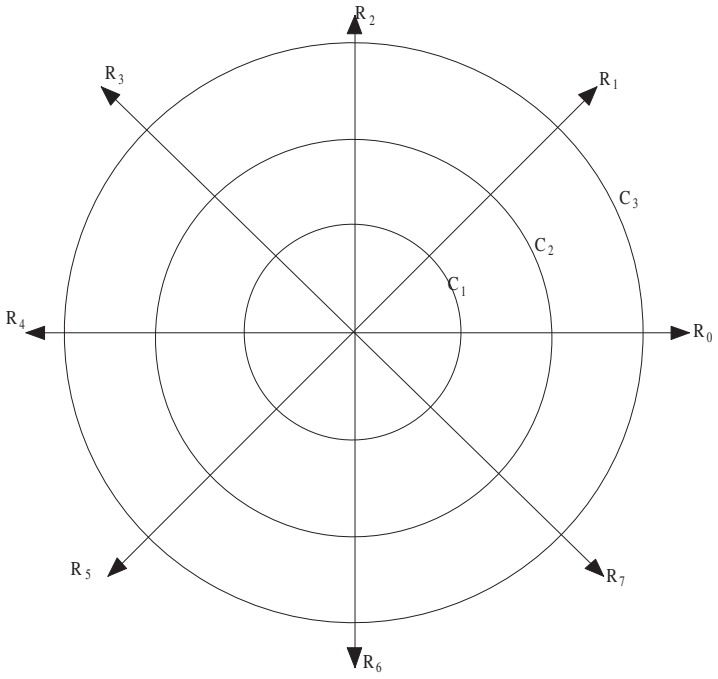

Figure 3.3 A circular grid.

(c) A can be displayed as a dihedral group if and only if all the following conditions hold.

i. A is dihedral; that is, it has two generators α and ρ such that $\alpha^2 = 1$, $\rho^k = 1$ for some $k > 1$, and $\alpha\rho = \rho^{-1}\alpha$.

ii. $|fix_A| \leq 1$.

iii. fix_α induces a set of disjoint paths.

iv. If ρ fixes an edge, then $|fix_A| = 0$.

The proof of Theorem 3.2 is an algorithm, stated below, that takes a graph G and an automorphism group A satisfying the conditions of the theorem, and draws G to display A. The drawing is on a *circular grid* as illustrated in Figure 3.3. An $m \times n$ circular grid has $n \geq 2$ equally spaced rays $R_0, R_1, \ldots, R_{n-1}$ from the origin, that is, the ray R_i makes an angle of $2\pi i/k$ to the x axis. There are $m \geq 1$ circles $C_1, C_2, \ldots, C_m$ centered at the origin, in increasing order of radius. However, the circles may not be equally spaced. The drawing algorithm below chooses a radius for each circle, and places vertices at the grid points, that is, at the intersection points between the circles and the rays.

To prove part (c) of Theorem 3.2, we need the following technical lemma.

LEMMA 3.1 Suppose that the radius of the circle C_i in the $m \times n$ circular grid is n^i, and there are three circular grid points that lie on a straight line ℓ. Then ℓ passes through the origin.

Proof: Suppose that u, v and w are circular grid points on C_i, C_j, and C_k, respectively, and that $i \geq j \geq k$.

Note the case $i = j = k$ is not possible.

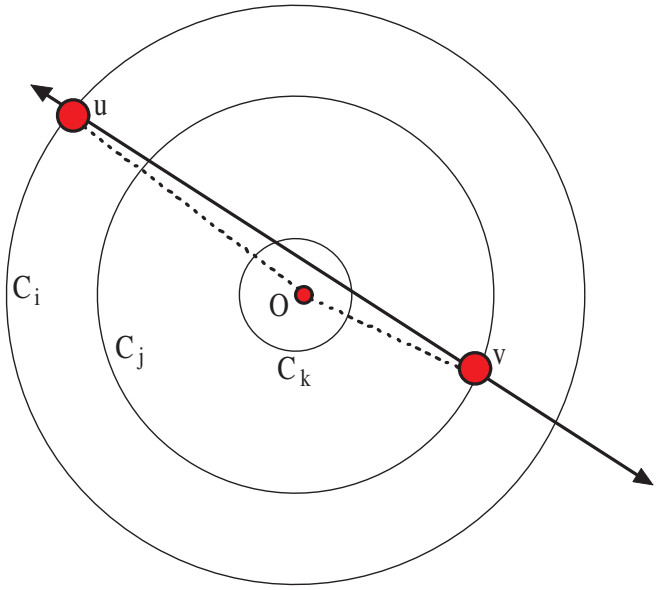

Figure 3.4 Three-in-a-line for the circular grid: first case.

First, consider the case that $i \geq j > k$, as in Figure 3.4. We show that the line segment between u and v cannot intersect C_k unless it passes through the origin.

Assume that such an intersection occurs. Then let $\theta = \angle Ouv$ and $\phi = \angle Ovu$. Since the line segment between u and v intersects C_k with $k < j$ and the radius of c_k is at least n times smaller than the radii of C_i and C_j, it follows that $\sin(\theta) \leq n^{-1}$ and $\sin(\phi) \leq n^{-1}$. One can deduce that for $n > 2$:

$$\sin(\theta + \phi) < 2n^{-1}. \tag{3.6}$$

However, considering the triangle uOv and noting that u and v are at circular grid points, we can see that $\theta+\phi$ is an integer multiple of $2\pi n^{-1}$. If both are nonzero, then $\theta+\phi \geq 2\pi n^{-1}$; this implies that $\sin(\theta+\phi) \geq 2n^{-1}$, contradicting the inequality above. It follows that both θ and ϕ are zero, and so the line segment between u and v passes through the origin.

For the case that $i > j = k$, as in Figure 3.5, a variation of the argument above can be used to show that ℓ passes through the origin.

□

Next, we prove Theorem 3.2. We prove each of the parts (a), (b), and (c) in turn.

Part (a) Suppose that A is displayed as a reflection. Then every vertex u on the line of reflection is fixed by the reflection and thus $u \in fix_A$. Any cycle or vertex of degree more than two on this line violates the non-degeneracy conditions, thus fix_A induces a set of disjoint paths.

Conversely, suppose that $A = \{1, \alpha\}$ is an automorphism group such that fix_A induces a set of disjoint paths. We use the following algorithm.

First, draw $V - fix_A$ on a circle about the origin, so that the x coordinates of u and $\alpha(u)$ are the same, then draw the edges induced by $V - fix_A$ as straight lines. Note that, so far, the drawing is axially symmetric about the y axis and it is non-degenerate.

Next, note that the edges induced by $V - fix_A$ cross the y axis at a finite number of places. Draw fix_A on the y axis, one path at a time, in such a way that the vertices of fix_A avoid the edges induced by $V - fix_A$.

Finally, draw the edges between $V - fix_A$ and fix_A.

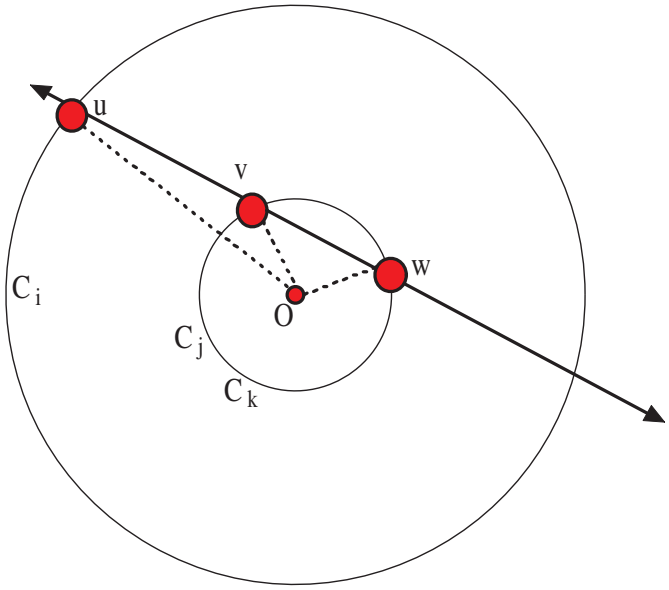

Figure 3.5 Three-in-a-line for the circular grid: second case.

This drawing displays α as a reflection in the y axis; note that it satisfies the non-degeneracy conditions.

Part (b) Suppose that A is displayed by a rotation. It is clear that A has one generator. A rotation fixes only one point of the plane, and thus A can fix at most one vertex. If A fixes an edge as well as a vertex w, then w must lie at the midpoint of the edge, and thus the drawing is degenerate.

Conversely, suppose that $A = \langle\rho\rangle$ is an automorphism group satisfying the conditions of the lemma. Assume, for the moment, that fix_A is empty. Our algorithm places every vertex u on a circle of radius one about the origin; it must choose the angle θ_u that the line between u and the origin makes with the x axis.

From Corollary 3.1, each orbit has the same size. Thus, there are n/k orbits $O_1, O_2, \ldots, O_{n/k}$, where $n = |V|$. We choose an element u_i from O_i and, for $j = 0, 2, \ldots, k-1$, place $\rho^j(u_i)$ so that $\theta_{\rho^j(u_i)} = 2\pi(i + jn/k)/n$. Effectively, this spaces the vertices equally around the circle so that the angle between consecutive elements of the same orbit is $2\pi/k$. Thus the drawing displays A with a rotation by $2\pi/k$. It is clear that this drawing is non-degenerate.

If fix_A is nonempty, then it has one element c, which we place at the origin. This preserves symmetry but introduces a possible degeneracy: the central fixed vertex may lie on an edge that forms a diameter of the circle. However, such an edge is fixed by a rotation by π and, from the conditions of part (b) of the lemma, cannot occur when fix_A is nonempty.

Part (c) Finally, suppose that A is dihedral. Suppose that $A = \langle\alpha, \rho\rangle$, where $\alpha^2 = 1$ and $\rho^k = 1$, with $k \geq 2$. If fix_ρ is nonempty, then it forms a trivial orbit of $\langle\rho\rangle$. Denote the nontrivial orbits of $\langle\rho\rangle$ by $O_1, O_2, \ldots, O_{n/k}$, where $n = |V - fix_A|$.

For $0 \leq i \leq k-1$, the automorphism ρ^i will be displayed as a rotation by $2\pi i/k$ about the origin, and the automorphism $\rho^{-i}\alpha\rho^i$ will be displayed as a reflection in the line through the origin at an angle of $2\pi i/k$ to the x axis.

We will use a circular grid with n rays $R_0, R_1, \ldots, R_{n-1}$ and n/k circles $C_1, C_2, \ldots, C_{n/k}$.

First, we draw $fix_{\rho^{-i}\alpha\rho^i}$ for each i, starting with $i = 0$. We assume first that $fix_\rho = \emptyset$.

Since fix_α is a set of disjoint paths, we can draw it on the x axis so that each vertex is at a grid point of the circular grid and no vertex lies on any edge with which it is not adjacent. If k is even, then we must ensure that $\alpha(fix_{\rho^{-k/2}\alpha\rho^{k/2}}) = fix_{\rho^{-k/2}\alpha\rho^{k/2}}$; this is easily achieved.

Now consider $fix_{\rho^{-i}\alpha\rho^i}$.

Note that if $u \in fix_{\langle\alpha\rangle}$, then $\rho^i(u) \in fix_{\rho^{-i}\alpha\rho^i}$; in other words, if u is fixed by α then every vertex in $orbit_\rho(u)$ is fixed by a conjugate of α. We can draw $fix_{\rho^{-i}\alpha\rho^i}$ on $R_{ni/k}$ and $R_{n(k-i)/k}$ by rotating the drawing of fix_α by $2\pi i/k$. In this way, we draw $orbit_\rho(u)$ for every vertex $u \in fix_\alpha$.

Every other orbit is drawn on the innermost circle C_1. We use a similar method to that for cyclic groups, except that we display α. To do this, we choose a vertex u_1 from an orbit, and draw u_1 on ray R_1. Then draw $\alpha(u_1)$ on ray R_{n-1}. Next, we choose a vertex u_2 from another orbit and draw u_2 on R_2 and $\alpha(u_2)$ on R_{n-2}. This continues until we have placed one vertex from each orbit. To place the remaining vertices from these orbits, we just rotate by $2\pi/k$.

The resulting drawing displays A; we must show that it is not degenerate.

From Lemma 3.1, we can assume that if there is a degeneracy, then there is a vertex w lying on an edge (u, v) with $w \neq u, v$, and the line through u, w, and v passes through the origin. If $u \in fix_\alpha$, then since v and w are on the line through u and the origin, we must have $v, w \in fix_\alpha$. This is impossible since the layout method for fix_α precludes degeneracies. We can deduce that neither u nor v is in $fix_{\rho^{-i}\alpha\rho^i}$ for any i. Hence, we conclude that u and v are on C_1. The only possible degeneracy is if w is the central vertex, fixed by all automorphisms; thus, $fix_A \neq \emptyset$. However, $\langle\rho\rangle$ fixes the edge (u, v), contradicting the conditions of the theorem. This completes the proof of Theorem 3.2.

The proof of Theorem 3.2 essentially consists of an algorithm for the following problem.

Geometric Automorphism Drawing Problem (GADP)

Instance: A graph G, and an automorphism group A of G given as a set of at most 2 generators.

Output: If possible, a straight-line drawing of G that displays A.

COROLLARY 3.2 There is a linear-time algorithm that solves the Geometric Automorphism Drawing Problem.

Note that the resolution of the drawing obtained by the proof of Theorem 3.2 is poor in the dihedral case, because the radii of the circles in the circular grid used increase exponentially.

Theorem 3.2 does not solve the main problem in symmetric graph drawing: given a graph, find its largest geometric automorphism group. The next section shows that it is NP-complete to find such a group.

3.4 Finding Geometric Automorphisms

In this section, we discuss the complexity of computing geometric automorphisms, and, since the problem is NP-complete, we briefly mention heuristics.

The relationship between automorphisms of a graph and symmetries of drawings of the graph suggests that the problem of drawing a graph symmetrically is at least as hard as graph isomorphism. Manning [Man90] has shown a surprisingly stronger result: the problem is NP-hard. The intuition behind Manning's result comes from two directions. First, as noted in Section 3.3, the major difficulties in drawing graphs symmetrically arise from the

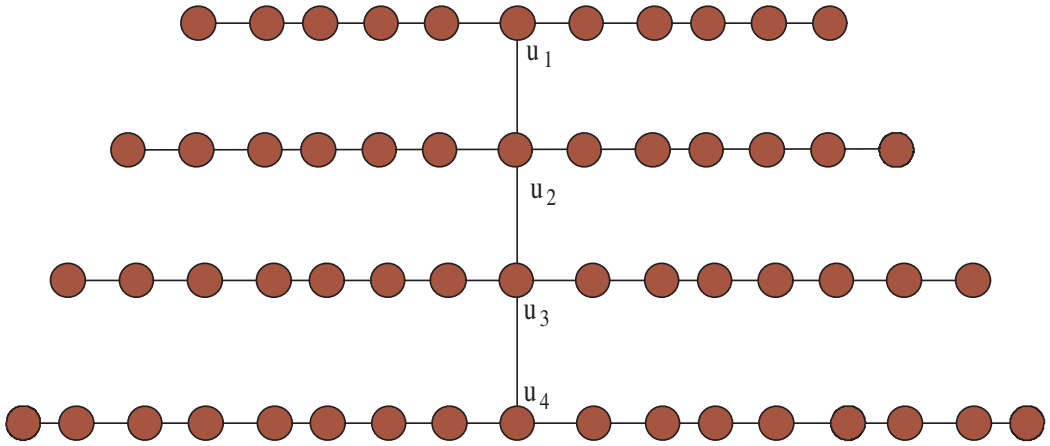

Figure 3.6 The auxiliary graph H.

fixed points of the automorphisms. Secondly, a result of Lubiw [Lub81] states that finding a fixed-point free automorphism of a graph is NP-complete.

In fact, Manning shows that a number of problems related to symmetric graph drawing are NP-hard. Here, we study just one of these problems: detecting whether a graph has an automorphism that can be displayed as a reflection.

Axial Geometric Automorphism Problem (AGAP)
Instance: A graph G.
Question: Is there an automorphism of G that can be displayed as a reflection?

Theorem 3.3 *The axial geometric automorphism problem is NP-complete.*

Proof: Lubiw [Lub81] showed that the following problem is NP-hard.

Fixed Point Free Automorphism Problem (FPFAP)
Instance: A graph G.
Question: Is there an automorphism of G with no fixed points?

We show that FPFAP reduces to AGAP.

Suppose that G is an instance of FPFAP with n vertices. We assume without loss of generality that G is connected and every vertex has degree at least 2. Define a graph H as follows: H has a path $P = (u_1, u_2, \ldots, u_{n+1})$. For $1 \leq i \leq n+1$, u_i is joined to two paths, each of length $n + i$. This is illustrated in Figure 3.6.

Now consider an automorphism β of H. It is clear that for $1 \leq i \leq n+1$, $\beta(u_i) = u_i$, and β either fixes or swaps the two paths joined to u_i. If a drawing D of H displays β, then it is displayed as a reflection, and P lies on the axis of reflection with the paths attached to each u_i on each side of the axis.

Now form a graph G' from H and G. The vertex set is the union of the vertex sets of H and G, plus extra vertices $w_0^v, w_1^v, \ldots, w_n^v$ for each vertex of v of G. For $2 \leq i \leq n$, join u_i to every vertex of G. For each vertex v of G, join w_0^v to v, and join all vertices $w_0^v, w_1^v, \ldots, w_n^v$ together to make a clique of size $n + 1$.

Note that G' can be formed in polynomial time.

We claim that G' has an axial geometric automorphism if and only if G has a fixed point free automorphism.

First, suppose that G has a fixed point free automorphism β. It is clear that one can extend β to G' to give an automorphism that satisfies part (a) of Theorem 3.2, and so G' has an axial geometric automorphism group.

Now suppose that G' has an axial geometric automorphism γ.

We claim that γ cannot map a vertex w of H to a vertex v of G, or to one of the new vertices w_i^v. This is because every vertex of G is adjacent to a clique of size $n+1$, while no vertex of H has this property. Further, γ cannot map a vertex of G to one of the new vertices w_i^v, because each w_i^v is in a clique of size $n+1$, and none of the original vertices have this property. Thus, γ restricted to H is an automorphism β of H; as mentioned above, the only drawing that displays β has P lying on the axis of reflection.

Also, γ restricted to G is an automorphism δ of G. Suppose that δ has a fixed point v. Recall that v is joined by an edge to a vertex u_i in P; this means that the induced subgraph fix_γ has a vertex of degree at least three. From Theorem 3.2, this is impossible. Thus δ is fixed-point-free.

Finally, note that AGAP is in NP, because one can guess an automorphism group, and, using Theorem 3.2, check whether it is geometric. □

The NP-completeness results have led to a number of heuristic approaches; see [dF99, Kam89, Lin92, LNS85].

The most common are the generic *multidimensional scaling*, or *force directed* methods [dF99, Ead84, Kam88, Lin92]. Roughly speaking, this method projects a high-dimensional drawing of the graph into low dimensions. The first step is to define a distance function d between vertices, and then the graph is drawn in a high-dimensional space in such a way that the Euclidean distance in the high-dimensional space is equal or close to the distances defined by d. In some cases, this (high-dimensional) drawing is unique up to isometry; this implies that every automorphism of the graph is a symmetry of the drawing. In other words, it achieves maximum symmetry in the high dimension. The next step is to project the high-dimensional drawing into a low-dimensional space (either 2 or 3 dimensions) in such a way that the distances are preserved as much as possible.

As an example of such a method, de Fraysseix [dF99] uses the *Czekanovski-Dice* semi-distance for a graph $G = (V, E)$:

$$d(u,v) = \sqrt{1 - 2\frac{|N_u \cap N_v|}{|N_u| + |N_v|}}. \tag{3.7}$$

(A *semi-distance* $d : X \to R$ is a function that is almost a distance function: it satisfies two of the axioms of a distance function: $d(u,v) = d(v,u)$ and $d(u,v) + d(v,w) \geq d(u,w)$. However, it is possible that there are distinct elements $u, v \in X$ with $d(u,v) = 0$.) De Fraysseix uses projections defined by the principal components of the corresponding inner product matrix whose entries are defined by a pair s, t of vertices as follows:

$$W_{st} = \frac{1}{2}(d_s^2 + d_t^2 - d_V^2), \tag{3.8}$$

where for $w = s, t$,

$$d_w^2 = \frac{1}{n}\sum_{u \in V} d^2(w,u), \tag{3.9}$$

and

$$d_V^2 = \frac{1}{n}\sum_{w \in V} d_w^2. \tag{3.10}$$

These projections are remarkably successful in displaying two dimensional symmetry; see [dF99] for details.

It is common to look at such methods as a system of forces: for example, one can simulate a system of forces between vertices where the force exerted on u by v is proportional to the distance $d(u, v)$. A minimum energy configuration defines a drawing, and in many cases this drawing displays symmetries. For example, one can view the Tutte method [Tut63, DETT99] in this way. In fact, one of the reasons for the popularity of force directed methods is the fact that the drawings often display some symmetry. One can give some explanation (see [EL00, Lin92]) of why the approach works.

3.5 Symmetric Drawings of Planar Graphs

In this section, we describe a linear-time algorithm to draw planar graphs with no edge crossings and as much symmetry as possible.

The concept of geometric automorphism in Section 3.2.3 can be extended to planar drawings: an automorphism β of a graph G is *planar* if there is a planar drawing of G that displays β, and an automorphism group A is a *planar automorphism group* if there is a planar drawing which displays every element of A.

The problem of finding automorphisms of a planar graph can be solved in linear time (see [HW74, Won75]); however, it is clear that not all automorphisms are geometric. Further, not every geometric automorphism is planar. For example, the complete graph K_4 with four vertices has a dihedral geometric automorphism group of size eight, but this group is not planar. The largest planar automorphism group of K_4 has size six.

The following theorem summarizes the result.

Theorem 3.4 *There is a linear-time algorithm that constructs maximum planar automorphism group of a planar graph.*

The remainder of this section is a sketch of a proof of Theorem 3.4. The algorithm to prove the theorem uses a connectivity decomposition. We decompose the graph into connected components, then decompose each connected component into biconnected components, and finally decompose each biconnected component into triconnected components. Different algorithms are needed for triconnected, biconnected, one-connected, and disconnected graphs. Each uses the algorithms for higher connectivity as subroutines. Details of the proof can be found in [HE05, HE06, HE03, HME06].

In Section 3.5.5, we briefly describe the drawing algorithms.

3.5.1 Triconnected planar graphs

This section describes an algorithm for finding planar automorphism groups of maximum size for triconnected planar graphs.

The uniqueness of the faces of a triconnected planar graph $G = (V, E)$ means that an automorphism group A defines a permutation group acting on the set F of faces of G. Effectively this means that A defines an automorphism group of the dual G^* of G. We can regard A as acting on G^*, and write, for example, "the face $\beta(f)$" for some $\beta \in A$, $f \in F$.

It is well known that a triconnected planar graph can be represented as the skeleton of a polyhedron in three dimensions [SR34]. A more surprising and less well known result, due to Mani [Bab95, Man71], states that the automorphism group of a triconnected planar graph can be completely encapsulated in the symmetries of a polyhedron. The symmetry finding algorithm relies on this fundamental result.

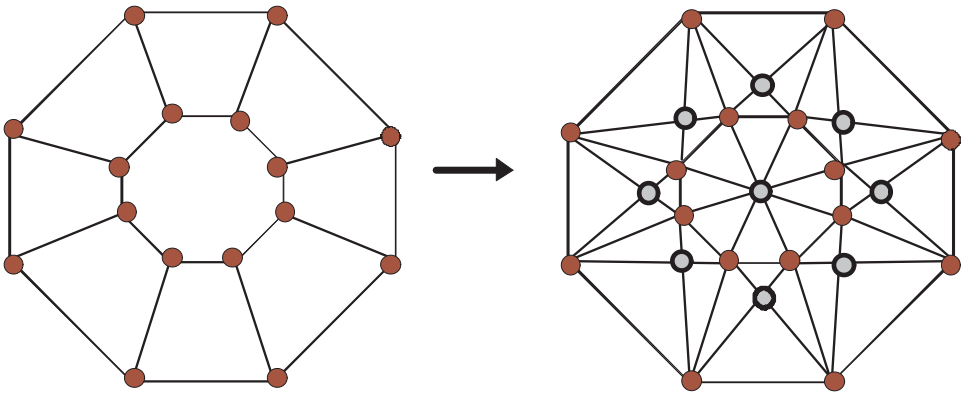

Figure 3.7 Example of star triangulation.

Theorem 3.5 *(Mani [Bab95, Man71]) Suppose that G is a triconnected planar graph. Then there is a convex polytope P in R^3 such that G is the skeleton of P and the full automorphism group of G is displayed by P.*

Mani's theorem leads to an elegant characterization of planar automorphisms of triconnected planar graphs.

Theorem 3.6 *Let G be a triconnected planar graph. An automorphism group of G is planar if and only if it is the stabilizer of a face of G.*

Proof: Every planar automorphism fixes the outside face. Further, if A stabilizes a face f then, using Theorem 3.5, a projection about f from the polyhedron to the plane gives a symmetric drawing. □

An outline of the algorithm for the triconnected case of Theorem 3.2 is as follows.

Algorithm `Max_PAG_tricon`

1. Find a face f of G such that the stabilizer $stab_A(f)$ of f in A is maximized.
2. Find the orbits of $stab_A(f)$.
3. Find generators of $stab_A(f)$.

For the first step, note that from Theorem 3.1, we must find an orbit (in the dual G^*) of minimum size. A linear-time algorithm of Fontet [Fon76] takes a triconnected planar graph G as input and outputs the orbits on vertices of $aut(G)$. Using Fontet's algorithm, we can compute the orbits of G^*, then choose an orbit of minimum size. Choose a face f in this minimum orbit; then f can be used as the outside face of an embedding that displays the maximum number of symmetries.

The next step is to find the orbits of $stab_A(f)$. This can be done by transformations of the graph, and then using Fontet's algorithm again. The first part of the transformation is *star triangulation*: we triangulate each internal face by inserting a new vertex in the face and joining it to each vertex of the face. This process is illustrated in Figure 3.7.

It is not difficult to show that the star triangulation takes linear time, and the new graph has exactly the same planar automorphism group as the original graph (see [HME06]).

Next, we transform the graph to ensure that the outside face f has more than three vertices. If f has three vertices v_0, v_1, v_3, we draw a hexagon surrounding G in the plane, with vertices $w_0, w_1, \ldots, w_5$ in clockwise order. Insert the edges v_0w_0, v_0w_1, v_0w_2, v_1w_2, v_1w_3,

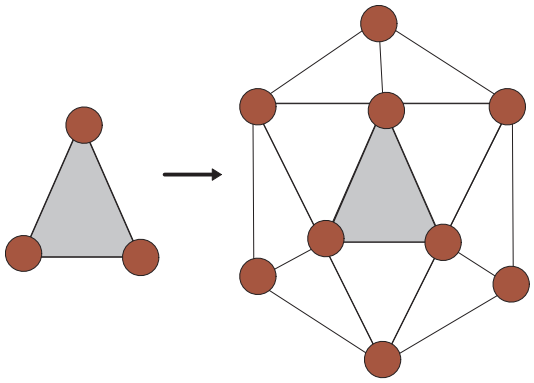

Figure 3.8 Adding an outside face.

v_1w_4, v_2w_4, v_2w_5, and v_2w_0. The transformation is shown in Figure 3.8. The transformation preserves automorphisms that fix f.

The transformed graph has a new outside face with more than three vertices, and all other faces are triangles. Now apply Fontet's algorithm to the transformed graph. The outside face must be fixed by all automorphisms, since all other faces have size three. Thus, Fontet's algorithm gives the orbits of $stab_A(f)$ in the transformed graph, and we can extract the orbits of A on the vertices of G.

The third and final step is to find generators of the planar automorphism group. Suppose that the vertices on the outside face f are $v_0, v_1, \ldots, v_{m-1}$, in clockwise order. If $v_0, v_1, \ldots, v_{m-1}$ are all fixed by A, then A is trivial. Otherwise, let v_i, v_j, v_k be three consecutive vertices in the same nontrivial orbit of A, where $j - i$ is as small as possible and v_k is the same as v_i if the orbit has size 2.

We need to introduce further terminology: a *flag* of an embedded graph is a triple (v, w, f), where v and w are adjacent vertices and f is a face that has the edge (v, w) on its boundary. The action of automorphisms on flags uniquely identifies them, as stated in the following lemma.

LEMMA 3.2 Let G be a triconnected planar graph. Let $F = (v, w, f)$ and $F' = (v', w', f')$ be flags of G. Then there is at most one automorphism of G that maps F onto F'. Moreover, there is a linear-time algorithm that finds that automorphism or determines that it does not exist.

Lemma 3.2 is folklore in graph automorphism theory; a proof is in [HME06].

We can apply Lemma 3.2 to find three possible automorphisms or prove that they do not exist. First, we compute three possible automorphisms α, ρ_1, ρ_2, as follows:

- α is the automorphism mapping the flag (v_i, v_{i+1}, f) onto the flag (v_j, v_{j-1}, f), if that automorphism exists. (That is, a reflection that exchanges v_i and v_j.)
- ρ_1 is the automorphism mapping the flag (v_i, v_{i+1}, f) onto the flag (v_j, v_{j+1}, f), if that automorphism exists. (That is, a rotation by $j - i$ positions.)
- ρ_2 does not exist in the case that $v_k = v_i$. Otherwise, ρ_2 is the automorphism mapping the flag (v_i, v_{i+1}, f) onto the flag (v_k, v_{k+1}, f), if that automorphism exists. (That is, a rotation by $k - i$ positions.)

This allows us to compute generators for A, as follows.

- If α does not exist, then ρ_1 exists and A is a cyclic group of size $m/(j-i)$ generated by the rotation ρ_1.
- If α exists but neither ρ_1 nor ρ_2 exists, then A is the group of size 2 generated by the reflection α.
- If α and ρ_1 exist, then A is the dihedral group of size $2m/(j-i)$ generated by the reflection α and the rotation ρ_1.
- Otherwise, α and ρ_2 exist, and A is the dihedral group of size $2m/(k-i)$ generated by the reflection α and the rotation ρ_2.

We summarize this section with the following lemma.

LEMMA 3.3 Algorithm `Max_PAG_tricon` computes generators for the largest planar automorphism group of a triconnected planar graph in linear time.

3.5.2 Biconnected planar graphs

If the input graph G is biconnected, then we break it into *triconnected components* and apply the algorithm for triconnected graphs in Section 3.5.1. However, this process is not as simple as it sounds.

We use a version of the "SPQR-tree" to represent the decomposition of a biconnected planar graph into triconnected components. Various versions of the SPQR tree appear in the literature; the version that we use is closely related to the original version of Tutte [Tut66].

It is useful to review the definition of triconnected components [HT73]. If G is triconnected, then G itself is the unique triconnected component of G. Otherwise, let u, v be a separation pair of G. We split the edges of G into two disjoint subsets E_1 and E_2, such that $|E_1| > 1$, $|E_2| > 1$, and the subgraphs G_1 and G_2 induced by E_1 and E_2 only have vertices u and v in common. Form the graph G'_1 by adding an edge (called a *virtual edge*) between u and v; similarly, form G'_2. We continue the splitting process recursively on G'_1 and G'_2. The process stops when each resulting graph reaches one of three forms: a triconnected simple graph, a set of three multiple edges (a triple bond), or a cycle of length three (a triangle). The triconnected components of G are obtained from these resulting graphs. They may be of three types:

1. a triconnected simple graph;
2. a *bond*, formed by merging the triple bonds into a maximal set of multiple edges;
3. a *polygon*, formed by merging the triangles into a maximal simple cycle.

The triconnected components of G are unique. See [HT73] for further details.

Now we can describe the SPQR tree. Each node v in the SPQR tree is associated with a graph $skeleton(v)$, corresponding to a triconnected component. There are several types of nodes in the SPQR tree, corresponding to the type of triconnected components described above. The edges of the SPQR tree are defined by the virtual edges, that is, if u and v are two nodes whose skeletons share a virtual edge, then u and v are connected in the SPQR tree.

The SPQR tree can be rooted at its center (if the tree has two centers, it can be rooted at either one). The motivation for using the rooted version is that the SPQR tree is unique for each biconnected planar graph [Bab95, DT92]. This means that the triconnected component corresponding to the root of the SPQR-tree is fixed by a planar automorphism group of a biconnected planar graph. Further, each leaf is mapped to a leaf. These two properties of the rooted SPQR tree are essential for our algorithm outlined below.

To state the algorithm, we need some more terminology. We say that a virtual edge e of *skeleton*(v) is a *parent (child) virtual edge* if e corresponds to a virtual edge of u which is a parent (resp. child) node of v. We define a *parent separation pair* $s = (s_1, s_2)$ of v as the two endpoints of a parent virtual edge e.

The overall algorithm is composed of three steps.

Algorithm `MAX_PAG_bicon`

Step 1. Construct the SPQR-tree T of G.

Step 2. *Reduction*: For each level i of T (from the lowest level to the root level)

(a) For each leaf node on level i, compute labels on the parent virtual edge in the leaf node.

(b) For each leaf node on level i, label the corresponding virtual edge in the parent node with the labels.

(c) Remove the leaf nodes on level i.

Step 3. Compute a maximum size planar automorphism group at the labeled center.

We briefly describe each step of the algorithm. The first step is to construct the SPQR-tree for the input biconnected planar graph. This can be done in linear time using the classical Hopcroft-Tarjan algorithm [HT73].

The second step, reduction, is the most important. This takes the rooted SPQR-tree of a biconnected graph, and proceeds up the SPQR-tree from the leaf nodes to the center level by level, computing labels. The labels consist of integer and boolean values that capture some information of the planar automorphisms of the leaf nodes. First, it computes the labels for the leaf nodes. Then, it labels the corresponding virtual edge in the parent node and delete each leaf node. The reduction process stops when it reaches the root.

The reduction process clearly does not decrease the planar automorphism group of the original graph. This is not enough; we need to also ensure that the planar automorphism group is not increased by reduction. This is the role of the labels. As a leaf v is deleted, the algorithm labels the virtual edge e of v in *skeleton*(u) where u is a parent of v. Roughly speaking, the labels encode enough information about the deleted leaf to ensure that planar automorphisms of the labeled reduced graph can be extended to a planar automorphisms of the original graph.

We illustrate the basic idea of the algorithm with an example.

Consider the biconnected graph represented in Figure 3.9. Here the graph G has an SPQR tree with three leaves; these are triconnected components, G_1, G_2, and G_3, illustrated by shaded blobs. The remainder of the graph, G^*, is illustrated by a shaded oval. This is connected to the leaves by separation pairs $\{u_i, v_i\}$, for $i = 1, 2, 3$..

Intuitively, G can be drawn with an axial symmetry (a reflection in a horizontal line) as long as:

1. L_1 is isomorphic to L_2 with an isomorphism that maps u_1 to u_2 and v_1 to v_2.
2. L_3 has an axial planar automorphism that swaps u_3 with v_3.
3. G^* has an axial planar automorphism that swaps u_3 with v_3, and maps u_1 to u_2 and v_1 to v_2.

To decide whether G can be drawn with an axial symmetry, we maintain a number of labels, including:

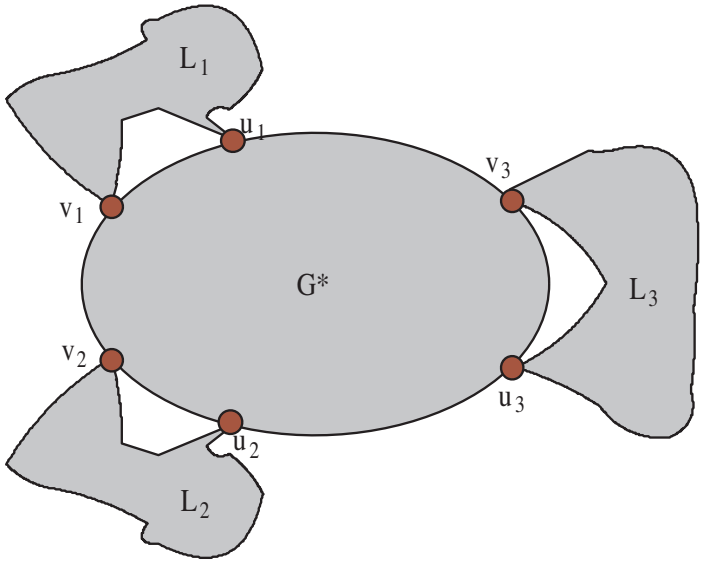

Figure 3.9 A biconnected graph.

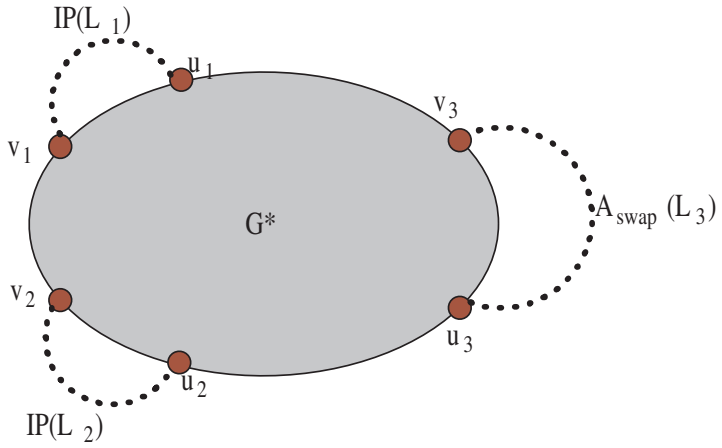

Figure 3.10 Labels on the reduced biconnected graph.

1. An *isomorphism code* IP that has the property that $IP(L_1) = IP(L_2)$ if and only if L_1 is isomorphic to L_2 with an isomorphism that maps u_1 to u_2 and v_1 to v_2.
2. A boolean *axial swap label* A_{swap} that has the property that $A_{swap}(L_3) = true$ if and only if L_3 has an axial planar automorphism that swaps u_3 with v_3.

These labels can be computed at Step 2(a) of Algorithm `MAX_PAG_bicon`, then transferred to the parent virtual edges in G^* at Step 2(b). Then Step 2(c) gives the labeled reduced graph illustrated in Figure 3.10.

The reduction then continues to the next iteration of Step 2, operating on the labeled reduced graph in Figure 3.10. This continues to the root of the SPQR tree.

In fact, the reduction step is much more complex than this example suggests. There are seven different kinds of labels and separate algorithms for computing these labels for each type of triconnected component. Details of these algorithms are in [HE05].

Step 3 of Algorithm `MAX_PAG_bicon` computes a maximum size planar automorphism group at the center, using the information encoded on the labels. Again this step is quite complex, with separate algorithms for computing these labels for each type of triconnected component and each type of center (the center of the SPQR tree can be a node or and edge). Details of these algorithms are in [HE05].

3.5.3 One-connected planar graphs

The algorithm for computing a maximum size planar automorphism group of one-connected planar graph uses a reduction process that is similar to the biconnected case. For one-connected graphs, we take the *block-cut vertex tree* (the *BC-tree*). The BC- tree defines the structure of the biconnected components of a graph. If G is a one-connected graph, then a maximal biconnected subgraph of G is a *block*, or a *biconnected component*. Two blocks share a *cut vertex*. The *BC-tree* has a *B-node* for each block of G and a *C-node* for each cut vertex of G. There is an edge between the B-node B and the C-node c if c is a vertex of B. The BC-tree can be computed in linear time [AHU83].

Again we can choose the center of the BC-tree as a root, and the rooted BC-tree is unique. This property allows a reduction and labeling process similar to that described in the previous section, although the details are very different; see [HE06]. The algorithm uses the algorithms for the biconnected case as subroutines.

3.5.4 Disconnected planar graphs

Drawing disconnected graphs is surprisingly challenging (see, for example, [FDK01]). In this section, we give an intuitive explanation of an algorithm for finding planar automorphisms of a disconnected graph G. The algorithm uses the algorithms for the higher-connectivity cases as subroutines. For the purposes of an intuitive explanation, we consider problems of arranging objects in the plane to maximize symmetry.

First, suppose that we have a set of colored discs, with n_j discs of color j, for $j = 1, 2, \ldots, m$. Each disc is circular and has radius one. We want to arrange the discs in the plane so that no two discs overlap, and the arrangement is as symmetrical as possible. We can make a picture something like a flower: one disc in the center, and the others as "petals." Such an arrangement is in Figure 3.11; here $m = 2$, $n_1 = 4$ and $n_2 = 6$, and the discs are arranged to have a dihedral group of size 6.

The center of the flower may be empty. In this case, all discs must be arranged as petals; if there are k petals, then n_j must divide k for $j = 1, 2, \ldots, m$. If the center of the flower has a disc of color i, $n_i - 1$ divides k, and for $j \neq i$, n_j divides k. We can deduce that the

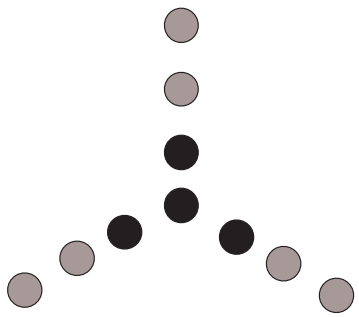

Figure 3.11 A symmetrical arrangement of circular discs.

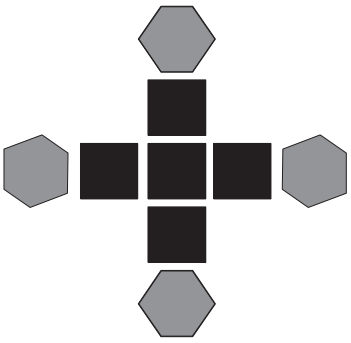

Figure 3.12 A symmetrical arrangement of polygonal discs.

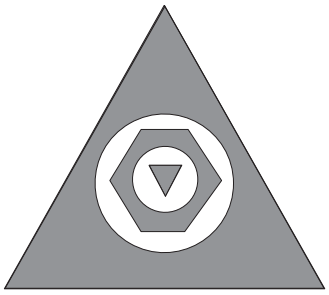

Figure 3.13 Nesting of discs with holes.

maximum symmetry group is dihedral of size $2k$ as long as the following equation holds:

$$k = \max\{\gcd(n_1, n_2, \ldots, n_m), \max_{i=1}^{m} \gcd(n_1, n_2, \ldots, n_{i-1}, n_i - 1, n_{i+1}, \ldots, n_m)\}. \quad (3.11)$$

With some clever computation of the gcds, we can compute equation (3.11) and a maximally symmetric layout of the discs in time $O(n_1 + n_2 + \cdots + n_m)$.

Now consider a problem with a little more complexity. Suppose that we have colored polygonal discs, with n_j discs of color j, for $j = 1, 2, \ldots, m$. Each disc is a regular polygon; all discs of color i have s_i sides, and have radius one. Again, we can make a symmetric picture something like a flower, as in Figure 3.12; here $m = 2$, $n_1 = 5$, $s_1 = 4$, $n_2 = 4$ and $s_2 = 6$.

In this case, we can obtain a dihedral symmetry group of size $2k$ if k satisfies either:

$$k = gcd(n_1, n_2, \ldots, n_m), \quad (3.12)$$

(for the case where the center is empty), or for some i,

$$k = \gcd(s_i, n_1, n_2, \ldots, n_{i-1}, n_i - 1, n_{i+1}, \ldots, n_m)\} \quad (3.13)$$

(for the case where a disc with s_i sides is in the center).

Again, using some clever computation of the gcds and maximizing over i, we can compute a maximally symmetric layout of the discs in time $O(s_1 n_1 + s_2 n_2 + \cdots + s_m n_m)$.

Now consider a more complex problem: suppose that some of the discs have holes. We have n_j discs of color j, for $j = 1, 2, \ldots, m$. The outside of each disc is a regular polygon; all discs of color i have s_i sides. For some values of i, the all discs of color i have a circular hole in the middle. Further, each disc is shrinkable or expandable; this means that we can fit one disc inside another to make a kind of "nest," as in Figure 3.13.

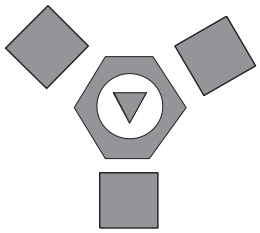

Figure 3.14 Symmetric arrangement of polygonal discs with holes.

Again, we can make a symmetric picture something like a flower, as in Figure 3.14; in this case, we can place a "nest" of discs in the center of the flower, as long as all but one of them have a hole.

Let H denote the set of colors of discs with holes. We can obtain a dihedral symmetry group of size $2k$ if there is a subset H' of H such that k satisfies one of the following:

$$k = \gcd\left(\gcd\{s_j : j \in H'\}, \gcd\{n_\ell : \ell \in H - H'\}\right) \tag{3.14}$$

(for the case where every discs in the center has a hole), or for some i,

$$k = \gcd\left(\gcd\{s_j : j \in H'\}, \gcd\{n_\ell : \ell \in H - H', \ell \neq i\}, s_i, n_i - 1\right), \tag{3.15}$$

for the case where there is a disc of color i, without a hole, in the center.

One can maximize over i and H' to compute a maximally symmetric layout of the colored polygonal discs, with and without holes, in time $O(s_1 n_1 + s_2 n_2 + \cdots + s_m n_m)$.

One can use such disc arrangement algorithms to construct maximally symmetric drawings of disconnected graphs. We can compute the connected components G_ℓ of a disconnected graph G and, using planar graph isomorphism algorithms, divide the components into isomorphism classes $N_1, N_2, \ldots, N_m$, where $|N_j| = n_j$. We compute maximal planar automorphism groups for G_j using the algorithm for connected graphs; assume for the moment that these groups are dihedral and the group for isomorphism class N_j has size $2s_j$. For the purposes of symmetric layout, the isomorphism class N_j is akin to a color class of polygonal disc with s_j sides. For some j, it is possible that the components in N_j has two faces fixed by their planar automorphism group. This is akin to a disc with a hole, because one fixed face can be the outside face and the other can be a central inside face.

There are some further complexities. First, some of the components may have no dihedral planar automorphism group: the group may be purely cyclic, or purely axial, or even trivial. This requires algorithms that are substantially more complex, but follow the same general pattern as above.

Secondly, the connected components may have several maximal planar automorphism groups, and the largest of these may not lead to the maximum planar automorphism group of the whole graph. An example is in Figure 3.15: the two pictures here show a graph with two drawings, one displaying 6 symmetries and one displaying 8 symmetries.

We say that a planar automorphism group A of G is *maximal* if A is not contained in another planar automorphism group of G. One must take all maximal groups into account when this graph is a connected component of a larger disconnected graph. Fortunately, this pathological case is relatively contained; the next Lemma explains why.

LEMMA 3.4 **[HE03]** A planar graph has at most 3 non-conjugate maximal planar automorphism groups.

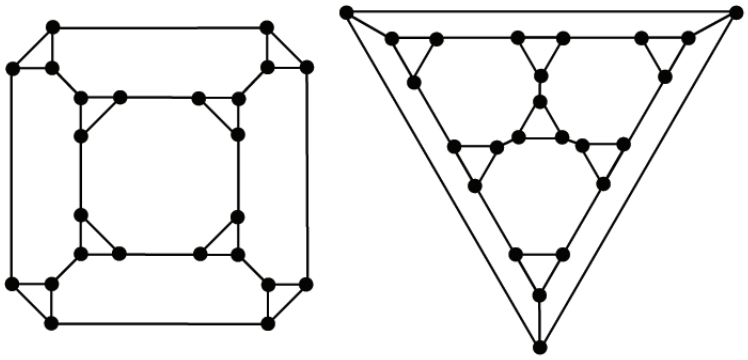

Figure 3.15 Display of two maximal planar automorphism groups.

This means that additional maximal planar automorphism groups only add a constant to the time complexity of the algorithms.

3.5.5 Drawing algorithms

The algorithms presented in the preceding sections take a planar graph as input and produce two outputs: a planar automorphism group of maximum size, and an embedding of the graph. In this section, we show how to use this information to construct a straight-line symmetric drawing of the graph. The drawing algorithms follow the same connectivity hierarchy.

For triconnected graphs, one could use the well-known barycenter algorithm of Tutte [Tut63, DETT99]. This algorithm draws symmetrically but unfortunately takes super-linear time. A much more complex algorithm, described in [HME06], runs in linear time. Note that the drawing can be "squashed" at a specified vertex on the outer face; that is, given an angle a and a vertex u on the outer face, we can adjust the drawing so that the angle at u on the outer face is at most a. The squashing can be done so that any axial symmetry that fixes a is preserved. This process, illustrated in Figure 3.16, is helpful for lower connectivity drawings.

For a biconnected planar graph, we use "augmentation": we increase the connectivity by adding new edges and new vertices to make it triconnected, while preserving the planar automorphism group. The easiest way to do this is to use the star triangulation method described in Section 3.5.1. Then we can apply the algorithm for constructing symmetric drawings of triconnected planar graphs with straight-line edges to construct a symmetric drawing.

Given an embedding of a one-connected planar graph, we use "attachment," as follows. First, we augment the biconnected component to make them triconnected, as above, and draw the triconnected components. Then we draw the root of the BC-tree; then we traverse the BC-tree "attaching" blocks as we go. We can scale blocks to fit inside faces of previously drawn blocks, using the "squash" operation described above.

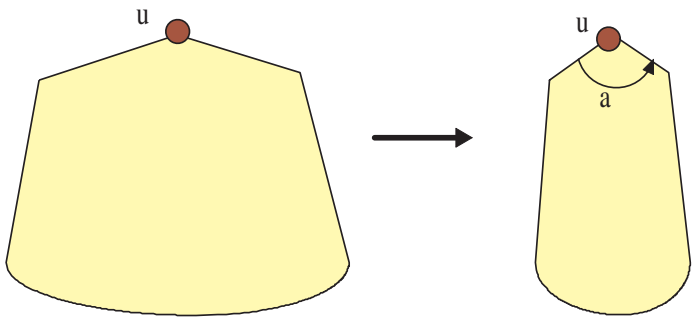

Figure 3.16 Squashing a triconnected component at u.

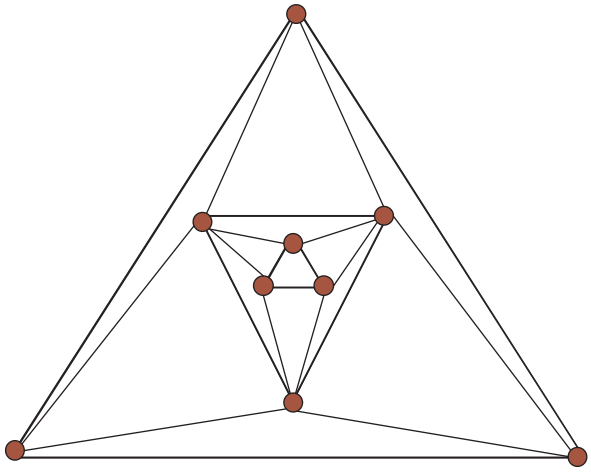

Figure 3.17 The graph G_3.

The drawing process takes linear time, and we can state the following result.

Theorem 3.7 *Given a planar graph G and a planar automorphism group A of G, we can construct a straight-line drawing of G that displays A in linear time.*

The drawings obtained in this way have poor resolution. Unfortunately, in the worst case, this is unavoidable, as the following example shows. Suppose that G_0 is a single triangle with vertices a_0, b_0, c_0. For $i > 0$, G_i is a planar graph with a triangular outside face $\{a_i, b_i, c_i\}$. We form G_i from G_{i-1} by adding the face $\{a_i, b_i, c_i\}$ and the edges $(a_i, a_{i-1}), (a_i, b_{i-1}), (b_i, b_{i-1}), (b_i, c_{i-1}), (c_i, c_{i-1}), (c_i, a_{i-1})$. The graph G_3 is shown in Figure 3.17.

The graph G_k has $3k$ vertices and has a dihedral planar automorphism group of size 6. However, one can show that every straight-line drawing of G_k that displays this dihedral group requires exponential area; that is, if it has a minimum distance of one between vertices, then the area of the drawing is $\Omega(2^k)$.

3.6 Conclusion

This chapter describes the symmetric graph drawing problem, and discusses some of its qualitative and algorithmic aspects. In particular, we characterize those automorphism groups that can be displayed as symmetries of a graph drawing, we show that the gen-

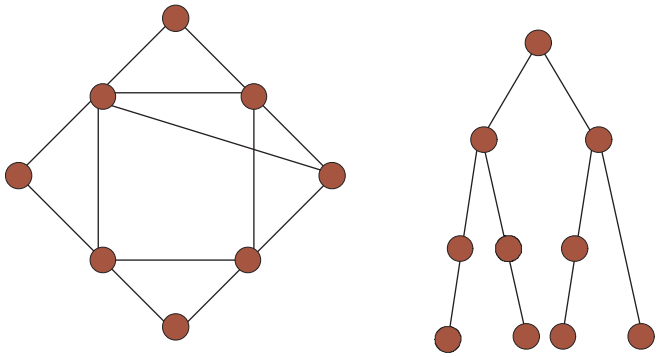

Figure 3.18 Almost symmetric drawings.

eral problem of finding such automorphisms is NP-complete, and we describe linear-time algorithms for finding and displaying such symmetries in the case where the input graph is planar.

In this section, we briefly mention some important aspects of symmetric graph drawing that have not been covered in this chapter and conclude with some open problems.

3.6.1 Further topics

Directed graphs. The model of symmetry needs some modification for directed graphs; for example, perhaps a *directed* geometric symmetry should either preserve the direction of every directed edge or reverse the direction of every directed edge. With a variety of modifications of the model, a number of algorithms have been developed for symmetrically directed graphs. Examples include algorithms for rooted trees [RT81, SR83], series-parallel digraphs [DETT99, HEL00], upward planar graphs [DTT92], and hierarchical graphs [ELT96].

Three-dimensional graph drawing is now well established and some attempts have been made to draw graphs symmetrically in 3D; see [HEQL98, HE00, Hon01].

Exact but exponential time algorithms often work well for small graphs. These include methods based on integer linear programming [BJ01, BJ03] and group theory [AHT07].

Approximation algorithms. The formal definition of the intuitive notion of symmetry display given in Section 3.2.3 is fairly strong. For example, it does not consider the drawings in Figure 3.18 to be symmetric at all. There have been several attempts to formalize the intuitive "approximate" symmetry such as shown in Figure 3.18. For example, Bachl [Bac99] gives a simple approach to approximate axial symmetry: if a graph has two large disjoint isomorphic subgraphs, then one can draw it so that a large part of the drawing displays axial symmetry. Finding such subgraphs is, of course, NP-complete; Bachl gives algorithms for some restricted cases. Other examples include [BJ03, CY02, CLY00].

3.6.2 Open problems

Here we list a couple of open problems in symmetric graph drawing.

Very very symmetric graph drawing. Consider the two drawings in Figure 3.19. The two drawings, according to the model in Section 3.2.3, have the same degree

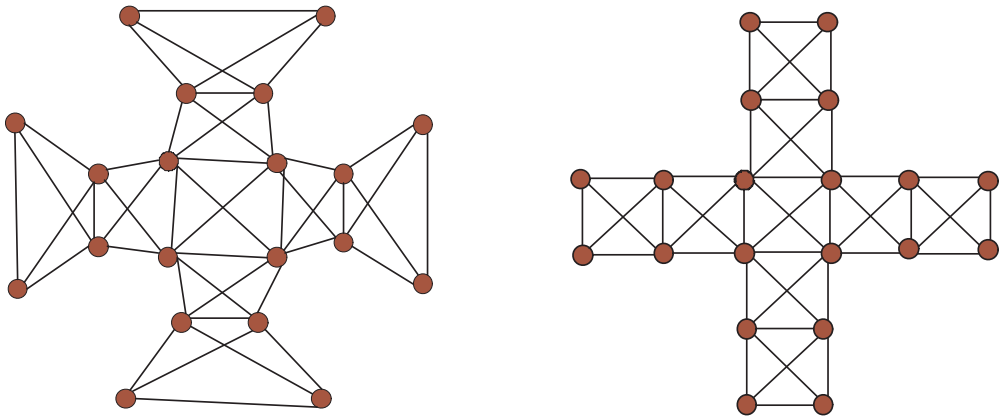

Figure 3.19 A symmetric drawing and a very very symmetric drawing.

of symmetry. However, intuitively the one on the right is more symmetric than the one on the left. The extra symmetry does not come from isometry of the plane; it arises in a more subtle way. Modeling this kind of "very very symmetric" drawing has not been done at this point. Further, algorithms to draw graphs very very symmetrically have not been designed.

An algorithmic version of Mani's Theorem. Theorem 3.5 is one of the most beautiful results in graph drawing. It is not clear how to make Mani's proof [Bab95, Man71] into an algorithm. It would be very interesting to find a linear-time algorithm that takes a triconnected planar graph as input and draws it as the skeleton of a convex polyhedron so that every automorphism of the graph is a symmetry of the polyhedron.

Acknowledgments

This work has been supported by the Australian Research Council. Parts of this chapter were written when the authors were visiting the University of Kyoto under Grant-in-Aid 16092101 for Scientific Research on Priority Areas from the Ministry of Education, Culture, Sports, Science and Technology of Japan.

References

[AHT07] David Abelson, Seok-Hee Hong, and Donald E. Taylor. Geometric automorphism groups of graphs. *Discrete Applied Mathematics*, 155(17):2211–2226, 2007.

[AHU83] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *Data Structures and Algorithms*. Addison-Wesley, Reading, MA, 1983.

[AM88] M. J. Atallah and J. Manning. Fast detection and display of symmetry in embedded planar graphs, 1988.

[Arm88] M. A. Armstrong. *Groups and Symmetry*. Springer-Verlag, 1988.

[Bab95] L. Babai. Automorphism groups, isomorphism, and reconstruction. In Groetschel Graham and Lovasz, editors, *Handbook of Combinatorics*, volume 2, chapter 27. Elsevier Science, 1995.

[Bac99] Sabine Bachl. Isomorphic subgraphs. In Kratochvíl [Kra99], pages 286–296.

[BJ01] Christoph Buchheim and Michael Jünger. Detecting symmetries by branch & cut. In Mutzel et al. [MJL02], pages 178–188.

[BJ03] Christoph Buchheim and Michael Jünger. An integer programming approach to fuzzy symmetry detection. In Giuseppe Liotta, editor, *Graph Drawing*, volume 2912 of *Lecture Notes in Computer Science*, pages 166–177. Springer, 2003.

[CLY00] Ho-Lin Chen, Hsueh-I Lu, and Hsu-Chun Yen. On maximum symmetric subgraphs. In Marks [Mar01], pages 372–383.

[CY02] Ming-Che Chuang and Hsu-Chun Yen. On nearly symmetric drawings of graphs. In *IV*, pages 489–, 2002.

[DETT99] G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis. *Graph Drawing*. Prentice Hall, Upper Saddle River, NJ, 1999.

[dF99] Hubert de Fraysseix. An heuristic for graph symmetry detection. In Kratochvíl [Kra99], pages 276–285.

[DT92] G. Di Battista and R. Tamassia. On-line planarity testing. Report CS-92-39, Comput. Sci. Dept., Brown Univ., Providence, RI, 1992.

[DTT92] G. Di Battista, R. Tamassia, and I. G. Tollis. Area requirement and symmetry display of planar upward drawings. *Discrete Comput. Geom.*, 7(4):381–401, 1992.

[Ead84] P. Eades. A heuristic for graph drawing. *Congr. Numer.*, 42:149–160, 1984.

[EL00] Peter Eades and Xuemin Lin. Spring algorithms and symmetry. *Theor. Comput. Sci.*, 240(2):379–405, 2000.

[ELT96] P. Eades, X. Lin, and R. Tamassia. An algorithm for drawing a hierarchical graph. *Internat. J. Comput. Geom. Appl.*, 6:145–156, 1996.

[FDK01] Karlis Freivalds, Ugur Dogrusöz, and Paulis Kikusts. Disconnected graph layout and the polyomino packing approach. In Mutzel et al. [MJL02], pages 378–391.

[Fon76] M. Fontet. Linear algorithms for testing isomorphism of planar graphs. In *Proceedings Third Colloquium on Automata, Languages, and Programming*, pages 411–423, 1976.

[HE00] Seok-Hee Hong and Peter Eades. An algorithm for finding three dimensional symmetry in trees. In Marks [Mar01], pages 360–371.

[HE03] Seok-Hee Hong and Peter Eades. Symmetric layout of disconnected graphs. In Toshihide Ibaraki, Naoki Katoh, and Hirotaka Ono, editors, *ISAAC*, volume 2906 of *Lecture Notes in Computer Science*, pages 405–414. Springer, 2003.

[HE05] Seok-Hee Hong and Peter Eades. Drawing planar graphs symmetrically, ii: Biconnected planar graphs. *Algorithmica*, 42(2):159–197, 2005.

[HE06] Seok-Hee Hong and Peter Eades. Drawing planar graphs symmetrically, iii: One-connected planar graphs. *Algorithmica*, 44(1):67–100, 2006.

[HEL00] Seok-Hee Hong, Peter Eades, and Sang Ho Lee. Drawing series parallel digraphs symmetrically. *Comput. Geom.*, 17(3-4):165–188, 2000.

[HEQL98] Seok-Hee Hong, Peter Eades, Aaron J. Quigley, and Sang Ho Lee. Drawing algorithms for series-parallel digraphs in two and three dimensions. In *Graph Drawing*, pages 198–209, 1998.

[HME06] Seok-Hee Hong, Brendan D. McKay, and Peter Eades. A linear time algorithm for constructing maximally symmetric straight line drawings of triconnected planar graphs. *Discrete & Computational Geometry*, 36(2):283–311, 2006.

[Hon01] Seok-Hee Hong. Drawing graphs symmetrically in three dimensions. In Mutzel et al. [MJL02], pages 189–204.

[HT73] J. Hopcroft and R. E. Tarjan. Dividing a graph into triconnected components. *SIAM J. Comput.*, 2(3):135–158, 1973.

[HW74] J. E. Hopcroft and J. K. Wong. Linear time algorithm for isomorphism of planar graphs. In *Proc. of the Sixth Annual ACM Symposium on Theory of Computing*, pages 172–184, 1974.

[Kam88] T. Kamada. *On Visualization of Abstract Objects and Relations*. PhD thesis, Department of Information Science, University of Tokyo, 1988.

[Kam89] T. Kamada. Symmetric graph drawing by a spring algorithm and its applications to radial drawing. Technical report, Department of Information Science, University of Tokyo, 1989.

[KK89] T. Kamada and S. Kawai. An algorithm for drawing general undirected graphs. *Inform. Process. Lett.*, 31:7–15, 1989.

[Kra99] Jan Kratochvíl, editor. *Graph Drawing, 7th International Symposium, GD'99, Stirín Castle, Czech Republic, September 1999, Proceedings*, volume 1731 of *Lecture Notes in Computer Science*. Springer, 1999.

[Lin92] X. Lin. *Analysis of Algorithms for Drawing Graphs*. PhD thesis, Department of Computer Science, University of Queensland, 1992.

[LNS85] R. J. Lipton, S. C. North, and J. S. Sandberg. A method for drawing graphs. In *Proc. 1st Annu. ACM Sympos. Comput. Geom.*, pages 153–160, 1985.

[Lub81] Anna Lubiw. Some np-complete problems similar to graph isomorphism. *SIAM J. Comput.*, 10(1):11–21, 1981.

[MA86] J. Manning and M. J. Atallah. Fast detection and display of symmetry in outerplanar graphs. Technical Report CSD-TR-606, Department of Computer Science, Purdue University, 1986.

[MA88] J. Manning and M. J. Atallah. Fast detection and display of symmetry in trees. *Congr. Numer.*, 64:159–169, 1988.

[Man71] P. Mani. Automorphismen von polyedrischen graphen. *Math. Annalen*, 192:279–303, 1971.

[Man90] J. Manning. *Geometric Symmetry in Graphs*. PhD thesis, Purdue Univ., 1990.

[Mar01] Joe Marks, editor. *Graph Drawing, 8th International Symposium, GD 2000, Colonial Williamsburg, VA, USA, September 20-23, 2000, Proceedings*, volume 1984 of *Lecture Notes in Computer Science*. Springer, 2001.

[MJL02] Petra Mutzel, Michael Jünger, and Sebastian Leipert, editors. *Graph Drawing, 9th International Symposium, GD 2001 Vienna, Austria, September 23-26, 2001, Revised Papers*, volume 2265 of *Lecture Notes in Computer Science*. Springer, 2002.

[RT81] E. Reingold and J. Tilford. Tidier drawing of trees. *IEEE Trans. Softw. Eng.*, SE-7(2):223–228, 1981.

[SR34] E. Steinitz and H. Rademacher. *Vorlesungen über die Theorie der Polyeder*. Julius Springer, Berlin, Germany, 1934.

[SR83] K. J. Supowit and E. M. Reingold. The complexity of drawing trees nicely. *Acta Inform.*, 18:377–392, 1983.

[Tut63] W. T. Tutte. How to draw a graph. *Proceedings London Mathematical Society*, 13(52):743–768, 1963.

[Tut66] W. T. Tutte. *Connectivity in Graphs*. University of Toronto Press, 1966.

[Wie64] Wielandt. *Finite Permutation Groups*. Academic Press, 1964.

[Won75] J. K. Wong. *Isomorphism Problems Involving Planar Graphs*. PhD thesis, Cornell Univ., 1975.