CHAPTER IV

ISOMORPHISM OF TRIVALENT GRAPHS AND OF CONE GRAPHS OF DEGREE TWO

Considering the length of time the graph isomorphism problem has remained open despite extensive work, it is natural to seek more tractable restrictions of the problem. We now consider such a restriction and develop polynomial time isomorphism tests for graphs of fixed valence.

We define the valence of a vertex as the number of edges incident to it and the valence of a graph $X$ as the maximum valence of its vertices.

Assume we have an efficient isomorphism test for graphs of a constant valence $k$, and we now wish to test isomorphism of graphs of valence $k+1$. We could try to reduce this problem to testing isomorphism of graphs of valence $k$ by substituting for each vertex of valence $k+1$ a suitable subgraph $G'$ which transforms the graph into a graph of valence $k$ while preserving isomorphism. There is a general result which states that no such subgraph $G'$ can exist except for reducing graphs of valence 5 to graphs of valence 4. For this reason, we believe that testing isomorphism of graphs of fixed valence is not an isomorphism complete problem (see also Chapter II, Section 2).

In this chapter, we give three polynomial time isomorphism tests for trivalent graphs, i.e., graphs of valence three. There are simple polynomial time isomorphism tests for graphs of valence one and two, so the trivalent graphs present the first non-trivial case. We first outline the basic algorithm, omitting the crucial central step of taking the setwise stabilizer in certain permutation groups. We then show how to implement this central step in two entirely different ways. One method applies the results of Chapter III, the other uses a new recursive procedure for computing setwise stabilizers in p-groups.

Our best algorithm is an $O(n^4)$ isomorphism test for trivalent graphs. In it we depart slightly from the basic approach taken by the first two methods. In particular, we improve the computational techniques for handling p-groups, do not take setwise stabilizers, and introduce a new type of valence reduction which is possible only because of certain properties of trivalent graphs.
The basic approach leading to the first two algorithm generalizes to graphs of higher valence. What makes trivalent graphs a special case is that the permutation groups in which the setwise stabilizer has to be determined are always 2-groups. For higher valence, more general types of permutation groups must be considered.

Most of the techniques used in the isomorphism test for graphs of valence \( k \) can also be used for a wider class of graphs. In particular, the isomorphism test for graphs of valence \( k \) may be used to test isomorphism of cone graphs of degree \( k-1 \). In this chapter, we will also consider cone graphs of degree two, called binary cone graphs. Note, however, that the \( O(n^4) \) isomorphism test for trivalent graphs is an exception and cannot be used for all binary cone graphs.

1. Basic Approach

1.1. Properties of the Automorphism Group

Recall that the distance of a vertex \( u \) from a vertex \( v \) in the graph \( X \) is the length of a shortest path between \( u \) and \( v \). Similarly, if \( e = (v_1,v_2) \) is an edge of \( X \), we define the distance of a vertex \( u \) from the edge \( e \) as the smaller of the distances of \( u \) from \( v_1 \) and of \( u \) from \( v_2 \). Note that \( v_1 \) and \( v_2 \) have distance 0 from \( e \).

The isomorphism tests of this chapter are based on the observation that the automorphism group of a trivalent graph (and of a binary cone graph) is almost a 2-group (cf. Chapter III, Definition 7).

**Theorem 1 (Tutte)**

Let \( X = (V,E) \) be a connected trivalent graph, \( e = (v_1,v_2) \) an edge of \( X \). Then \( \text{Aut}_e(X) \), the group of all those automorphisms of \( X \) which stabilize the edge \( e \), is a 2-group.

**Proof** Let \( V_k \) be the vertices of \( X \) at distance \( k \) from \( e \). Note that \( \text{Aut}_e(X) \) is the setwise stabilizer of \( V_0 \) in the full automorphism group of \( X \), and that \( \text{Aut}_e(X) \) must also setwise stabilize the sets \( V_j \). Generalizing Definition 4 of Chapter III, we define \( A^{(k)}(X) \) as the pointwise stabilizer in \( \text{Aut}(X) \) of all vertices in \( V_j, j \leq k \). Note that \( A^{(0)} \) has index 1 or 2 in \( \text{Aut}_e(X) \). We will prove that the index of \( A^{(k+1)} \) in \( A^{(k)} \), \( k \geq 0 \), is a power of 2, from which the theorem follows.

Let \( u \) be an arbitrary vertex in \( V_{k+1} \). There is at least one vertex \( u' \in V_k \) such that \((u,u')\) is an edge of \( X \). Now if \( k = 0 \), then there is another vertex \( u'' \in V_k \) such that
(u', u'') is in E; otherwise, there is a vertex u'' ∈ V_{k+1} such that (u', u'') is in E. We consider the orbit of u in \( A^k \). Since u' is fixed in A^k, every vertex w in the orbit of u must satisfy w ∈ V_{k+1} and (w, u') ∈ E. Since the valence of u' is at most 3 and u'' ∉ V_{k+1}, the orbit of u in A^k has length either 1 or 2. Since \( A^{k+1} \) is obtained from \( A^k \) by successively stabilizing vertices in V_{k+1}, the index \( (A^k : A^{k+1}) \) must be a power of 2 (see Chapter II, Theorem 3).

By the same considerations, we readily obtain

**Corollary 1**

Let \( X = (V, E) \) be a binary cone graph with root v. Then Aut_v(X) is a 2-group.

In Chapter V, we show how to generalize these results to graphs of fixed valence and to cone graphs of fixed degree. In this chapter, we will consider the following two problems:

**Problem 1**

Given a connected trivalent graph X and an edge e of X, determine generators for Aut_e(X).

**Problem 2**

Given a binary cone graph X with root v, determine generators for Aut_v(X).

Clearly, a polynomial time solution for Problem 2 gives us a polynomial time isomorphism test for binary cone graphs. Furthermore, a polynomial time solution for Problem 1 will give us a polynomial time isomorphism test for connected trivalent graphs. To see this, let X and X' be two connected trivalent graphs to be tested for isomorphism. Consider the following procedure.

Pick an edge e of X. For every edge e' of X' do the following: Divide e by inserting as midpoint a new vertex z, and likewise divide e' inserting a new vertex z'. Add the edge (z, z'), as shown in Figure 1 below. For the resulting graph Z we determine Aut_{(z, z')}(Z). Note that there is an isomorphism mapping the edge e to the edge e' iff every generating set of Aut_{(z, z')}(Z) contains a permutation exchanging z with z'. Observe also that the graph Z depends on the choice of the edges e and e'.

Now it is also clear that a polynomial time solution to Problem 1 gives a polynomial time isomorphism test for trivalent graphs which are not connected: Split the graphs X and X' to be tested for isomorphism into connected components; each
component is then a connected graph of valence at most three. Classify these components into isomorphism classes. Then \( X' \) and \( X \) are isomorphic iff exactly half of the components in each isomorphism class belong to the graph \( X \).

1.2. Overall Structure of the Algorithm

We outline the basic approach to solving Problem 1. Let \( X = (V, E) \) be a connected trivalent graph, \( e = (v_1, v_2) \) an edge of \( X \), and assume we wish to determine \( \text{Aut}_e(X) \). Let \( V_k \) be the set of vertices of distance \( k \) from \( e \), and let \( h \) be the height of \( X \), i.e., let \( V = V_0 \cup V_1 \cup \cdots \cup V_h \). We define subsets \( E_k \) of edges of \( X \), \( 0 \leq k \leq h \), by

\[
E_k = \{ (u, w) \mid u \in V_k, w \in V_k \cup V_{k+1} \}
\]

assuming that \( V_{h+1} = \emptyset \). We now define the graphs \( X_j \), \( 0 \leq j \leq h+1 \):

\[
\begin{align*}
X_0 &= (V_0, \emptyset) \\
X_1 &= (V_0 \cup V_1, E_0) \\
X_2 &= (V_0 \cup V_1 \cup V_2, E_0 \cup E_1) \\
\vdots \\
X_h &= (V, E_0 \cup \cdots \cup E_{h-1}) \\
X_{h+1} &= (V, E) = X
\end{align*}
\]

We wish to determine \( \text{Aut}_e(X_k) \) for each graph \( X_k \).

**Example 1**

Let \( X = (V, E) \), where \( V = \{1, \ldots, 12\} \), \( E = \{(1,2), (1,3), (1,4), (2,5), (2,6), (3,4), (3,7), (4,8), (5,9), (5,10), (6,9), (6,10), (7,11), (7,12), (8,11), (8,12), (9,11), (10,12)\} \). \( X \) is a connected trivalent graph. Let \( e = (1,2) \). Then \( V_0 = \{1,2\} \), \( V_1 = \{3,4,5,6\} \), \( V_2 = \{7,8,9,10\} \), and \( V_3 = \{11,12\} \). Also, \( E_0 = \{(1,2), (1,3), (1,4), (2,5), (2,6)\} \), \( E_1 = \{(3,4), (3,7), (4,8), (5,9), (6,9), (6,10), (7,11), (7,12), (8,11), (8,12), (9,11)\} \), and \( E_2 = \{(4,8), (5,9), (6,9), (6,10), (7,11), (7,12), (8,11), (8,12), (9,11), (10,12)\} \).
(5,10), (6,9), (6,10), \( E_2 = \{(7,11), (7,12), (8,11), (8,12), (9,11), (10,12)\} \), and \( E_3 \) is empty. The graphs \( X_1 \) and \( X_2 \) are shown in Figures 2 and 3 below.

The graph \( X_1 \)

Figure 2

The graph \( X_2 \)

Figure 3

Note that \( \text{Aut}_e(X_0) \) is always \( \langle (v_1,v_2) \rangle \). We will determine \( \text{Aut}_e(X) = \text{Aut}_e(X_{h+1}) \) by determining generators for \( \text{Aut}_e(X_{k+1}) \) from generators for \( \text{Aut}_e(X_k) \), \( 0 \leq k \leq h \). Let \( A = \text{Aut}_e(X_k) \). The method for determining \( \text{Aut}_e(X_{k+1}) \) divides into two parts. First, determine the subgroup \( B \) of \( A \) consisting of all permutations in \( A \) which may be extended to an automorphism of \( \text{Aut}_e(X_{k+1}) \), and extend \( B \) to the larger permutation domain. Second, we determine generators for the pointwise stabilizer of all vertices in \( V_0 \cup \cdots \cup V_k \) in the group \( \text{Aut}_e(X_{k+1}) \). We will see that the first step can be reduced to finding the setwise stabilizer in 2-groups, whereas the second step, due to the definition of the edge set \( E_{k+1} \), can be accomplished by inspection.

Specifically, we proceed as follows: Consider the group \( \overline{A} \) obtained by restricting \( A = \text{Aut}_e(X_k) \) to the vertex set \( V_k \). Generators for \( \overline{A} \) are readily obtained from the generators for \( A \) since \( A \) stabilizes \( V_k \) setwise. We now determine the subgroup \( \overline{B} \) consisting of those permutations in \( \overline{A} \) which may be extended to automorphisms of the graph \( (V_k \cup V_{k+1}, E_k) \). Since \( \overline{B} \) is a subgroup of \( \overline{A} \), every element of \( \overline{B} \) may be extended to an automorphism in \( \text{Aut}_e(X_{k+1}) \). Note that \( \overline{B} \) is the restriction of the group \( \text{Aut}_e(X_{k+1}) \) to \( V_k \). We remark for the present that the extension of an element of \( \overline{B} \) to an element of
\( \text{Aut}_e(X_{k+1}) \) is a simple matter.

Now assume that we have extended every generator of \( \overline{B} \) to an element of \( \text{Aut}_e(X_{k+1}) \). We have to add generators for the normal subgroup of \( \text{Aut}_e(X_{k+1}) \) fixing pointwise the vertices in \( V_k \). These additional generators come from two sources: automorphisms which fix \( V_k \cup V_{k+1} \) pointwise, and automorphisms which fix \( V_0 \cup \cdots \cup V_k \) pointwise. The former must be a generating set for the pointwise stabilizer of \( V_k \) in \( \text{Aut}_e(X_k) \). It is easily obtained using the techniques of Chapter II. Using the notation of Section 2 of Chapter III, we note that the latter set generates the group \( \Lambda^{(k)}(X_{k+1}) \), and is obtained by inspection of the edge set \( E_k \), as we will now explain.

Let us call an edge \((u,w)\) of \( X \) a cross edge if \( u \) and \( w \) have the same distance from \( e \). We observe that \( X_{k+1} \) has no cross edges connecting vertices in \( V_{k+1} \). So the generators for this group are transpositions \((u,w)\), \( u, w \in V_{k+1} \), where for each edge \((u,z)\) in \( E_k \) there is an edge \((w,z)\) in \( E_k \), and vice versa. (Note that \( z \) is a vertex in \( V_k \).)

The union of these three sets clearly generates \( \text{Aut}_e(X_{k+1}) \).

\textbf{Example 2}

Consider the graph \( X \) of Example 1. We sketch the determination of \( \text{Aut}_e(X_2) \) from \( \text{Aut}_e(X_1) \), where \( e \) is the edge \((1,2)\). We find here that the group \( \text{Aut}_e(X_1) \) is \( A = \langle (1,2)(3,5)(4,6), (3,4) \rangle \). The group \( \overline{A} \), therefore, contains the following eight permutations: \( (), (3,4), (5,6), (3,4)(5,6), (3,5)(4,6), (3,6)(4,5), (3,6,4,5), (3,5,4,6) \).

Since there is the edge \((3,4)\) but not an edge \((5,6)\) in \( E_2 \), we find that only the following four permutations in \( \overline{A} \) are also in \( \overline{B} \): \( (), (3,4), (5,6), (3,4)(5,6) \). Thus, \( \overline{B} = \langle (3,4), (5,6) \rangle \), an elementary Abelian group of order 4. The generators of \( \overline{B} \) may be extended to the permutations \((3,4)(7,8)\), and \((5,6)(9,10)\) in \( \text{Aut}_e(X_2) \). Next, inspecting the graph \( X_2 \), we find that the transposition \((9,10)\) is the only generator for \( \Lambda^{(1)}(X_2) \). Furthermore, the pointwise stabilizer of \( \{3,4,5,6\} \) in \( \text{Aut}_e(X_1) \) is the trivial group. We therefore conclude that \( \text{Aut}_e(X_2) = \langle (3,4)(7,8), (5,6)(9,10), (9,10) \rangle \).

1.3. Reduction to the Setwise Stabilizer in a 2-Group

Having outlined the global structure of the algorithm for Problem 1, we now go into the details of its design. We begin with the two parts of Step one: determining
generators for $B$, the restriction of $\text{Aut}_e(X_{k+1})$ to $V_k$, and extending these generators to elements in $\text{Aut}_e(X_{k+1})$.

Throughout, we assume that $X = (V,E)$ is a connected trivalent graph, $e = (v_1,v_2)$ an edge of $X$. As before, $V_k$ is the set of vertices at distance $k$ from $e$ and $h$ is the height of $X$. The edge sets $E_1, ..., E_h$ and the subgraphs $X_0, ..., X_{h+1}$ are defined as in Subsection 1.2.

Let $A = \text{Aut}_e(X_k)$. Since $A$ stabilizes $V_k$ setwise, we can obtain $\bar{A}$, the restriction of $A$ to the set $V_k$, simply by restricting each generator of $A$ to $V_k$. By Theorem 1, both $A$ and $\bar{A}$ are 2-groups. Given the subset $Y$ of $X$, there is a natural embedding of $\text{Sym}(Y)$ as a subgroup into $\text{Sym}(X)$. With this embedding in mind, we define the subgroup $B$ of $A$ by

$$B = \{ \pi \in A \mid (\exists \psi \in \text{Sym}(V_{k+1}))(\pi \psi \in \text{Aut}_e(X_{k+1})) \}$$

and let $\bar{B}$ be the restriction of $B$ to $V_k$. Clearly $\bar{B} < \bar{A}$.

Our first task will be to show how to determine $\bar{B}$ from a setwise stabilizer in some 2-group. Then we will address the problem of how to extend $\bar{B}$ to a subgroup of $\text{Aut}_e(X_{k+1})$.

Recall Theorem 7 of Chapter II. The theorem showed how to obtain the automorphism group of an arbitrary graph from the intersection of a specific permutation group with a direct product of symmetric groups, i.e., from the setwise stabilizer in a particular permutation group. The theorem was obtained by considering the induced action of a vertex permutation on the set of all vertex pairs, thereby translating edges into a suitable labelling of points in the new permutation domain. We will now use a similar trick to obtain $\bar{B}$ as the setwise stabilizer of an isomorphic representation of $\bar{A}$. A minor difficulty arises from the fact that there are edges in $E_k$ connecting vertices in $V_k$ with vertices in $V_{k+1}$. These edges do not translate into pairs of vertices of $V_k$, but certain collections of these edges can be represented as small subsets of $V_k$ as we now explain:

Let $u \in V_{k+1}$. Define the ancestry of $u$ as the set of vertices $w \in V_k$ such that $(u,w)$ is an edge of $X$ (and thus in $E_k$). Since $X$ is connected, every vertex in $V_{k+1}$ has a nonempty ancestry. Furthermore, since $X$ is trivalent, every ancestry has cardinality at most 3.

We will classify the edges in $E_k$ into types and will group them into families. Let $(w,v) \in E_k$. If $(w,v)$ is a cross edge, i.e., if both $w$ and $v$ are in $V_k$, then the edge has the
type $t_{0,2}$. Otherwise, let $w \in V_{k+1}$, $v \in V_k$. Then the type of the edge $(w,v)$ is $t_{i,j}$, where $i$ is the cardinality of the ancestry of $w$, and $i-1$ is the number of vertices in $V_{k+1}$ with the same ancestry as $w$. In the trivalent case, note that $j \leq 3$ and $i \leq 2$. Next, if $(w,v)$ has type $t_{0,2}$, then its family is $\{(w,v)\}$ and thus consists of a single cross edge only. Otherwise, if $(w,v)$ has type $t_{i,j}$, $i > 0$, then the family of this edge is the set $\{(w,v), (w,v_2), \ldots, (w,v_j), (w_2,v), \ldots, (w_i,w_j)\}$, consisting of the $i$ vertices in $E_k$ connecting the vertices $w, w_2, \ldots, w_i$ in $V_{k+1}$ with their common ancestry, the vertices $v, v_2, \ldots, v_j$ in $V_k$. Figure 4 below shows the families and types which occur in the trivalent case. Note that a family $F$ of edges of type $t_{i,j}$, $i > 0$, spans the bipartite graph $K_{i,j}$.

![Types and Families of Trivalent Graphs](image)

If $F$ is a family of edges of type $t_{i,j}$, then $(F)_k$ denotes the set of $j$ vertices of $V_k$ incident to the family, and $(F)_{k+1}$ denotes the set of $i$ vertices in $V_{k+1}$ incident to the family.

**Lemma 1**

Let $\alpha \in \text{Aut}_q(V_{k+1})$. For every family $F$ in $E_k$ of type $t_{i,j}$, either $\alpha$ stabilizes $F$ or it maps $F$ onto a family $F'$ of the same type.
Proof Let \( F \) be any family of type \( t_{ij} \). If \( \alpha \) stabilizes \( (F)_k \), then \( \alpha \) must also stabilize \( (F)_{k+1} \). Therefore \( \alpha \) stabilizes the family \( F \) of edges. If \( \alpha \) maps \( (F)_k \) into a set \( (F')_k \), then, by the definition of family, \( \alpha \) must map \( (F)_{k+1} \) into \( (F')_{k+1} \), and so the family \( F \) is mapped into a family \( F' \) of edges of type \( t_{r,s} \), \( r \geq i, s \geq j \). By the pigeonhole principle, \( r = i \) and \( s = j \), from which the lemma follows.

**Lemma 2**

If for every family \( F \) in \( E_k \) the automorphism \( \alpha \in \text{Aut}_e(X_k) \) either stabilizes \( (F)_k \) or maps this set into the set \( (F')_k \) of a family \( F' \) of type equal to \( F \), then \( \alpha \) can be extended to an automorphism in \( \text{Aut}_e(X_{k+1}) \).

Proof If \( \alpha \) satisfies the hypothesis of the lemma, then it clearly maps cross edges to cross edges. So, let \( F \) and \( F' \) be families of edges which are not cross edges. Then, by the definition of family, \( (F)_{k+1} \cap (F')_{k+1} = \emptyset \), from which the lemma follows.

**Corollary 2 (Furst, Hopcroft, Luks)**

Let \( \alpha \in \text{Aut}_e(X_{k+1}) \) be an automorphism which fixes every vertex in \( V_k \). Then \( \alpha \) stabilizes every set \( (F)_{k+1} \), \( F \) a family in \( E_k \).

Let \( W \) be the collection of all subsets of \( V_k \) of size 1, 2, or 3. Note that \( G \), defined by the induced action of \( \bar{A} \) on \( W \), is again a 2-group. We label the points in \( W \) with the labels \( t_{ij} \), \( 0 \leq i \leq 2, 1 \leq j \leq 3 \), where the point \( z \) is labelled \( t_{ij} \) if \( z = (F)_k \) for some family \( F \) of edges in \( E_k \) of type \( t_{ij} \). The remaining, unlabelled points of \( W \) are now labelled \( t_{0,0} \). Observe that any point in \( W \) can have at most two labels; that is, the only multiple labelling possible is \( t_{1,2} \) and \( t_{0,2} \). Such points may be labelled \( t_{0,1,2} \). Note that we have used up to nine labels. Let \( H \) be the subgroup of \( G \) consisting of all elements in \( G \) which respect the labelling of \( W \), and note that \( H \) is the setwise stabilizer of the subsets of points of \( W \) with the same label. Let \( \bar{B} \) be the subgroup of \( \bar{A} \) corresponding to the subgroup \( H \) of \( G \). By Lemmata 1 and 2, it is now clear that \( \bar{B} \) is the subgroup of \( \bar{A} \) consisting of all permutations of \( V_k \) which may be extended to an automorphism in \( \text{Aut}_e(X_{k+1}) \).

**Example 3**

Consider the graph \( X \) of Example 1. The edge set \( E_2 \) contains the following four families: \( \{(3,4)\} \), a family of type \( t_{0,2} \), \( \{(3,7)\} \), a family of type \( t_{1,1} \), \( \{(4,8)\} \), also of type \( t_{1,1} \), and \( \{(5,9), (5,10), (6,9), (6,10)\} \), a family of type \( t_{2,2} \). The set \( W \) consists of the following 14 points: \( x_1 = \{3\} \), \( x_2 = \{4\} \), \( x_3 = \{5\} \), \( x_4 = \{6\} \), \( x_5 = \{3,4\} \), \( x_6 = \{3,5\} \), \( x_7 = \{3,6\} \), \( x_8 = \{4,5\} \), \( x_9 = \{4,6\} \), \( x_{10} = \{5,6\} \), \( x_{11} = \{3,4,5\} \), \( x_{12} = \{3,4,6\} \), \( x_{13} = \{3,5,6\} \), \( x_{14} = \{4,5,6\} \).
The group $G$ defined by the induced action of $\overline{A} = \langle(3,4), (3,5)(4,6)\rangle$ on $W$ is generated by the two permutations $(x_1,x_3)(x_2,x_4)(x_5,x_{10})(x_7,x_9)(x_{11},x_{13})(x_{12},x_{14})$ and $(x_1,x_2)(x_6,x_3)(x_7,x_9)(x_{13},x_{14})$.

The edge set $E_2$ induces the following four blocks on $W$ via the labelling described above: $J_1 = \{x_3\}$, labelled $t_{0,2}$, $J_2 = \{x_1,x_2\}$, labelled $t_{1,1}$, $J_3 = \{x_{10}\}$, labelled $t_{2,2}$. The remaining points in $W$ form in the block $J_4$ and are labelled $t_{0,0}$. $\Box$

Because of the presence of singletons in $W$, it is easy to obtain generators for $B$ from generators of the subgroup $H$ of $G$, the setwise stabilizer of the induced partition blocks of $W$. We will now show how to extend these generators to elements of $\text{Aut}_e(X_{k+1})$.

Let $\pi \in B$ be a permutation of $V_k$. Since generators for $\text{Aut}_e(X_k)$ are already known, we may assume that a representation matrix for $\text{Aut}_e(X_k)$ is also known where we assume that the points in $V_k$ are stabilized first. Thus, in order to extend $\pi$ to an automorphism $\varphi$ in $\text{Aut}_e(X_k)$, we proceed as follows: We test membership of $\pi$ in $\text{Aut}_e(X_k)$. In general, we will discover that $\pi$ is not in this group, but in the course of the membership test we will have computed $\pi' = \pi\psi_1^{-1} \cdots \psi_r^{-1}$ where $\pi'$ fixes $V_k$ pointwise (cf. Chapter II, Section 3). So $\varphi = \psi_r \cdots \psi_1$ is an element of $\text{Aut}_e(X_k)$ and must agree with $\pi$ on the set $V_k$, since $\pi \in A$. Next, note that it is easy to extend $\varphi$ to a permutation $\chi$ of $V_0 \cup \cdots \cup V_{k+1}$ which preserves the edges in $E_k$, since the families $F$ in $E_k$ induce a partition of $V_{k+1}$ with the sets $(F)_{k+1}$ as its blocks. Therefore, it is clear how to find the desired extension of $\pi$ to an automorphism in $\text{Aut}_e(X_{k+1})$.

**Lemma 3**

Let $K_1$ be a generating set for $B$, $K_1$ its extension in $\text{Aut}_e(X_k)$. Let $K_2$ be a generating set for $C$, the pointwise stabilizer of $V_k$ in $\text{Aut}_e(X_k)$. Then $K_1 \cup K_2$ generates $B$, the subgroup of all automorphisms in $\text{Aut}_e(X_k)$ which may be extended to automorphisms in $\text{Aut}_e(X_{k+1})$.

**Proof** Clear from the definition of $B$. $\Box$

As a consequence of Lemma 3 and the preceding discussion, we now have a way of finding generators for the group $B$ and, furthermore, a way of extending these generators to elements in $\text{Aut}_e(X_{k+1})$. Thus, we have an algorithm for Step one of the overall construction outlined previously.
We now turn to Step two, the determination of those automorphisms in $\text{Aut}_e(X_{k+1})$ which pointwise stabilize the vertices of $V_0 \cup \cdots \cup V_k$.

By Corollary 2, every automorphism in $A^{(k)}(X_{k+1})$ must setwise stabilize the vertex sets $(F)_{k+1}$, $F$ the families of $E_k$. Clearly, if $F$ is any family, then the vertices in $(F)_{k+1}$ may be permuted arbitrarily, thus $A^{(k)}(X_{k+1})$ must be the direct product of the symmetric groups of these sets. Since we assume that $X$ is a trivalent graph, no set $(F)_{k+1}$ has cardinality exceeding 2, thus $A^{(k)}(X_{k+1})$ is generated by disjoint transpositions and is therefore an elementary Abelian 2-group.

**Example 4**

Let $X$ be the graph of Example 1, and consider the vertex set $V_2 = \{7, 8, 9, 10\}$. The families of $E_2$ partition $V_2$ into the blocks $B_1 = \{7\}$, $B_2 = \{8\}$, and $B_3 = \{9, 10\}$. Thus, the group $A^{(2)}(X_3)$, consisting of all automorphisms in $\text{Aut}_e(X_3)$ which fix the vertices 1 through 6, is precisely $\langle (9, 10) \rangle$. □

### 1.4. Binary Cone Graphs

Having outlined the design of an algorithm for Problem 1, we now discuss briefly how these techniques apply to Problem 2, determining the automorphisms of binary cone graphs.

Let $X$ be a binary cone graph of height $h$ with root $v$, $V_k$ the set of vertices of distance $k$ from the root. Let $E_k$ be the subset of edges consisting of the BFS-tree edges connecting the vertices in $V_k$ with the vertices in $V_{k+1}$, and all cross edges incident to vertices in $V_k$. Define the graphs $X_k$, $1 \leq k \leq h$, by

$$X_k = (V_0 \cup \cdots \cup V_k, E_0 \cup \cdots \cup E_{k-1})$$

As in the trivalent case, we determine $\text{Aut}_v(X_k)$ for each graph $X_k$ by determining generators for $\text{Aut}_v(X_{k+1})$ from generators for $\text{Aut}_v(X_k)$ and the edge set $E_k$. Note that $\text{Aut}_v(X_1)$ has order either 1 or 2 and is determined easily from the graph $X_1$.

Just as before, we classify the edges in $E_k$ into types and group them into families. Since $X$ is a binary cone graph, the only possible types are $t_{0,2}$, $t_{1,1}$, and $t_{2,1}$, thus we can represent $\tilde{A}$ on the set $W$ consisting of all subsets of $V_k$ of size 1 or 2. Again, the families $F$ in $E_k$ induce a partition of $W$, in this case into at most four blocks. The setwise stabilizer of these blocks then gives us the subgroup $\bar{B}$, the restriction of
Aut\(_r(X_{k+1})\) to the vertex set \(V_k\).

From these observations, it should be clear that Problem 2 can be solved by the same techniques as the ones developed for Problem 1. Because \(W\) is of size \(O(n^2)\) for binary cone graphs, instead of \(O(n^3)\) in the trivalent case, the algorithm for Problem 2 has a faster running time. However, this difference in performance can be eliminated by replacing certain edges in \(E_{k+1}\) with specially labelled cross edges. We will discuss this technique in Section 4 of this chapter. In the remainder of the chapter we merely state the results for binary cone graphs without explicit proofs. The reader should have no difficulty in working out the necessary details.

2. An Algorithm for Determining the Automorphisms of Trivalent Graphs

In Section 1 we have developed the basic design of a polynomial time algorithm for Problem 1 (and thus also for Problem 2). Algorithm 1 below formally specifies the method. The major work is done in the main loop which successively determines generating sets for the groups \(\text{Aut}_e(X_k)\) given the connected, trivalent graph \(X\).

By far the largest part of the loop is devoted to Step one, constructing the subgroup \(B\) of \(A = \text{Aut}_e(X_k)\) consisting of those automorphisms in \(A\) which may be extended to an automorphism of \(X_{k+1}\). This happens in Lines 5-21, and includes finding generators for \(\overline{B}\), extending those generators, and finding generators for the pointwise stabilizer of \(V_k\) in \(A\). Step two, determining the group \(A^{(k)}(X_{k+1})\), the pointwise stabilizer of \(V_0 \cup \cdots \cup V_k\) in \(\text{Aut}_e(X_{k+1})\), is done in Lines 22 and 23 and is, as already remarked, very straightforward.

The algorithm has certain inefficiencies. We discuss some of the sources of inefficiency at the end of this section, and improve the algorithm with the material of the next section. Finally, in Section 4, we design an almost practical algorithm for trivalent graph isomorphism.
Algorithm 1 (Automorphism of a Connected, Trivalent Graph)

Input The connected, trivalent graph \( X = (V,E) \) and the edge \( e = (v_1,v_2) \) of \( X \).

Output A generating set for \( \text{Aut}_e(X) \).

Method

1. begin
2. Using breadth-first search, determine the vertex sets \( V_0, \ldots, V_h \) and the edge sets \( E_0, \ldots, E_h \);
3. Determine \( K_0 \), a generating set for \( \text{Aut}_e(X_0) \), by inspection;
4. for \( k := 0 \) to \( h \) do begin
5. From the set \( K_k \), construct the set \( \overline{K} \), generating the group \( \overline{A} \) that is the restriction of \( \text{Aut}_e(X_k) \) to \( V_k \);
6. Construct \( W \), the set of all subsets of \( V_k \) of size 1, 2, or 3;
7. From \( \overline{K} \), construct \( D \), a generating set for the group \( G \) defined by the induced action of \( \overline{A} \) on \( W \);
8. Classify the edges in \( E_k \) by type and group them into the families \( F_1, \ldots, F_r \);
9. Partition \( W \) into the blocks \( J_1, \ldots, J_9 \) according to the types of the families \( F \) and the sets \( (F)_k \);
10. Find a generating set for \( H \), the setwise stabilizer of the blocks \( J_1, \ldots, J_9 \) in the 2-group \( G \);
11. From the generating set for \( H \), construct \( \overline{D} \), a generating set for \( \overline{B} \), the restriction of \( \text{Aut}_e(X_{k+1}) \) to \( V_k \);
12. Construct a representation matrix for \( A = \text{Aut}_e(X_k) \), where the vertices in \( V_k \) are stabilized first;
13. Initialize \( K_{k+1} \), the generating set for \( \text{Aut}_e(X_{k+1}) \), to contain the generators of the pointwise stabilizer of \( V_k \) in \( \text{Aut}_e(X_k) \);
14. for each generator \( \pi \) in \( \overline{D} \) do begin
15. find \( \varphi \in A \) whose restriction to \( V_k \) is \( \pi \);
16. if \( k < h \) then begin
17. find \( \chi \), the extension of \( \varphi \) to an automorphism of \( X_{k+1} \);
18. add \( \chi \) to \( K_{k+1} \);
19. end
20. else add \( \varphi \) to \( K_{k+1} \);
21. end;
127

22. **for each set** \((F)_{k+1}\) **do**
23. \[\text{if } (F)_{k+1} = \{u, w\} \text{ then add } \pi = (u, w) \text{ to } K_{k+1};\]
24. **end**;
25. **output**\((K_{h+1})\);
26. **end**

We analyze Algorithm 1 first without accounting for the exact bound on Step 10 which is in charge of finding the setwise stabilizer in a 2-group. We assume that Step 10 requires \(T(m)\) steps, where \(m\) is the degree of \(G\), assuming further that \(G\) is presented by a generating set of size \(O(m^2)\) and that the generating set determined for \(H\) also is at most of size \(O(m^2)\). Clearly these assumptions are realistic.

Assume that \(X\) has \(n\) vertices, and observe that \(X\) cannot have more than \(3n^2\) edges. Let \(n_k\) be the cardinality of \(V_k\), and note that \(E_k\) is \(O(n_k + n_{k+1})\) in size.

We begin by estimating the time required in each iteration of the **for**-loop extending through Lines 4-24. Assuming that \(K_k\) has cardinality \(O(n^2)\), Line 5 requires \(O(n^3)\) steps. The resulting generating set is for a group of degree \(n_k\). Therefore, by Chapter II, we may reduce the set \(\overline{K}\) to size \(O(n_k^3)\) at the cost of \(O(n^3 + n_k^2 + n_k^6)\) steps. Line 6 requires \(O(n_k^2)\) steps, and results in a set \(W\) of size \(O(n_k^2)\). Thus, we can construct \(D\) from \(\overline{K}\) in \(O(n_k^2)\) steps (Line 7), processing \(O(n_k^2)\) generators. Classification of the \(O(n_k + n_{k+1})\) edges in \(E_k\) requires at most \(O(n_k + n_{k+1})\) steps since there are only 7 distinct types and the largest possible family consists of 6 edges, i.e., since the families have constant size. Subsequently, the set \(W\) can be partitioned in \(O(n_k^2 + n_k^6)\) steps, so that Lines 5-9 require a total of \(O(n^3 + n^2 + n_k^2 + n_k^6 + n_k^6 + n_{k+1})\) steps.

By assumption, Line 10 requires \(T(n_k^2)\) steps and delivers a generating set of size at most \(n_k^6\), since \(H\) has degree \(O(n_k^2)\).

Line 11 constructs a generating set for \(\overline{M}\). Because of the possibly very large generating set for \(H\), this line costs \(O(n_k^2)\) steps. The resulting generating set can be reduced to size \(O(n_k^2)\), in the same time bound. Line 12 now requires \(O(n_k^2 + n^6)\) steps and allows us to initialize \(K_{k+1}\) in \(O(n^3)\) steps in Line 13.

The loop in Lines 14-21 is executed at most \(O(n_k^2)\) times. Step 15 is done by partially sifting \(\pi\), thus requires \(O(n_k)\) steps. Step 16 can be done in \(O(n_k + n_{k+1})\) steps, thus Lines 14-21 take at most \(O(n^2 + n_k^2 + n_k^6 + n_k^6 + n_{k+1})\) steps.
The loop in Lines 22 and 23 takes no more that \( O((n_k+n_{k+1}) n) \) steps, assuming the permutations are stored as vectors of length \( n \). Therefore, Lines 5-23 require a total of at most \( O(n^6 + n^2k + T(n^3)) \) steps, observing that \( n \geq n_k \). Note that the set \( K_{k+1} \) is of size at most \( O(n^2) \), and that \( h \) is at most \( O(n) \). Thus, the loop of Lines 4-24 requires a total of \( O(n^6 + \sum_{k=1}^{h} T(n^2_k)) \) steps.

Under the assumption that \( T(m) \) increases monotonically with \( m \) and at least as fast as linearly in \( m \), we obtain \( \sum_{k=1}^{h} T(n^2_k) \leq T(n^3) \). Clearly, the running time of the loop dominates all other steps. Therefore, in summary, we have just shown

**Theorem 2**

Let \( X \) be a connected, trivalent graph with \( n \) vertices. Assume we have a procedure which determines, in \( T(m) \) steps, an \( O(m^2) \) generating set for the setwise stabilizer in a 2-group \( G \) of degree \( m \), presented by a generating set also of size \( O(m^2) \). Then a generating set for \( \text{Aut}_e(X) \) can be found in \( O(n^6 + T(n^3)) \) steps.

**Corollary 3**

Let \( X \) be a binary cone graph with \( n \) vertices and root \( v \). Assume we have a procedure which determines, in \( T(m) \) steps, an \( O(m^2) \) generating set for the setwise stabilizer in a 2-group \( G \) of degree \( m \), presented by a generating set also of size \( O(m^2) \). Then a generating set for \( \text{Aut}_e(X) \) can be found in \( O(n^6 + T(n^3)) \) steps.

We now apply the results of Chapter III, Section 3. By Theorem 18 of Chapter III, we obtain immediately

**Corollary 4** (Furst, Hoffmann, Hopcroft, Luks)

Let \( X \) be a connected, trivalent graph with \( n \) vertices. Then generators for \( \text{Aut}_e(X) \) can be found in \( O(n^{21}) \) steps.

**Corollary 5**

Let \( X \) be a binary cone graph with \( n \) vertices and root \( v \). Then generators for \( \text{Aut}_r(X) \) can be found in \( O(n^{14}) \) steps.

We have just solved Problems 1 and 2 in polynomial time. As a consequence, we now have a polynomial time isomorphism test for binary cone graphs and for trivalent graphs.

The bounds in Corollaries 4 and 5 can be significantly improved. The improvements come from several sources: For one, we should seek to reduce the degree of
the 2-group $G$ in which we determine the setwise stabilizer $H$. In the trivalent case, it is not hard to reduce this degree to $O(n^2)$ by replacing families $F$ of type $t_{1,3}$ by three specially labelled cross edges connecting the vertices in $(F)_k$. Note that the newly added edges for families of type $t_{1,3}$ must have labels different from the labels for the edges added for families of type $t_{2,3}$. The other source for improving the running time is to seek better techniques for Step 10, and we will explore this idea in the next section. Finally, a certain amount of duplication of work can be eliminated, resulting in a further savings in the running time.

3. Setwise Stabilizers in $p$-Groups (Method 2)

As we have seen above, the efficiency of Algorithm 1 depends crucially on the timing of Step 10, the determination of a setwise stabilizer in a 2-group. In Chapter III, we gave a polynomial time algorithm for finding setwise stabilizers in $p$-groups (Chapter III, Theorem 18). We will now develop a different method for this problem based on new techniques.

The techniques to be introduced are only indirectly related to the techniques of the preceding chapters. We therefore begin with an intuitive, perhaps somewhat vague, development of the underlying ideas.

Let $G < \text{Sym}(X)$ be an intransitive permutation group with orbits $\Delta_1, \ldots, \Delta_s$. Let $Y$ be a subset of $X$. Assume we wish to determine $G_Y$, the setwise stabilizer of $Y$ in $G$. Let $Y_i = Y \cap \Delta_i$, $G^{(0)} = G$, and $G^{(i)} = (G^{(i-1)})_{Y_i}$, $1 \leq i \leq s$. Since $G$ stabilizes each of its orbits, clearly $G_Y = G^{(s)}$.

We have just dissected $Y$ into the sets $Y_i$ and have reduced the problem of finding $G_Y$ to the problem of finding the stabilizer of the (possibly smaller) sets $Y_i$. A priori, it is not obvious that the stabilizer of a smaller set $Y_i$ can be computationally determined more easily than the stabilizer of the larger set. However, if we succeed in breaking up the sets $Y_i$ in turn, and repeat this recursively until $Y$ is eventually decomposed into singleton sets, then there is hope for a polynomial time algorithm for finding $G_Y$ since, by Chapter II, we know how to determine pointwise stabilizers efficiently.

We consider how to break up the sets $Y_i$. The major difficulty is that $G$ acts transitively on the orbit $\Delta_i$. However, if there is a subgroup $H$ of $G$ acting intransitively on
$\Delta$, then, for this subgroup $H$, we may dissect $Y_1$ by intersecting it with the orbits of $H$.
In fact, since $I < G$, such a subgroup $H$ always exists.

Not every intransitive subgroup $H$ is of use: Let $G$ be a transitive group, $H$ any
intransitive subgroup of $G$, and let $H_Y$ be the setwise stabilizer of $Y$ in $H$. To obtain $G_Y$,
it is not sufficient to know the subgroup $H_Y$. Instead, we also need to determine the
setwise stabilizer of $Y$ in each right coset $H\pi$ of $H$ in $G$, i.e., we need to find

$$(H\pi)_Y = \{ \psi \in H\pi \mid Y^\psi = Y \}$$

Note that $(H\pi)_Y$ could be empty. Since we have to find $(H\pi)_Y$ for each right coset $H\pi$
in $G$, only subgroups $H$ of small index in $G$ are useful.

While $H_Y$ is a subgroup and thus is easily represented by a small generating set,
the representation of the sets $(H\pi)_Y$ is not immediately clear. Here, we will use the
following

**Lemma 4**

Let $G, H < S_n, \pi$ any permutation. If $(G\pi)\cap H$ is not empty, then it is a right coset of
$G\cap H$.

**Proof** Let $\varphi, \psi \in (G\pi)\cap H$. Then $\varphi\psi^{-1} \in G$ and $\varphi\psi^{-1} \in H$,
and so $(G\pi)\cap H \subset (G\cap H)\psi$. Conversely, let $\psi \in (G\pi)\cap H, \varphi \in G\cap H$. Clearly $\varphi\psi \in H$. Furthermore,
since $G\cap H < G$, $\varphi\psi \in G\psi$. Thus $(G\cap H)\psi \subset (G\pi)\cap H$, from which the lemma fol-
lows.

Observe that $(H\pi)_Y = (H\pi) \cap (\text{Sym}(Y) \times \text{Sym}(\overline{Y}))$. Thus, we may represent nonempty
sets $(H\pi)_Y$ succinctly by a small generating set for $H_Y$ and a coset representative
$\psi \in (H\pi)_Y$.

As a further consequence of Lemma 4, the step for breaking up $Y$ in the intransi-
tive case has a generalization to determining the stabilizer of $Y$ in cosets of intransi-
tive groups. The details of this will be outlined later.

Summarizing the ideas just sketched: We aim at finding a recursive procedure for
determining the setwise stabilizer in groups and in right cosets of groups. The pro-
dure is to split the set $Y$ to be stabilized into smaller sets by considering a sub-
group tower consisting of intransitive groups with small index in the containing sub-
group. We do not expect all permutation groups to possess a suitable subgroup tower.
However, in the case of $p$-groups such a tower always exists, and below we will develop
a suitable algorithm realizing these concepts.
3.1. The Algorithm

We now develop the details of the algorithm for finding the setwise stabilizer in p-groups. We pointed out earlier that the essential prerequisite for the algorithm is the existence of a subgroup tower consisting of intransitive subgroups of small index. We show that p-groups always have such towers because of their imprimitivity structure (cf. Chapter III, Subsection 3.3).

Recall the construction of a Sylow p-subgroup $P$ of $S_n$ containing a given p-group as subgroup, and consider the associated cone graph(s). If $X$ is the (directed, regular) cone graph of height $h$ associated with $P$, then the leaves of the $p$ subtrees of height $h-1$ must form a system of imprimitivity for $P$, and the subgroup of $P$ which (setwise) stabilizes each set of imprimitivity has index $p$ in $P$. Developing this argument slightly, we obtain immediately

**Lemma 5**

Let $G$ be a p-group with the nontrivial orbit $\Delta$. Then $\Delta$ may be partitioned into $p$ sets $\Gamma_1, \ldots, \Gamma_p$ of equal size which form a system of imprimitivity for (the action of) $G$ on $\Delta$, and the subgroup $H$ of $G$ stabilizing each set $\Gamma_i$ setwise has index $p$ in $G$.

Lemma 5 provides the basic tool for finding suitable intransitive subgroups. Note that the required system of imprimitivity may be found using the techniques of Subsection 3.3 in Chapter III.

Let $G_0$ be an intransitive p-group with orbits $\Delta_1, \ldots, \Delta_s$. Recall that we plan to stabilize the set $Y$ in $G_0$ by successively stabilizing the sets $Y_i = Y \cap \Delta_i$. Clearly, the stabilization of $Y_i$ depends only on the action of $G_0$ on the set $\Delta_i$. One might now conclude that we should stabilize $Y_i$ in the transitive constituent of $G_0$ in $\text{Sym}(\Delta_i)$. But this is not right, for the stabilization of $Y_i$ in $G_0$ may affect the way in which the group can act on the remaining orbits. For this reason we only deal with subgroups $G$ that have degree equal to the degree of $G_0$. Along with the subgroup $G$ of $G_0$ (or a right coset of $G$) we will keep track of a set $Z$. Here $G$ is known to setwise stabilize $Z$ and we are considering the action of $G$ on $Z$ for the purpose of stabilizing $Y \cap Z$. We alternate recursively between the following two steps:

(a) $G$ acts intransitively on $Z$ with orbits $\Delta_1, \ldots, \Delta_s$: Successively consider the sets $Y_i = Y \cap \Delta_i$, $1 \leq i \leq s$.

For the group $G$, stabilize the sets $Y_i$ by calling Step (b) recursively.
For the coset \( G\pi \) of \( G \), find all those permutations in \( G\pi \) which map \( Y_1 \) into \( Y \) by
calling Step (b) recursively. Next, in the resulting set, find all those permuta-
tions mapping \( Y_2 \) into \( Y \), et cetera.

(b) \( G \) is transitive on \( Z \) (but stabilizes \( Z \) setwise): Find a subgroup \( H \) of index \( p \) in \( G \)
that stabilizes a maximal system of imprimitivity for the action of \( G \) on \( Z \). Let
\( G = H\varphi_1 + \cdots + H\varphi_p \).

For the group \( G \), determine the stabilizer of \( Y \cap Z \) in each \( H\varphi_i \) by recursively cal-
ling Step (a).

For the coset \( G\pi \) of \( G \), stabilize \( Z \cap Y \) (in \( Y \)) in the cosets \( H\varphi_i \pi \) for each \( i \), again
using Step (a).

In either case, "paste together" the nonempty cosets (and subgroups), obtaining
\( G_{Y \cap Z} \) (or \( (G\pi)_{Y \cap Z} \)). This part is discussed in more detail below.

For each recursive application of Step (a), note that the size of \( Z \) is smaller by a fac-
tor of \( \frac{1}{p} \). The recursion ends when \( Z = \{z\} \) is a singleton and \( G \) fixes \( z \). In this case,
\( G_{Y \cap Z} \) is clearly \( G \). Moreover, by Theorem 3 of Chapter II, \( (G\pi)_{Y \cap Z} \) is nonempty iff
\( z'' \in Y \) whenever \( z \in Y \). In that case, we have \( (G\pi)_{Y \cap Z} = G\pi \).

We now specify more succinctly the recursive procedure just sketched. Let
\( G < \text{Sym}(X) \) be a group, \( Y \) and \( Z \) subsets of \( X \). For \( \pi \in \text{Sym}(X) \) we define
\[
S_Y(G\pi, Z) = \{ \psi \in G\pi \mid (\forall z \in Z)(z^\psi \in Y \iff z \in Y) \}.
\]

If \( Z \) is setwise stable in \( G \), then, by Lemma 4 above, \( S_Y(G\pi, Z) \) is either empty, or it is a
right coset of \( S_Y(G, Z) = G_{Y \cap Z} \). Rules (a) and (b) above now translate as follows:

(a) If \( Z \) is stable in \( G \) and \( G \) is intransitive on \( Z \) with the orbits \( \Delta_1, \ldots, \Delta_w \), then
\[
S_Y(G\pi, Z) = S_Y(S_Y(... S_Y(G\pi, \Delta_1), \ldots, \Delta_{w-1}), \Delta_w).
\]

(b) If \( G = H\varphi_1 + H\varphi_2 + \cdots + H\varphi_p \), then
\[
S_Y(G\pi, Z) = S_Y(H\varphi_1\pi, Z) + S_Y(H\varphi_2\pi, Z) + \cdots + S_Y(H\varphi_p\pi, Z).
\]

Finally, for the base case of the recursion,

(c) If \( G \) fixes the point \( z \), then \( S_Y(G\pi, \{z\}) = G\pi \) provided that \( z'' \in Y \) whenever \( z \in Y \);
otherwise, \( S_Y(G\pi, \{z\}) = \phi \).
We now turn to the problem of pasting cosets together. The basic result used is Lemma 4. Specifically, if
\[ G\pi = H\varphi_1\pi + H\varphi_2\pi + \cdots + H\varphi_p\pi \]
then, by Rule (b) above,
\[ S_Y(G\pi, Z) = S_Y(H\varphi_1\pi, Z) + S_Y(H\varphi_2\pi, Z) + \cdots + S_Y(H\varphi_p\pi, Z), \]
where "+" denotes the disjoint union. By Lemma 4, a nonempty set \( S_Y(H\varphi_i\pi, Z) \) is a coset \( <K_i>\psi_i \) and the nonempty set \( S_Y(H\varphi_j\pi, Z) \) is the coset \( <K_j>\psi_j \), where
\[ <K_i> = <K_j> = S_Y(H, Z). \]
Thus,
\[ S_Y(H\varphi_1\pi, Z) + S_Y(H\varphi_2\pi, Z) + \cdots + S_Y(H\varphi_p\pi, Z) = <S_Y(H, Z), K>\psi, \]
where \( \psi \) is one of the \( \psi_j \) and \( K = \{ \psi_i\psi_j^{-1} \mid S_Y(H\varphi_i\pi, Z) \neq \emptyset \} \). This gives us a method for collecting the nonempty cosets in Step (b) and representing their union succinctly.

It is important to note that the above rules are valid for all permutation groups although their usefulness is limited since a permutation group \( G \) need not have a suitable subgroup tower of intransitive subgroups. In general, the problem is Rule (b) which creates as many recursive calls as the index of \( H \) in \( G \). With p-groups it is always possible to find a subgroup \( H \) of index \( p \) in \( G \), but typically there may not be an intransitive subgroup \( H \) of acceptably small index.

Algorithm 2 below gives the details of the algorithm for p-groups. Rule (a) is handled in Lines 12-25 and Rule (b) in Lines 26-41. The base case, Rule (c), is dealt with in Lines 7-11. The heart of the algorithm is the recursive procedure STABILIZE. It accepts as input parameters a description of a (right coset of a) p-group, \( <K>\pi \), and a subset \( Z \) of the permutation domain. \( <K> \) is known to stabilize \( Z \). The procedure computes \( S_Y(<K>\pi, Z) \) and so either returns the coset \( <K'>\psi, \) where \( <K'> = <K>\cap_Z \psi \), or it returns an indication that \( S_Y(<K>\pi, Z) \) is empty in the boolean variable isempty.

Correctness of the algorithm is straightforward from the preceding discussion and from Lemmata 4 and 5. The time requirements will be discussed subsequently.
ALGORITHM 2 (Setwise Stabilizer in a p-Group)

Input Generating set $K$ for a p-group $G$ acting on the set $X$ of size $n$, and a subset $Y$ of $X$.

Output Generating set $K'$ of $G_Y$, the setwise stabilizer of $Y$ in $G$. In case that $G_Y$ is the trivial group, the set $K'$ equals $\{()\}$.

Comment The recursive procedure STABILIZE, Lines 5-44, works through the tower of intransitive subgroups.

Method
1. begin
2. STABILIZE( $\emptyset$, $K$, $X$; $\psi$, $K'$, isempty);
3. output($K'$);
4. end.

5. procedure STABILIZE ( $\pi$, $K$, $Z$; $\psi$, $K'$, isempty);
   comment The procedure searches the coset $<K>\pi$ of the p-group $<K>$, where $<K>$ is known to stabilize the set $Z$. It determines the set of all permutations in the coset which map $Z \cap Y$ into $Y$ and map $Z-Y$ into $X-Y$. This set is either empty (and then the variable $\text{isempty}$ is true), or it is the coset $<K'>\psi$, where $<K'>$ is the setwise stabilizer of $Z \cap Y$ in $<K>$, (and then $\text{isempty}$ is false).
6. begin
7. if $G$ fixes $Z$ pointwise then begin
8. $K' := K$;
9. $\psi := \pi$;
10. isempty := $((Z \cap Y)^c \subset Y)$ and $((Z-Y)^c \subset X-Y)$;
11. end
12. else begin
13. determine the orbits $\Delta_1$, ..., $\Delta_s$ of $<K>$ in $Z$;
14. if $s > 1$ then begin
   comment $<K>$ acts intransitively on $Z$;
15. $K_0 := K$;
16. $\pi_0 := \pi$;
17. $i := 1$;
isempty := false;

while i < s and not isempty do begin
  STABILIZE(\pi_{i-1}, K_{i-1}, \Delta_i; \pi_i, K_i, isempty)
  i := i+1;
end;

\psi := \pi_n;
K' := K_n;
end

else begin

comment $<K>$ acts transitively on $Z$, so there must be a system of imprimitivity for $<K>$ partitioning $Z$ into exactly $p$ subsets of equal size;

find $p$ sets of imprimitivity, $\Gamma_1, \ldots, \Gamma_p$, for the action of $<K>$ on $Z$;

find a generating set $\bar{K}$ for the subgroup $H$ of $<K>$ which stabilizes each of the $\Gamma_i$ setwise, and find a complete right transversal $\{\varphi_1, \ldots, \varphi_p\}$ for $H$ in $<K>$;

isempty := true;

comment find the first nonempty coset;

i := 1;

while i <= p and isempty do begin
  STABILIZE(\varphi; \pi, \bar{K}, Z; \psi, K', isempty);
  i := i+1;
end;

comment find the remaining nonempty cosets;

while i <= p do begin
  STABILIZE(\varphi; \pi, \bar{K}, Z; \psi_i, K_i, isempty);
  if not isempty then
    K' := K' \cup \{ \psi_i \psi^{-1} \};
  i := i+1;
end;
end;

return(\psi, K', isempty);
end.
3.2. Analysis of Algorithm 2

We now analyze Algorithm 2's running time. The key issue here is the running time of the procedure STABILIZE. Assume that STABILIZE is called with the input parameters \( \pi, K, \) and \( Z, <K> \) a p-group. Clearly, the time required to determine \( S_Y(<K>\pi,Z) \) depends on:

- \( n \), the degree of the permutations,
- \( m \), the cardinality of \( Z \),
- \( p \), the prime dividing the order of \( <K> \), and
- \( k \), the largest cardinality of any generating set considered.

Without loss of generality, we will assume that \( k \) is \( O(n^2) \). Under the additional assumption that \( Z \) is an orbit of \( <K> \), i.e., that \( <K> \) acts transitively on \( Z \), we let \( T(m,n,p) \) be the running time of procedure STABILIZE and proceed to give an estimate for the asymptotic growth of this function.

Rather than estimating the time required by accounting for each line in the procedure, we will identify the work done by STABILIZE at each level of recursion and the number of recursive calls made. Since \( <K> \) is a p-group, the cardinality \( m \) of \( Z \) must be a power of \( p \) if \( <K> \) is to act transitively on this set. Thus, if \( m > 1 \), STABILIZE proceeds as follows:

1. \( Z \) is partitioned into \( p \) sets of imprimitivity, \( \Gamma_1, ..., \Gamma_p \), of equal size.
2. A generating set \( \bar{K} \) for the subgroup \( H \) of \( <K> \) is determined, where \( H \) stabilizes each set \( \Gamma_i \). Furthermore, a complete right transversal \( \{\varphi_1, ..., \varphi_p\} \) for \( H \) in \( <K> \) is found. Note that \( H \) is intransitive on \( Z \).
3. The sets \( S_Y(H\varphi_i\pi,Z) \) are determined. Because of the intransitivity of \( H \), this involves \( p \) consecutive recursive calls with the new sets \( Z = \Gamma_i \) to be considered.

   Note that \( H \) is transitive on \( \Gamma_i \), and that these sets have cardinality \( \frac{m}{p} \).
4. The nonempty sets found in Step 3 are combined into the coset \( S_Y(<K>\pi,Z) \).

Consequently, calling STABILIZE with \( \pi, K, \) and \( Z \) causes \( p^2 \) recursive calls on the procedure with sets \( Z' \) of size \( \frac{m}{p} \), from which we obtain the basic recurrence

\[
T(m,n,p) \leq p^2 T\left(\frac{m}{p},n,p\right) + f(m,n,p),
\]

where \( f(m,n,p) \) is the time required for the work on the first level of recursion, i.e.,
the work of Steps 1 through 4 above, excluding the cost of the recursive calls.

For estimating the overhead $f(m,n,p)$, we first observe that Steps 1 and 2 are the most time consuming ones. Now, by Corollary 5 of Chapter III, Step 1 requires at most $O(m^2 \cdot n^2 \cdot \log_2^*(m))$ steps, recalling our assumption that $K$ has size $O(n^2)$. Next, we note that Step 2 can be accomplished in $O(n^4 + p^2 \cdot n^4)$ steps if we modify the algorithms of Chapter II such that, instead of fixing points, (maximal) sets of imprimitivity are stabilized. This leads to a subgroup tower of width $p$ (cf. Theorem 12 of Chapter III; see also Proposition 3 in Subsection 4.1 below). Since $p \leq m$, we obtain the estimate

$$f(m,n,p) \leq c \cdot m^2 \cdot n^4$$

for a suitable constant $c$.

Before estimating the recurrence obtained thus far, we need to ensure that our assumption about the maximum size of the occurring generating sets is not invalidated by Step 4. By Lemma 5, we know that Step 4 can result in at most $p-1$ additional generators for $K'$. We observe that the base case (Lines 7-11 in Algorithm 2) delivers a generating set of size $O(n^2)$ at most, given that the original generating set is of that size. Each transitive level in the recursion recomputes an $O(n^2)$ size generating set for the subgroup to be considered, and the maximum depth of recursion is at most $\log_p(n)$. Thus, all intermediate generating sets are at most $O(n^2)$ in size. As a consequence, we obtain the recurrence

$$T(p \cdot m,n,p) \leq p^2 \cdot T(m,n,p) + c_1 \cdot m^2 \cdot n^4$$

For the base case, $m = 1$, we obtain

$$T(1,n,p) \leq c_1 \cdot n^2$$

where we may choose the constants for the base case and the estimation of $f(m,n,p)$ uniformly. An elementary induction now gives us

$$T(p^k,n,p) \leq c \cdot k \cdot p^{2k} \cdot n^4$$

for a suitable constant $c$. Consequently, we have just proved

**Lemma 6**

Let $<K>$ be a transitive $p$-group of degree $n$, $|K| = O(n^2)$. Then a generating set for the setwise stabilizer of $Y$ in $<K>$ can be found in $O(n^6 \cdot \log_p(n))$ steps.
We now consider the case where $G = \langle K \rangle$ is an intransitive $p$-group of degree $n$ with the orbits $\Delta_1, \ldots, \Delta_s$, assuming again that $|K|$ is $O(n^2)$. Let $m_i$ be the length of the orbit $\Delta_i$. Then the stabilization of $Y \cap \Delta_i$ requires no more than $T(m_i, n, p)$ steps, thus is $O(m_i^2 \cdot n^4 \cdot \log_2(m_i))$. Since $\sum_{i=1}^{s} m_i^2$ is not greater than $n^2$, we obtain

**Theorem 3**

Let $G = \langle K \rangle$ be a $p$-group of degree $n$. Then a generating set for the setwise stabilizer of $Y$ in $G$ can be determined in $O(|K| \cdot n^2 + n^8 \cdot \log_p(n))$ steps.

**Proof** Note that in $O(|K| \cdot n^2 + p^2 \cdot n^4)$ steps we can find an $O(n^2)$ size generating set for $G$. The theorem follows therefore from Lemma 6 and the above discussion.

We apply Theorem 3 to the results of Section 2, and obtain the following improvements over Corollaries 4 and 5:

**Corollary 6**

Let $X$ be a connected trivalent graph with $n$ vertices. Then generators for $\text{Aut}_e(X)$ may be found in $O(n \log(n))$ steps.

**Corollary 7**

Let $X$ be a binary cone graph with $n$ vertices and root $v$. Then generators for $\text{Aut}_v(X)$ may be found in $O(n \log(n))$ steps.

This implies, in particular, that isomorphism of trivalent graphs may be tested in $O(n^{18} \log(n))$ steps. Here we first split the graphs into connected components, followed by classifying the components into isomorphism classes using Corollary 6 and the results of Section 1. The details are straightforward.

4. An $O(n^4)$ Isomorphism Test for Trivalent Graphs

In this section, we will develop an $O(n^4)$ isomorphism test for trivalent graphs. More precisely, we give an $O(n^3)$ method for determining sufficient information about $\text{Aut}_e(X)$, $X$ a connected trivalent graph, to permit testing isomorphism in the manner described in Section 1 of this chapter. The design departs slightly from the overall approach of the preceding two methods in that the algorithm does not rely on setwise stabilization in 2-groups as the central step. Instead, other techniques are used, including the intersection of two 2-groups. The following are the key technical points of the design:
(1) Many of the previously given algorithms for computing with p-groups are improved. We give these algorithms in Subsection 4.1.

(2) A new problem for 2-groups, called the \textit{imprimitivity problem}, is defined and solved efficiently. The problem asks for the largest subgroup of a given 2-group which has a prescribed (minimal) system of imprimitivity. This problem can be understood as a paradigm for testing isomorphism of trivalent graphs with an especially simple graph-theoretic structure. The $O(n^3)$ solution given in Subsection 4.2 is the critical component of the isomorphism test.

(3) The notion of \textit{restricted isomorphism gadget} is developed and is exploited in order to simplify the graph-theoretic structure of trivalent graphs. Consequently, the $O(n^3)$ solution to the imprimitivity problem can be used for trivalent graphs in general. This technique is explained in Subsection 4.3.

\section{4.1. Improved Algorithms for p-Groups}

We develop a number of new or improved algorithms for computing in p-groups which are given as permutation groups. As we will show, the previously given algorithms for p-groups can be improved significantly. In certain cases, however, the improvement seems to depend on the availability of a special type of generating set. While the algorithms are developed for arbitrary primes $p$, we will later apply them in the special case of $p = 2$.

We begin with the problem of determining systems of imprimitivity for p-groups. Let $G = \langle \pi_1, \ldots, \pi_k \rangle$ be a transitive p-group of degree $n$. Let $x$ and $y$ be two distinct points in the permutation domain, and recall Algorithm 2 of Chapter III. The algorithm determines the smallest set of imprimitivity of $G$ containing both $x$ and $y$ and requires, by the proof of Theorem 15 of Chapter III, $O(k \cdot n \cdot \log_2^*(n))$ steps. If $G$ is an imprimitive p-group and if we could guarantee that the smallest set of imprimitivity containing both $x$ and $y$ is not the entire permutation domain $X$, then we could dispense with the trial-and-error step of Corollary 4 of Chapter III, thereby possibly eliminating a factor of $n$ in the time bound.

We now show that it is possible to find points $x$ and $y$ with this property. As a consequence, we can achieve an $O(k \cdot n \cdot \log_2^*(n))$ algorithm for finding a nontrivial sys-
tem of imprimitivity for the p-group G. Note that this is not an unqualified improvement over Corollary 4 of Chapter III, since we have no assurance that the sets of imprimitivity are of cardinality p. However, in only $O(k \cdot n \cdot \log_2^*(n) \cdot \log_2(n))$ steps we can determine a complete imprimitivity structure for the p-group G (and hence a Sylow p-subgroup of the symmetric group which contains G). This second result is a strict improvement over Corollary 5 of Chapter III.

In order to understand the approach to selecting suitable points x and y, we need to introduce the following concepts and facts:

**Definition 1**

Let $\pi$ and $\psi$ be elements of the group G. Then $[\pi, \psi] = \pi^{-1}\psi^{-1}\pi\psi$ is called the commutator of $\pi$ by $\psi$.

The term commutator derives from the identity $\pi\psi = \psi\pi[n, \psi]$.

Let $K_G$ be the subset of G containing all commutators of elements of G. Then $K_G$ generates a normal subgroup $G'$ of G which is called the commutator subgroup or the derived group of G. We will explore algebraic properties of commutators and derived groups in Subsection 5.2 of Chapter VI in further detail. For the present, we state without proof that $G'$ is the minimal normal subgroup of G whose factor group $G/G'$ is abelian. Hence every normal subgroup $N$ of G whose factor group $G/N$ is abelian contains the derived group $G'$. We will make use of this characterization of $G'$ below.

Let G be a transitive group, N a nontrivial, intransitive and normal subgroup of G. Then the orbits of N are a system of imprimitivity for G. To see this, let A be one of the orbits of N, $\pi$ an element of G. Since N is a normal subgroup of G, $N^\pi$, the conjugation of N by $\pi$, is N. Now $A^\pi$ is an orbit of $N^\pi = N$, hence $A^\pi$ must be one of the orbits of N, and therefore $\Delta$ is a nontrivial set of imprimitivity for G.

We now return to the problem of finding two points x and y which lie in a nontrivial set of imprimitivity of the transitive p-group G. Assume given a generating set $\{\pi_1, \ldots, \pi_k\}$ for the transitive p-group G of degree $n > p$. We compute the $k-1$ commutators $[\pi_1, \pi_2], \ldots, [\pi_1, \pi_k]$ in $O(k \cdot n)$ steps. There are two possibilities: We either find a nontrivial commutator $\psi = [\pi_1, \pi_i] \neq 0$, or $\pi_1$ commutes with every element of G, hence is in the center of G (cf. Definition 9 of Chapter III).

If $\pi_1$ is in the center, then $\psi = \pi_1$ or $\psi = \pi_1^p$ generates an intransitive normal subgroup of G; hence, the cycles of $\psi$ must be the orbits of this subgroup $<\psi>$ and are sets of imprimitivity for G. Therefore, in this case, we find a nontrivial system of imprimitivity for G in $O(k \cdot n)$ steps. (Note that $\pi_1^p$ can be computed in $O(n)$ steps.)
If \( \psi = [\pi_1, \pi_1] \neq () \) is a nontrivial commutator, then we argue that any two distinct points \( x \) and \( y \) in a cycle of \( \psi \) must lie in a nontrivial set of imprimitivity of \( G \). To see this, consider the intransitive normal subgroup \( H \) of \( G \) stabilizing the sets in a maximal system of imprimitivity of \( G \). Then the factor group \( G/H \) is abelian of order \( p \), hence \( H \) contains the commutator subgroup \( G' \) of \( G \). Consequently, the commutator subgroup \( G' \) is an intransitive normal subgroup of \( G \) whose orbits are nontrivial sets of imprimitivity for \( G \). Using Algorithm 2 of Chapter III we therefore have

**Proposition 1 (Hoffmann, Sims)**

Given a generating set \( \{\pi_1, \ldots, \pi_k\} \) for the \( p \)-group \( G \) of degree \( n \), where \( \pi_1 \neq () \). Then one can determine whether \( G \) is imprimitive, and if so, find a nontrivial system of imprimitivity for \( G \) in \( O(k \cdot n \cdot \log_2^*(n)) \) steps.

**Proof** If \( G \) is primitive, then every commutator \( [\pi_1, \pi_1] \) is trivial and \( \pi_1 \) is a \( p \)-cycle. The conclusion now follows from the above discussion.

We now show how to find efficiently a complete imprimitivity structure for the \( p \)-group \( G \). Without loss of generality, we may assume that \( G \) is imprimitive and transitive. The intransitive case is handled by separately processing the transitive constituents of \( G \).

Suppose we iterate the following two steps until the group \( G^* \) has become trivial or cyclic:

(a) Compute a system \( S = \{S_1, \ldots, S_r\} \) of nontrivial sets of imprimitivity for \( G \) using Proposition 1.

(b) Replace the group \( G \) with the homomorphic image \( G^* \) that has the kernel \( H \), where \( H \) is the setwise stabilizer of the sets \( S_i, 1 \leq i \leq r \). Generators for \( G^* \) can be obtained from the generators for \( G \) by replacing the permutation \( \pi \) with the permutation \( \pi^* \) acting on the sets \( S_i \) in accordance with the way these sets are mapped by \( \pi \).

After iterating the two steps, we have determined parts of an imprimitivity structure for \( G \) and, in particular, know \( p \) maximal sets of imprimitivity for \( G \). We explain how to close the "gaps" and complete this partial imprimitivity structure.

Since Proposition 1 guarantees finding a nontrivial system of imprimitivity if one exists, it suffices to show how to obtain generators for \( G^\Gamma \), where \( \Gamma \) is a maximal set of imprimitivity of \( G \). Furthermore, since from a single set \( \Delta \) of imprimitivity we can determine the remaining sets in the system containing \( \Delta \) in \( O(k \cdot n) \) steps, we may
restrict attention to a transitive constituent of \( G_\Gamma \). We will apply the following well-known

**Lemma 7 (Schreier)**

Let \( G = <\pi_1, \ldots, \pi_k> \) be a transitive \( p \)-group of degree \( n \), \( \Gamma_1, \ldots, \Gamma_p \) a maximal system of imprimitivity for \( G \). Let \( \pi_i \) be a generator which does not stabilize \( \Gamma_1 \), and let \( \psi_1, \ldots, \psi_p \) be suitable powers of \( \pi_i \) such that \( \psi_j \) maps \( \Gamma_1 \) onto \( \Gamma_j \) for \( 1 \leq j \leq p \). For any element \( \varphi \in G \), let \( \Phi(\varphi) \) denote the permutation \( \psi_j^{-1} \) where \( \varphi \) maps \( \Gamma_1 \) onto \( \Gamma_j \). Then the setwise stabilizer \( G_{\Gamma_1} \) is generated by the set

\[
K' = \{ \psi_j \pi_i \Phi(\psi_j \pi_i) \mid 1 \leq j \leq p, 1 \leq t \leq k \}.
\]

(Lemma 7 is usually stated in a more general form). Note that \( K' \) contains at most \((k-1)p+1\) nontrivial permutations. Since each generator in \( K' \) stabilizes \( \Gamma_1 \), we obtain a corresponding generating set for the transitive constituent \( H = G_{\Gamma_1}^{(r)} \) by deleting in each generator in \( K' \) the cycles containing points not in \( \Gamma_1 \). Consequently, we obtain a generating set for \( H \) consisting of \( p \cdot k \) permutations of degree \( \frac{n}{p} \). Hence, the total length in symbols of the generating set so obtained for \( H \) is not larger than \( kn \). We now state the resulting algorithm in more detail:

1. Determine the orbits of \( G \).
2. For each transitive constituent \( G^{(A)} \) of \( G \), \( \Delta \) an orbit of \( G \), set \( H \) to \( G^{(A)} \) and execute Steps 3 to 7.
3. Using Proposition 1, determine a nontrivial system of imprimitivity for \( H \).
4. If the system just determined is not maximal, then replace \( H \) with the group \( H^* \) acting on the \( r \) sets of imprimitivity just found for \( H \) and go to Step 3.
5. Let \( \Gamma \) be a maximal set of imprimitivity for \( H \). Using Lemma 7, determine a generating set \( K'' \) for the transitive constituent \( H^{(1)} = H_{\Gamma}^{(r)} \).
6. Replace \( H \) with \( H^{(1)} \). If a maximal system of imprimitivity for \( H \) is known, then return to Step 5. If \( H \) is transitive and of order \( p \), then continue with Step 7, otherwise return to Step 3.
7. Complete the systems of imprimitivity for each distinct set size using the original set of generators for the transitive constituent \( G^{(A)} \) presently processed.
We analyze the time complexity of this algorithm in

**Proposition 2 (Hoffmann, Sims)**

Given a generating set $K = \{n_1, ..., n_k\}$ for the $p$-group $G$ of degree $n$, then a complete imprimitivity structure for $G$ can be determined in $O(k \cdot n \cdot \log^* n \cdot \log_p(n))$ steps.

**Proof** Let $\Delta$ be an orbit of $G$ of length $t \leq n$. The transitive constituent $G^{(\Delta)}$ has degree $t$.

In the worst case, each application of Proposition 1 (in Line 3 above) requires $O(k \cdot t \cdot \log^* (t))$ steps and determines a maximal system of imprimitivity. Observe that the total length of the generating sets determined by Line 5 remains bounded by $k \cdot t$.

A careful implementation of this line can avoid constructing the set $K'$ of Lemma 7 by computing the images only for points in the maximal set of imprimitivity chosen. Hence, Line 5 requires at most $O(k \cdot t)$ steps, rather than $O(k \cdot t \cdot p)$ steps.

Now Lines 3 and 5 need to be done at most a total of $\log_p(t)$ times each and this work then clearly dominates the running time. Hence a total of $O(k \cdot t \cdot \log^* (t) \cdot \log_p(t))$ steps suffices to determine a complete imprimitivity structure for the transitive constituent. Since the sum of the orbit lengths is $n$ and since $t \leq n$, the conclusion follows.

Note that Proposition 2 is a significant improvement over Corollary 5 of Chapter III.

A number of algorithms for $p$-groups can be made much more efficient if the $p$-group is specified by a generating set with special properties. We now define precisely the properties of such useful generating sets, and review a modification of Algorithm 3 of Chapter II for computing them.

**Definition 2**

Let $G$ be a group with a subgroup tower

$$I = G^{(m)} \triangleleft G^{(m-1)} \triangleleft \cdots \triangleleft G^{(0)} = G$$

in which each factor group $G^{(i)}/G^{(i+1)}$, $0 \leq i < m$, has no nontrivial normal subgroup. Then the subgroup tower is a composition series of $G$. If, furthermore, each group $G^{(i+1)}$ setwise stabilizes a maximal set of imprimitivity of $G^{(i)}$, then the composition series is called an imprimitivity series.

It is well known that every $p$-group has a composition series in which each factor group is cyclic of order $p$ (cf. Theorems 9 and 10 of Chapter III). Moreover, every $p$-
group also has an imprimitivity series.

**Definition 3**

A sequence \([\pi_m, \pi_{m-1}, ..., \pi_1]\) of elements of the p-group \(G\) is a *composition* (an imprimitivity) sequence for \(G\) if the groups \(G^{(i)} = \langle \pi_m, ..., \pi_{i+1} \rangle, 0 \leq i < m\), together with \(G^{(m)} = I\), form a composition (an imprimitivity) series for \(G\).

**Proposition 3**

Given a generating set \(K = \{\pi_1, ..., \pi_k\}\) of a transitive p-group \(G\) of degree \(n\) and order \(p^m\), and given a complete imprimitivity structure for \(G\). Then an imprimitivity sequence for \(G\) can be computed in \(O(k \cdot m \cdot n + (p-1)^2 \cdot m^3 \cdot n)\) steps.

**Proof** We determine the sequence using a modification of Algorithm 3 of Chapter II.

First, we renumber the points in the permutation domain such that
\[
\{1, ..., p\}, \{p+1, ..., 2p\}, ..., \{n-p+1, ..., n\};
\{1, ..., p^2\}, ..., \{n-p^2+1, ..., n\};
\vdots
\{1, ..., \frac{n}{p}\}, ..., \left\{\frac{p-1)n}{p} + 1, ..., n\right\}
\]
is the determined imprimitivity structure. This enumeration requires \(O(n)\) steps and is crucial for the efficiency of the subsequent computations. The corresponding change to the generators of \(G\) requires \(O(k \cdot n)\) steps.

We enumerate the sets of imprimitivity in the above structure ordered by decreasing cardinality from 1 to \(s = p \cdot \frac{n-1}{p-1}\). Hence \(\{1, ..., \frac{n}{p}\}\) will be the first set, \(\{n\}\) the last. For \(1 \leq i \leq s\), let \(G^{(i)}\) be the setwise stabilizer of \(\Delta\) in \(G^{(i-1)}\), where \(\Delta\) is the \(i^{th}\) set of imprimitivity in the above enumeration, and \(G^{(0)} = G\). The subgroup tower so defined contains an imprimitivity series for \(G\) which we determine with the familiar sifting and closing under pair product formation of Algorithm 3 of Chapter II.

Clearly, the index of \(G^{(i+1)}\) in \(G^{(i)}\) is either 1 or \(p\). Given \(\pi \in G^{(i)}\), we test in which coset of \(G^{(i+1)}\) the element \(\pi\) of \(G^{(i)}\) is contained: Let \(\Delta_i\) be the \(i+1^{st}\) set of imprimitivity. In the system containing \(\Delta_i\), there are also the sets \(\Delta_0, ..., \Delta_p\) such that the union of these sets with \(\Delta_i\) is again a set of imprimitivity which is stable in \(G^{(i)}\) since \(\Delta_i\) is maximal. Now the coset of \(G^{(i+1)}\) containing \(\pi\) is determined by the set \(\Delta_i\) onto which \(\pi\) maps \(\Delta_i\). This set can be found in constant time by computing the image of a single point \(x \in \Delta_i\) under \(\pi\) because of the enumeration of the points in the
permutation domain. Hence, in constant time the coset containing \( \pi \) can be determined.

We fill a matrix of coset representatives with \( \frac{n-1}{p-1} \) rows and \( p-1 \) columns. Initially, all entries in this matrix are empty. If \( G^{(i+1)} \) has index \( p \) in \( G^{(i)} \), then row \( i \) will eventually contain \( p-1 \) coset representatives for \( G^{(i+1)} \) in \( G^{(i)} \); otherwise, row \( i \) remains empty. (Note that for the coset \( G^{(i+1)} \) the identity serves as coset representative and does not have to be stored). Since the order of \( G \) is \( p^m \), there are eventually \( m \) nonempty rows in the matrix.

Consider sifting an element \( \pi \in G \). For at most each row we have to determine the proper coset in which the partially sifted \( \pi \) lies. If the corresponding coset representative is already known, then we must multiply with its inverse from the right requiring \( O(n) \) steps. This can happen at most \( m \) times. Hence the cost of sifting \( \pi \) is \( O(mn) \).

We have to sift \( k \) generators. Furthermore, the completed table has \( m \cdot (p-1) \) nonempty entries, hence at most \( (p-1)^2 \cdot m^2 \) pair products are sifted. Consequently, the matrix can be fully determined in \( O(k \cdot m \cdot n + (p-1)^2 \cdot m^3 \cdot n) \) steps.

Finally, since the composition factors \( G^{(i)}/G^{(i+1)} \) are cyclic of prime order, it follows that the nonempty entries in the first column of the matrix form an imprimitivity sequence for \( G \).

In the intransitive case, we stabilize all the sets of imprimitivity in one orbit before stabilizing any set in subsequent orbits. A routine extension of the proposition establishes that the same time bound holds for intransitive \( p \)-groups.

The reason for determining an imprimitivity sequence rather than a composition series is that we lack efficient membership tests in the groups arising. For example, one would like to determine a \( p \)-step central series for \( G \) by the same means, but this does not seem to be possible.

Given a complete imprimitivity structure for the \( p \)-group \( G \) and a composition sequence, we now show how to construct new imprimitivity sequences for \( G \) in substantially less time. The crucial observation is the following

**Theorem 4 (Hoffmann)**

Let \( G = H+H\pi+H\pi^2+\cdots+H\pi^{p-1} \) be a \( p \)-group with a subgroup \( H \) of index \( p \) in \( G \). Let \( \Delta \) be an orbit of \( G \) on which \( H \) is transitive, and let \( \{ \Gamma, \Gamma_2, \ldots, \Gamma_p \} \) be a maximal system of
imprimitivity contained in $\Delta$. Let $H = H_0 + H_1 \psi + \cdots + H_p \psi^{p-1}$. Then $G_\varphi = \langle H_\varphi \rangle$, where $\varphi = \pi \psi^{-j}$ and $\psi^j$, $0 \leq j < p$, is the unique power of $\psi$ mapping the set $\Gamma$ onto $\Gamma^\varphi$.

**Proof** Since $\varphi$ stabilizes $\Gamma$, the group $G^{(1)} = \langle H_\varphi \rangle$ is a subgroup of $G_\varphi$. Since $H_\varphi$ has index $p^2$ in $G$ and $G_\varphi$ has index $p$, it suffices to show that $\varphi$ is not in $H_\varphi$. Now if $\varphi = \pi \psi^{-j} \in H_\varphi$, then $\pi = \varphi \psi^j \in H$ and so $H = G$, a contradiction. Hence the index of $H_\varphi$ in $G^{(1)}$ must be $p$, i.e., $G^{(1)} = G_\varphi$. 

We apply the theorem to composition sequences, and obtain

**Corollary 8 (Hoffmann)**

Let $[\pi_m, \ldots, \pi_1]$ be a composition sequence for the $p$-group $G$, $\Gamma$ a maximal set of imprimitivity of $G$, and assume that $\psi = \pi_i$ is the leftmost generator in the sequence not stabilizing $\Gamma$. That is, $\Gamma^\psi \neq \Gamma$ and $\Gamma^{\pi_k} = \Gamma$ for all $k \geq i$. For $k < i$ let $\varphi_k = \pi_k \psi^{-j}$, where $\psi^j$, $0 \leq j < p$, is the unique power of $\psi$ mapping $\Gamma$ to $\Gamma^{\pi_k}$. Then $[\pi_m, \ldots, \pi_{i+1}, \varphi_{i-1}, \ldots, \varphi_1, \pi_i]$ is also a composition sequence for $G$ and $[\pi_m, \ldots, \pi_{i+1}, \varphi_{i-1}, \ldots, \varphi_1]$ is a composition sequence for $G_\varphi$.

**Proof** Since $\Gamma$ is maximal, the definition of the $\varphi_k$, $1 \leq k < i$, makes sense. The conclusion now follows with a simple induction from Theorem 4.

The corollary is a significant improvement over (the more general method of) Lemma 7. It exploits essentially that the composition factors of $p$-groups are cyclic of prime order. Corollary 8 now implies the following immediate and useful corollaries:

**Corollary 9 (Hoffmann)**

Given a composition sequence $[\pi_m, \ldots, \pi_1]$ for the $p$-group $G$ of degree $n$, and given a complete imprimitivity structure for $G$. Then an imprimitivity sequence for $G$ can be determined in $O(m^2n)$ steps.

**Proof** In $O(mn)$ steps the points in the permutation domain can be enumerated as in the proof of Theorem 4, and the generators $\pi_k$ in the composition sequence can be adjusted accordingly.

We repeatedly apply Corollary 8, successively stabilizing maximal sets of imprimitivity. If a set is not yet stable, we obtain a composition sequence for the stabilizer.

There are $O(n)$ scans of the sequence to determine whether the next set of imprimitivity is already stable, each scan requiring $O(m)$ steps. Exactly $m$ times we find that in the sequence considered the next maximal set of imprimitivity is not yet stable, and each time this happens, the computation of the new sequence requires at most $O(mn)$ steps. Therefore, a total of $O(m^2n)$ steps suffice to construct an
imprimitivity sequence. •

**Corollary 10 (Hoffmann)**

Given a composition sequence \([\pi_0, \ldots, \pi_1]\) for the p-group G of degree n, and given a complete imprimitivity structure for G. Then an imprimitivity sequence for \(G_x\) can be determined in \(O(mn \log_p(n))\) steps.

**Proof** Without loss of generality, we may assume that \(G_x \neq G\). Let \(\Gamma_1, \ldots, \Gamma_r\) be the sets of imprimitivity in the given imprimitivity structure of G, such that each set contains x and the sets are ordered by decreasing cardinality. Observe that \(r \leq \log_p(n)\). Successively stabilize these sets using Corollary 8. The time bound now follows readily from the proof of Corollary 8.

The general procedure for finding a generating set for a point stabilizer in p-groups is given by Proposition 3 above. The vast improvement in efficiency in Corollary 10 demonstrates the usefulness of specifying p-groups by composition sequences and complete imprimitivity structures.

We develop next an intersection algorithm for p-groups. We first sketch a recursive procedure for intersecting two p-groups and then reorganize the sequence of computation in order to minimize the incurred overhead (and with it the running time). In particular, we will exploit the properties of imprimitivity sequences in order to reduce the amount of work to be performed. Note that for the resulting group the intersection algorithm finds an imprimitivity sequence and a complete imprimitivity structure (the structure of the p-groups intersected is inherited).

Let \(G, H < \text{Sym}(X)\), and let \(G \times H\) be their (external) direct product acting on \(X \times X\), two disjoint copies of \(X\). The elements of \(G \times H\) are pairs \(\bar{\pi} = (\pi_1, \pi_2)\), where \(\pi_1 \in G\) and \(\pi_2 \in H\). Let \(A\) be a subgroup of \(G \times H\). Then the **first projection**, \(\text{proj}_1(A)\), is the set

\[
\{ \pi_1 \in G \mid \exists \pi_2 \in H, (\pi_1, \pi_2) \in A \}.
\]

The **second projection**, \(\text{proj}_2(A)\), is the set

\[
\{ \pi_2 \in H \mid \exists \pi_1 \in G, (\pi_1, \pi_2) \in A \}.
\]

It is easily shown that both projections are groups, hence \(\text{proj}_1(A) < G\) and \(\text{proj}_2(A) < H\). The two projections are also called the G- and the H-component of \(A\), respectively.
Let $A < \text{Sym}(X) \times \text{Sym}(X)$, $\bar{\pi} \in \text{Sym}(X) \times \text{Sym}(X)$, $\Delta$ any subset of $X$. We define the set

$$J(\bar{\pi}, \Delta) = \{ (\alpha_1, \alpha_2) \in A\bar{\pi} \mid \forall x \in \Delta, x^{\alpha_1} = x^{\alpha_2} \}$$

That is, $J(\bar{\pi}, \Delta)$ is the set of all $\bar{\alpha}$ in the right coset $A\bar{\pi}$ of $A$ which map the points in $\Delta$ identically in both projections. The following lemma is elementary and explains the basic intuition about the set function $J$:

**Lemma 8 (Luks)**

Let $A < \text{Sym}(X) \times \text{Sym}(X)$, $\bar{\pi} = (\pi_1, \pi_2) \in \text{Sym}(X) \times \text{Sym}(X)$, and $G, H < \text{Sym}(X)$. Then the following is true:

(a1) If $A \Delta X$ is stable in $\text{proj}_1(A)$, then $J(A, \Delta)$ is a group.

(a2) If $\Delta \subseteq X$ is stable in $\text{proj}_1(A)$, then $J(A, \Delta)$ is either empty, or it is a right coset of $J(A, \Delta)$.

(a3) $J((G \times H)\bar{\pi}, X) = \{ (\varphi, \varphi) \mid \varphi \in G\pi_1 \cap H\pi_2 \}$. In particular, $J(G \times H, X)$ is isomorphic to $G \cap H$.

**Proof**

(a1): Let $(\alpha_1, \alpha_2)$ and $(\beta_1, \beta_2)$ be elements of $J(A, \Delta)$. Since $\Delta$ is stable in $\text{proj}_1(A)$, for all $x \in \Delta$, the image points $x^{\alpha_1}$ and $x^{\beta_1}$ are in $\Delta$, hence $x^{\alpha_1\beta_1^{-1}}$ is also in $\Delta$. Therefore, $x^{\alpha_1\beta_1^{-1}} = x^{\alpha_2\beta_2^{-1}}$, and so $(\alpha_1, \alpha_2)(\beta_1, \beta_2)^{-1}$ is also in $J(A, \Delta)$.

(a2): Let $\bar{\alpha} \in J(A, \Delta)$, $\bar{\varphi} \in J(A, \Delta)$. Since $\Delta$ is stable in $\text{proj}_1(A)$, we have $\bar{\alpha}\bar{\varphi} = (\alpha_1\varphi_1, \alpha_2\varphi_2) \in J(A, \Delta)$. Hence $J(A\bar{\pi}, \Delta)$ contains the right coset $J(A, \Delta)\bar{\varphi}$. Next, if $\bar{x} \in J(A\bar{\pi}, \Delta)$, then, for all $x \in \Delta$, $x^{\pi_1^{-1}}$ is in $\Delta$, hence $x^{\pi_1^{-1}} = x^{\varphi_2^{-1}}$. But $\bar{x} x_1^{-1}$ is in $A$ and hence in $J(A, \Delta)$. Thus $J(A\bar{\pi}, \Delta)$ is the right coset $J(A, \Delta)\bar{\varphi}$.

(a3): Trivial. *

The overall design of a recursive intersection algorithm is based on the following lemma:

**Lemma 9 (Luks)**

Let $A < \text{Sym}(X) \times \text{Sym}(X)$, $\bar{\pi} = (\pi_1, \pi_2) \in \text{Sym}(X) \times \text{Sym}(X)$. Then the following holds:

(b1) If $A = A' + A' \chi_2 + \cdots + A' \chi_k$ and $\Delta \subseteq X$, then

$$J(A, \bar{\pi}, \Delta) = J(A', \bar{\pi}, \Delta) + J(A' \chi_2, \bar{\pi}, \Delta) + \cdots + J(A' \chi_k, \bar{\pi}, \Delta).$$

(b2) If $\text{proj}_1(A)$ setwise stabilizes the disjoint sets $\Gamma$ and $\Gamma'$, where $\Gamma, \Gamma' \subseteq X$, then

$$J(A, \bar{\pi}, \Gamma + \Gamma') = J(A, \bar{\pi}, \Gamma, \Gamma').$$

(b3) If $\text{proj}_1(A)$ fixes the point $x \in X$, then $J(A, \bar{\pi}, \{x\}) = A\bar{\varphi} \bar{\pi}$ iff there exists a $\bar{\varphi} = (\varphi_1, \varphi_2) \in A$ such that $x^{\varphi_1 \varphi_2^{-1}} = x^{\varphi_2^{-1}}$. 

Proof (b1) is trivial and (b2) is straightforward, so we only show (b3): Here we seek all elements \((a_1,a_2) \in A\) such that \(x^{a_1a_2} = x^{a_2a_2}\). Now \(x^{a_1} = x\), hence we need to solve \(x^{a_1a_2} = x^{a_2}\) for some \(a_2 \in \text{proj}_2(A)\). Clearly, if no such \(a_2\) exists, then \(J(A,x)\) must be empty. So, let \(\varphi = (\varphi_1,\varphi_2)\) be an element of \(A\) such that \(x^{\varphi_1\varphi_2} = x^{\varphi_2}\). Clearly then, \(x^{\varphi_1\varphi_1} = x^{\varphi_2\varphi_2}\), and so, by Lemma 8 (b2), \(J(A,x) = J(A,x)\varphi\). Finally, since \(\text{proj}_1(A)\) fixes \(x\), we have \(J(A,x) = A_x\), where \(A_x\) is the stabilizer of \(x\) in both copies of \(X\) on which \(A\) acts.

Note the close analogy with the rules for computing \(S_T(G\pi,Z)\) when determining setwise stabilizers.

We wish to design an efficient algorithm for computing \(J((G\times H)\pi,X)\), where both \(G\) and \(H\) are \(p\)-groups contained in \(\text{Sym}(X)\). Based on the rules (b1) - (b3) of Lemma 9, we can design a recursive algorithm analogous to Algorithm 2 of Section 3 above. Because of the imprimitivity structure of \(p\)-groups, one alternates rules (b1) and (b2) working recursively through the imprimitivity structure of \(G\) until the base case, using rule (b3), is reached. Then a point stabilizer has to be found, and this is done using Corollary 10. The details are fairly straightforward, and a careful implementation would result in an \(O(n^4 \log p(n))\) method for intersecting two \(p\)-groups of degree \(n\).

We improve the basic recursive algorithm by reorganizing the computation such that all base case problems \(J(A\pi,x)\) are processed together for each point \(x \in X\). This reduces both the overhead during the recursion proper and the time spent for each base case, since we now determine a point stabilizer \(A_x\) or a set stabilizer \(A'\) only \(O(n)\) times. The overhead computation can be further sped up using Corollaries 8 and 10, and this second improvement becomes a crucial factor in the timing of the resulting algorithm.

The reorganization of the computation for \(J((G\times H)\pi,X)\) is developed as follows: Since \(G\) is a \(p\)-group, we may visualize the imprimitivity structure of \(G\) as a forest of complete \(p\)-ary trees, where the \(p\) sons of each interior vertex are ordered cyclically (cf. Chapter III). If \(G\) has \(s\) orbits, this imprimitivity forest of \(G\) consists of \(s\) trees, the leaves of each tree comprising an orbit of \(G\). We assume that the points in the permutation domain have been renumbered as in Proposition 3 above, i.e., the trees are drawn as planar graphs and the cyclic ordering of the sons of each vertex is \(w_1 < w_2 < \cdots < w_p < w_1\), where the \(p\) sons \(w_i\) have been enumerated left to right. The
leaves of the trees are also enumerated from left to right in ascending order, and corresponding to each interior vertex v of a tree T there is a set Δv of imprimitivity for G consisting precisely of the leaves of the subtree Tv rooted in v.

For simplicity, assume that G is transitive, i.e., the imprimitivity forest of G consists of a single tree T, and let us visualize the process of computing J((G×H)(ψ,X)). With every vertex v of T we associate a subproblem J(Aχ,Δv) which has to be solved, where proj1(A) setwise stabilizes Δv. If v is a leaf, then Δv = {v}, i.e., we have to solve the base case using rule (b3).

Let w1, ..., wp be the sons (from left to right) of the interior vertex v, Γi, 1 ≤ i ≤ p, the set of leaves of the subtree Twi. Then proj1(A) acts transitively on Δv iff, for at least one generator α of A and every point x ∈ Γ1, xα1 ∈ Γ1. In this case, we have to apply rule (b1) and find the largest subgroup A' of A such that proj1(A') stabilizes Γ1. Clearly the index of A' in A is p. So, let

\[ A = A' + A'\bar{\varphi}_2 + \cdots + A'\bar{\varphi}_p. \]

Applying rule (b2) also, we now obtain the p subproblems associated with w1:

\[ J(A\chi,Γ_1) = B\eta_1, \]

\[ J(A'\bar{\varphi}_2\chi,Γ_1) = B\eta_2, \]

\[ \vdots \]

\[ J(A'\bar{\varphi}_p\chi,Γ_1) = B\eta_p; \]

next, associated with w2, we obtain

\[ J(B\eta_1,Γ_2) = C\xi_1, \]

\[ \vdots \]

\[ J(B\eta_p,Γ_2) = C\xi_p; \]

and similarly for the remaining sons of v, provided none of these sets is empty.

If proj1(A) is intransitive on Δv, then it must stabilize the sets Γ1, hence here we obtain the subproblem

\[ J(A\chi,Γ_1) = B\eta \]

associated with w1,

\[ J(B\eta,Γ_2) = C\xi \]

associated with w2, and so on for the remaining sons.

This situation suggests guiding the computation of J((G×H)(ψ,X)) by a single depth-first traversal of the imprimitivity tree (forest) of G. For this traversal, we
keep two lists, $R$ and $K$, where $R$ is a list of coset representatives (initially $R = \{\chi\}$), and $K$ is a list of generators, initially for $G \times H$, and later for subgroups $A$ of $G \times H$. Reaching a vertex $v$ from above with $K$ and $R$ shall mean that we wish to solve each of the problems $J(A, \Delta_v)$, where $A = \langle K \rangle$, and $\pi_1(A)$ setwise stabilizes $\Delta_v$, and $\chi$ is in $R$. When leaving the vertex $v$ to return to its father with the lists $K'$ and $R'$, this shall mean that $J(<K>, \Delta_v) = <K'>$, and that for each $\chi \in R$ there is, correspondingly, an entry $\tilde{\chi} \in R'$ such that $J(<K>, \Delta_v) = <K'>\tilde{\chi}$ or that $J(<K>, \Delta_v)$ is empty. In the latter case, the entry $\tilde{\chi}$ in $R'$ is "marked".

Now assume that $\pi_1(A)$ is transitive on $\Delta_v$, and the $p$ subproblems $J(A', \Delta_v)$ of $J(A, \Delta_v)$ are the nonempty cosets $B' \tilde{x}_j$ of $B' = J(A', \Delta_v)$, $1 \leq j \leq p$. Then, by Lemma 8 (a2), the union of these cosets is the coset $J(A, \Delta_v)\tilde{x}_1$. Here we will show that $J(A, \Delta_v) = <B', \tilde{x}_1\tilde{x}_1^{-1}>$.

At all times during the computation, we will arrange the generators in $K$ such that $K = \{(\pi_r, 2), ..., (\pi_m, 2), (\pi_m, 1, \pi_m), ..., (\pi_1, 1, \pi_1)\}$ is an imprimitivity sequence for $A = <K>$ and $[\pi_m, ..., \pi_n]$ is an imprimitivity sequence for $\pi_1(A)$. We call such a generating set a set in canonical form. The initial generating set for $G \times H$ is of the form

$$\{(\pi_r, 2), ..., (\pi_m, 2), (\pi_1, 1, \pi_1), ..., (\pi_1, 1)\}$$

where $r = m + m'$, $[\pi_r, ..., \pi_m]$ is an imprimitivity sequence for the group $H$ of order $p^m$ and $[\pi_m, ..., \pi_1]$ is an imprimitivity sequence for the group $G$ of order $p^m$. Clearly the initial generating set is in canonical form.

Throughout the processing of the interior nodes in the imprimitivity forest of $G$, we show below that this format can be maintained dynamically for all intermediate subgroups $A$ of $G \times H$ which arise. This includes the processing at the leaves with the base case of the recursion.

The entries in $R$ have to be grouped recursively, so that the process of "gluing" cosets can be carried out efficiently. Here it may be helpful to visualize $R$ as the list of leaves of a complete $p$-ary tree.

Let $A = <K>$ be a subgroup of $G \times H$, where $G < \text{Sym}(X)$ is a $p$-group, $H < \text{Sym}(X)$, and $K$ is in canonical form. Assume that $\Delta_c X$ is an orbit of $\tilde{A} = \pi_1(A)$, and that $\Gamma_1, \ldots, \Gamma_p$ is a maximal system of imprimitivity of $\tilde{A}$ contained in $\Delta$. By carrying out
the operations of Corollary 8, it is clear that we can determine the subgroup \( A' \) of \( A \) such that \( \tilde{A}' = \text{proj}_1(A') \) setwise stabilizes \( \Gamma_1 \). In particular, if \((\pi_{i,1}, \pi_{i,2})\) is the leftmost generator in \( K \) such that \( \Gamma_1 \tilde{\pi}_{i,1} \neq \Gamma_1 \), then the sequence

\[
((1), \pi_{i,2}), \ldots, (\pi_{i+1,1}, \pi_{i+1,2}), \tilde{\psi}_{i-1}, \ldots, \tilde{\psi}_1, (\pi_{i,1}, \pi_{i,2})
\]

is also a canonical sequence for \( A \), and by dropping \((\pi_{i,1}, \pi_{i,2})\) we obtain a canonical sequence for \( A' \).

**Lemma 10**

For the subgroup \( A < G \times H \), where \( G \) is a p-group, let \( A = A' + A_2 + \cdots + A_p \), where \( \text{proj}_1(A') = \text{proj}_1(A) \). Assume that, for \( 1 \leq i \leq p \), \( B_{\xi_i} \) is the nonempty set \( J(A', A) \), where \( A \) is setwise stable in \( \text{proj}_1(A) \) and \( \varphi_1 = ((1), (1)) \). If \( K \) is a generating set for \( B \) in canonical form, then \( K' = \{K, \tilde{\xi}_{2}\}^{-1} \) is a generating set for \( J(A, A) \) and is also in canonical form.

**Proof** By Lemma 9 (b1), \( J(A, \varphi_2) = B_{\xi_1} + \cdots + B_{\xi_p} \). Now \( A' \) has index \( p \) in \( A \) and the sets \( B_{\xi_1} \) are not empty. By Lemma 9 (a2), therefore, \( B = J(A', A) \) has index \( p \) in \( C = J(A, A) \). Now \( \tilde{\xi}_1 = \tilde{\eta}_1 \tilde{\psi} \) and \( \tilde{\xi}_2 = \tilde{\eta}_2 \tilde{\phi}_2 \tilde{\psi} \), where \( \tilde{\eta}_1 \) and \( \tilde{\eta}_2 \) are in \( A' \). Hence \( \tilde{\xi}_{2}^{-1} = \tilde{\eta}_2 \tilde{\phi}_2 \tilde{\eta}_1^{-1} \in B < A' \) implies that \( \tilde{\phi}_2 \in A' \), a contradiction to the assumption that \( A' \) has index \( p \) in \( A \). Since \( B \) has index \( p \) in \( C \), it follows that \( C = \langle K, \tilde{\xi}_{2}\rangle \). Since \( \text{proj}_1(A') \neq \text{proj}_1(A) \), it follows similarly that \( \text{proj}_1(\tilde{\xi}_{2}^{-1}) = \tilde{\eta}_{2} \tilde{\phi}_{2} \tilde{\eta}_{1}^{-1} \in \text{proj}_1(A') \). Hence \( \text{proj}_1(B) \) has index \( p \) in \( \text{proj}_1(C) \), i.e., \( K' = \{K, \tilde{\xi}_{2}\}^{-1} \) is in canonical form.

Having outlined the operations required to implement rules (b1) and (b2) during a traversal of the interior vertices of the imprimitivity forest of \( G \times H \), we now consider the implementation of the base case of the recursion when reaching a leaf \( v \). We reach \( v \) with a canonical generating set for \( A < G \times H \), where \( \text{proj}_1(A) \) fixes \( v \), and a list \( R \) of coset representatives. The length of \( R \) is \( p^h \), \( h \) the height of the tree containing \( v \), hence is bounded by \( n \), the degree of \( G \) and of \( H \). Because of rule (b3), we must do the following at \( v \):

(a) Determine the orbit of \( x \) in \( \text{proj}_2(A) \).

(b) For each orbit point \( y \), find an element \( \varphi \in A \) such that \( x^{\varphi} = y \).

(c) Using Corollary 10, find a canonical generating set for \( A_x \).

(d) For each (unmarked) \( \bar{\pi} \in R \), test whether \( x^{\bar{\pi}^{-1}} \) is in the orbit of \( x \) in \( \text{proj}_2(A) \).

If so, we multiply \( \bar{\pi} \) with \( \varphi \) from the left, otherwise we mark \( \bar{\pi} \).
We now specify the details of the intersection algorithm formally. In the following, both G and H are p-groups of degree n contained in Sym(X). We assume that the groups are specified by the imprimitivity sequences K_G and K_H, and that the imprimitivity structures for G and H are given.

We traverse each tree in the imprimitivity forest for G in preorder. The root of the first tree is visited with K and an initial list R = [(0,0)]. If K' and R' is the result of traversing the tree T, then the root of the next tree is visited with those two lists. The computation for the traversal is now as follows:

**Algorithm 3 (Intersection of two p-Groups)**

**Comment** Only the recursive procedure Visit is specified which is in charge of traversing the next tree in the imprimitivity forest of G. The global initialization and the driving routine are straightforward.

**Method**

1. procedure visit vertex v with lists K and R;
2. begin
3. if v is not a leaf then begin
4. let x be the leftmost leaf of T_v;
5. let h be the height of vertex v (leaves have height 0);
6. if there is a generator \( \varphi = (\varphi_1, \varphi_2) \) such that \( x^\varphi_1 \geq x + p^{h-1} \) then begin
   comment proj_1(A), A = <K>, is transitive on \( \Delta \), the set of leaves of T_v;
7. using Corollary 8, find a generating set K1 in canonical form for A', the largest subgroup such that proj_1(A') stabilizes \( \Gamma \), the set of leaves of T_w, w a son of v;
8. for each \( \bar{\varphi} \in R \) do
9. replace \( \bar{\psi} \) with the p-tuple \( \bar{\psi}, \bar{\varphi} \bar{\psi}, \bar{\varphi}^2 \bar{\psi}, \ldots, \bar{\varphi}^{p-1} \bar{\psi} \), where \( \bar{\varphi} \) is the leftmost generator in K not stabilizing \( \Gamma \) in the first projection;
10. end (if there is a generator)
11. else
12. K1 := K;
13. \textbf{for} \(i := 1\) \textbf{to} \(p\) \textbf{do} begin
14.  visit son \(w_i\) of \(v\) with lists \(K_1\) and \(R\), and let \(K_2\) and \(R_2\) be the returned lists;
15.  \(K_1 := K_2;\)
16.  \(R := R_2;\)
17.  \textbf{end};
18.  \textbf{if} \(<K>\) is transitive on \(A\) \textbf{then} begin
19.    \textbf{if} there is a \(p\)-tuple \(\vec{c}_1, \ldots, \vec{c}_p\) in \(R\) such that each of its members is unmarked \textbf{then} begin
20.      \textbf{comment} new generator found, glue cosets together;
21.      append \(\vec{c}_p J_1^{-1}\) to \(K_1;\)
22.    \textbf{end};
23.    \textbf{for} each \(p\)-tuple \(\vec{c}_1, \ldots, \vec{c}_p\) in \(R\) \textbf{do}
24.      \textbf{if} there is at least one unmarked \(\vec{c}_i\) \textbf{then}
25.        replace the tuple with \(\vec{c}_i\)
26.      \textbf{else}
27.        replace the tuple with \(\vec{c}_1;\)
28.      \textbf{end} (if \(<K>\) is transitive);
29. \textbf{end (if \(v\) is not a leaf)}
30. \textbf{else} begin
31.    \textbf{comment} \(v\) is a leaf;
32.    determine a canonical generating set \(K_1\) for \(A_v\) using Corollary 10, where \(A = <K>;\)
33.    calculate the orbit of \(v\) in \(\text{proj}_2(A)\), and for each orbit point \(u\) determine \(\bar{a} \in A\) such that \(v \bar{a}_2 = u;\)
34.    \textbf{for} each unmarked \(\bar{\pi} = (\pi_1, \pi_2)\) in \(R\) \textbf{do}
35.      \textbf{if} \(w = v \pi_1 \pi_2^{-1}\) is in the orbit of \(v\) in \(\text{proj}_2(A)\) \textbf{then}
36.        replace \(\bar{\pi}\) with \(\bar{\varphi} \bar{\pi}\), where \(\bar{\varphi}\) is the coset representative of \(A_v\) in \(A\) corresponding to the orbit point \(w;\)
37.      \textbf{else}
38.        \textbf{mark} \(\bar{\pi};\)
39. \textbf{end;}
40. \textbf{end.}
The correctness of the algorithm is clear from Lemmata 9 and 10, except perhaps for Line 9. From Corollary 8 we know that $K_1 \cup \bar{\varphi}$ is a generating set for $\Lambda = \langle K \rangle$. Now $\text{proj}_1(A')$ setwise stabilizes $\Gamma$, the set of leaves in the tree $T_\omega$, $w$ the leftmost son of $v$, whereas $\text{proj}_1(A)$ is transitive on $\Delta$. It follows that $\varphi_1$, the first projection of $\bar{\varphi}$, cyclically permutes the sets $\Gamma = \Gamma_1, \ldots, \Gamma_p$, where $\Gamma_i$ is the set of leaves of the subtree rooted in the $i^{th}$ son of $v$. Consequently, the permutations $\varphi_1^i$, $0 \leq i < p$ are in distinct cosets of $\text{proj}_1(A')$ in $\text{proj}_1(A)$, hence the $\bar{\varphi}_i^j$ are in distinct cosets of $A'$ in $A$. Now the correctness of Line 9 follows from Lemma 9 (b1).

We account for the time required by the above procedure by determining the cost of the computation performed at each node in the imprimitivity forest. It is clear that we can construct each imprimitivity tree for $G$ such that, for each interior vertex $v$, its height $h$, the quantity $p^{h-1}$, and the leftmost leaf in the subtree $T_v$ can be determined in constant time. The cost of this is charged to the preconditioning step and does not alter its time bound. Moreover, as already mentioned, the number of permutations in $R$ never exceeds $p^h \leq n$, where $h$ is the height of the highest tree in the imprimitivity forest (since for each time the length of $R$ increases by a factor of $p$ when visiting a vertex $v$, it is reduced to its previous size when leaving $v$).

Let $v$ be any interior vertex. We execute Lines 4 - 29. The dominant steps are

Line 7 requiring $O(n^2)$ steps, since a canonical generating set for $G \times H$ has contains less than $2n$ permutations,

Lines 8-9 also requiring $O(n^2)$ steps, and

Lines 22-26 again requiring $O(n^2)$ steps.

Hence, the total work required at the $O(n)$ interior vertices is at most $O(n^3)$.

We now analyze the number of steps required to do the leaf processing. First, observe that Line 32 can be implemented in $O(n^2)$ steps. To see this, recall that we examine $O(n)$ generators to compute the orbit of $v$. Recalling Algorithm 4 of Chapter II, it is clear that we may compute a set of Schreier vectors for each orbit point $u$. Now if $\bar{x}_1 \bar{x}_2 \cdots \bar{x}_k$ is a Schreier vector for the orbit point $u$, it follows that $\bar{x}_1 \cdots \bar{x}_j$ is a Schreier vector for some other orbit point for each $1 \leq j < k$. Hence all Schreier vectors can be evaluated in a total of $O(n^2)$ steps, since $k$ is bounded by $n$. It follows, that Lines 32 - 37 can be implemented in $O(n^2)$ steps.

By Corollary 10, Line 31 may require $O(n^2 \log_p(n))$ steps. As there are $n$ leaves to be processed, it would seem that the intersection of two $p$-groups of degree $n$ may
require up to $O(n^3 \cdot \log_p(n))$ steps. However, by a more careful analysis we can demonstrate that $O(n^3)$ steps suffice.

We now derive the $O(n^3)$ upper bound on the time required to intersect two p-groups $G$ and $H$ of degree $n$. We assume that $G$ and $H$ have orders $p^m$ and $p^{m'}$. Rather than timing Line 31 for each leaf, we will determine the combined total of operations required by this line for all leaves. Recall that the initial generating set for $G \times H$ is of the form

$$[((), \pi_{r, 2}), \ldots, (((), \pi_{m+1, 2}), (\pi_{m, 1}), \ldots, (\pi_{1, 1})]$$

where $r = m + m'$, $[\pi_{r, 2}, \ldots, \pi_{m+1, 2}]$ is an imprimitivity sequence for $H$, and $[\pi_{m, 1}, \ldots, \pi_{1, 1}]$ is an imprimitivity sequence for $G$. Let us call generators of the form $(((), \pi)$ type 2 generators and generators of the form $(\pi, ())$ type 1 generators. Finally, call a generator of the form $(\varphi, \psi), \varphi \neq (), \psi \neq ()$, a generator of type 3. Throughout the computation of the intersection algorithm, the list $K$ of generators will consist, in sequence, of generators of type 2 followed by generators of type 1 which in turn are followed by generators of type 3. The type 3 generators are computed from coset representatives (cf. Lemma 10), and their number is at most equal to the number of interior vertices in the imprimitivity forest of $G$ which have been visited completely.

We note that $\text{proj}_2(A)$ is determined only by generators of types 2 or 3. Moreover, since $\text{proj}_2(A) < H$, the combined number of generators of types 2 and 3 cannot exceed $m'$ at any time.

Now consider computing an imprimitivity sequence for the groups $A_x$ from an imprimitivity sequence for $A$ using Corollary 10. Here we determine, successively, setwise stabilizers of maximal sets of imprimitivity of $H$ containing $x$. In doing so, there are two possibilities: Either the set is already stable. In this case we perform $O(m')$ steps, since we only compute the image of a single point for each of the (at most) $m'$ generators of type 2 or 3. We encounter this situation at most $\log_p(n)$ times at each leaf. Hence this part requires at most $O(m' \cdot \log_p(n))$ steps for all leaves. Otherwise, if the set is not yet stable, we drop one generator (of type 2 or type 3) and adjust the remaining generators in $O(m' \cdot n)$ steps. Since one generator is dropped in the operation, this situation can be encountered a total of $m + m'$ times, since at most $m'$ generators of type 2 can be dropped and at most $m$ generators of type 3 can be introduced. Consequently, substep (c) requires a combined total of $O(m' \cdot (m + m') \cdot n)$ steps, which is $O(n^3)$. In summary, the above argument establishes the following
**Theorem 5 (Hoffmann)**

Let $G, H \leq \text{Sym}(X)$ be $p$-groups of degree $n$. Given complete imprimitivity structures and imprimitivity sequences for both groups, and given $\pi \in \text{Sym}(X) \times \text{Sym}(X)$, then $J((G \times H)\pi, X)$ can be determined in $O(n^3)$ steps.

**Corollary 11**

Let $G, H < \text{Sym}(X)$ be two $p$-groups of degree $n$. Given complete imprimitivity structures and imprimitivity sequences for $G$ and $H$, then an imprimitivity sequence for $G \cap H$ may be determined in $O(n^3)$ steps.

Algorithm 3 can be easily modified to perform the work of the procedure STABILIZE in Algorithm 2 of Section 3 above, i.e., to compute setwise stabilizers. Exploiting the strong similarity in the recursive rules for the functions $J$ and $S_Y$, we merely have to alter the leaf processing (Lines 31 - 37 of Algorithm 3 above).

The new leaf processing is much simpler due to the simpler rule for the recursion base case:

$$S_Y(G\pi, \{v\}) = G\pi \quad \text{iff} \quad v^\pi \in Y \text{ whenever } v \in Y$$

(provided, of course, that $G$ fixes $v$). When visiting the leaf $v$, we must compute $w = v^\psi$ for each of the (up to $n$) elements in $R$. If $v$ is in $Y$ but $w$ is not, or vice versa, then $\psi$ must be marked, otherwise no further processing is required. This takes a total of $O(n)$ steps. Now if $G$ fixes $x$, then $S_Y(G, \{x\}) = G$, and so the list of generators is never affected, hence it remains, in particular, an imprimitivity sequence for $G$. Consequently, the total amount of work at the leaves is $O(n^3)$. In summary, we have just shown

**Proposition 4 (Hoffmann)**

Let $G < \text{Sym}(X)$ be a $p$-group of degree $n$. Given a complete imprimitivity structure for $G$ and an imprimitivity sequence, then a composition sequence for the setwise stabilizer $G_T$ of $Y \subseteq X$ in $G$ can be determined in $O(n^3)$ steps.

### 4.2. The Imprimitivity Problem for 2-Groups

We consider the problem of determining the largest subgroup $H$ of a given 2-group $G$ such that $H$ has a prescribed (minimal) system of imprimitivity. Group-theoretically this problem is of interest in its own right, but it has a geometric
interpretation which motivates it as a suitable abstraction of determining the automorphisms of very simple binary cone graphs. We begin with a formal statement of the problem and develop its solution before applying it to such cone graphs.

**Problem 3 (Imprimitivity Problem for 2-Groups)**

Given a complete imprimitivity structure and an imprimitivity sequence for the 2-group \( G \) of degree \( n \), and given a partition \( S \) of the permutation domain into blocks of size 2 and at most one block of size 1: Find the largest subgroup \( H \) of \( G \) such that \( S \) is a minimal system of imprimitivity for \( H \).

Note that it suffices to consider the case where \( n \) is even. For if \( G \) has odd degree, then \( H \) must be a subgroup of \( G_x \) where \( \{x\} \) is the singleton block in \( S \). Hence we may consider the problem in \( G_x \) viewed as a permutation group of even degree. We will give an \( O(n^3) \) algorithm for solving the imprimitivity problem for 2-groups.

Let \( G \triangleleft \text{Sym}(L) \) be a 2-group of even degree, \( S \) a partition of \( L \) into blocks of size 2. Let \( L^2 \) be the set of all subsets of \( L \) of cardinality 2, \( G' \) the isomorphic image of \( G \) in \( \text{Sym}(L^2) \). Here the isomorphism \( ' \) is provided by the induced action of \( G \) on the set \( L^2 \). Using the setwise stabilizer algorithm, the imprimitivity problem for \( G \) may be solved as follows:

1. Determine an imprimitivity sequence and a complete imprimitivity structure for \( G' \).
2. Determine \( G'_S \) and observe that \( H' = G'_S \).
3. Find \( H \triangleleft \text{Sym}(L) \) isomorphic to \( G'_S \).

Since \( G' \) has degree \( \binom{n}{2} \), an implementation of this procedure seems to require \( O(n^6) \) steps (cf. Proposition 4). We plan to improve this bound by considering the imprimitivity problem as an intersection problem of \( G \) with a group isomorphic to \( C_2 \cup S_m \), where \( m = \frac{n}{2} \).

The group-theoretic structure of \( C_2 \cup S_m \) is sufficiently complex so that we do not wish to intersect this group with \( G \) directly. Instead, we plan to determine a 2-group \( B \triangleleft \text{Sym}(L) \) such that \( B \cap G = H \). It so happens that we can find a suitable 2-group \( B \) in \( O(n^2 \log_2(n)) \) steps. Therefore, by Corollary 11, we can determine \( H \) in \( O(n^3) \) steps.

The group \( B \) will be constructed by considering the induced action of \( G \) on the set \( L^2 \). However, instead of working with \( G' \), we will consider the group \( \tilde{G} \), a Sylow 2-subgroup of \( \text{Sym}(L^2) \) containing \( G' \). We will show that \( \tilde{G} \) can be determined from a
complete imprimitivity structure for \( G \), in \( O(n^2) \) steps. Next, we determine \( \tilde{G}_S \), the setwise stabilizer of \( S \) in \( \tilde{G} \). Since \( \tilde{G} \) is a Sylow 2-subgroup of a symmetric group, a representation of \( \tilde{G}_S \) can be computed in \( O(n^2 \log_2(n)) \) steps (cf. Theorem 17 of Chapter III). Note that \( \tilde{G}_S \) contains as subgroup \( G'_S \), the isomorphic image of the sought group \( H \) in Sym(\( L^2 \)).

We restrict \( G_S \) to \( S \) thereby obtaining a 2-group \( \tilde{D} \). More precisely, a composition sequence for \( \tilde{D} \) can be computed in \( O(n^2) \) steps from the (implicit) representation of \( G_S \). Now recall that \( S \) partitions \( L \), hence the blocks in \( S \) do not "overlap". Consequently, there must be a 2-group \( D < \text{Sym}(L) \) with \( S \) as a minimal system of imprimitivity such that the induced action of \( D \) on \( S \) is \( \tilde{D} \). We take the largest such group \( B \), which is \( B = C_2 \cap \tilde{D} \). Since \( \tilde{D} \) is the restriction of \( G_S \) to \( S \) and since \( \tilde{G}_S \) contains \( H' \), it follows that \( B \) contains \( H \).

**Lemma 11**

Let \( G < \text{Sym}(L) \) be a 2-group, where \( L = \{1, \ldots, n\} \) and \( n \) is even. Let \( \tilde{G} \) be a Sylow 2-subgroup of \( \text{Sym}(L^2) \) containing \( G' \), the isomorphic image of \( G \) in \( \text{Sym}(L^2) \). Let \( S \) be a partition of \( L \) into blocks of size 2, and let \( \tilde{D} \) be the restriction of \( \tilde{G}_S \) to \( S \). If \( H \) is the largest subgroup of \( G \) such that \( S \) is a minimal system of imprimitivity for \( H \), then \( \tilde{H} \), the homomorphic image of \( H \) in \( \text{Sym}(S) \) (induced by the natural action of \( H \) on the blocks in \( S \)), is a subgroup of \( \tilde{D} \).

**Proof** Let \( \pi \) be an element of \( H \), \( \pi' \) the corresponding element in \( H' \), and \( \tilde{\pi} \) the image of \( \pi \) in \( \tilde{H} \). Then \( \pi' \in \tilde{G}_S \), since \( S \) is a minimal system of imprimitivity for \( H \). Hence the restriction of \( \pi' \) to \( S \) is \( \tilde{\pi} \).

**Corollary 12 (Hoffmann)**

Assume the hypotheses of Lemma 11. Then \( B = C_2 \cap \tilde{D} \) is the largest 2-group in \( \text{Sym}(L) \) such that \( S \) is a minimal system of imprimitivity for \( B \) and \( \tilde{B} \), the homomorphic image of \( B \) in \( \text{Sym}(S) \) is \( \tilde{D} \). Moreover, \( B \) contains \( H \), i.e., \( H = B \cap G \).

**Proof** Let \( A < \text{Sym}(L) \) be a 2-group such that \( S \) is a minimal system of imprimitivity for \( A \) and such that \( \tilde{A} = \tilde{D} \). Since \( S \) is a minimal system of imprimitivity, the kernel \( K_A \) of the homomorphism \( A \to \tilde{A} \) is elementary abelian of order \( 2^r \), where \( r \leq m = \frac{n}{2} \). Let \( B = C_2 \cap \tilde{D} \). Then the kernel \( K_B \) of \( B \to \tilde{B} \) is elementary abelian of order \( 2^m \). Since the degree of \( A \) and \( B \) is \( n \), \( K_A < K_B \), hence \( A < B \), i.e., \( B \) is maximal. Finally,
let $\pi \in H$. By Lemma 11, the image $\overline{\pi}$ of $\pi$ in $\overline{H}$ is also in $\overline{D}$. Since $K_0$ has order $2^m$, therefore, $\pi \in H$. •

We explain how to determine $\tilde{G}$ from a complete imprimitivity structure for $G$. The construction is based on a correspondence between a complete imprimitivity structure of $G$ and a complete imprimitivity structure of $G'$.

**LEMMA 12 (Hoffmann)**

Let $G < \text{Sym}(L)$ be a 2-group, where $L = \{1, \ldots, n\}$ and $n$ is a power of 2, and let $F$ be a complete imprimitivity structure for $G$. Let $\Gamma$ be a set of imprimitivity in $F$ of equal size, not necessarily distinct but of cardinality greater than one. Assume that $\Gamma_1, \Gamma_2, \Gamma'_1, \Gamma'_2$ are (not necessarily nontrivial) sets of imprimitivity in $F$ such that $\Delta = \Gamma_1 + \Gamma_2$ and $\Delta' = \Gamma'_1 + \Gamma'_2$. If $G'$ is the isomorphic image of $G$ in $\text{Sym}(L^2)$, then the sets

$$
\Phi_a(\Delta, \Delta') = \{ \{x,y\} \mid (x \in \Gamma_1, y \in \Gamma_2) \text{ or } (x \in \Gamma_2, y \in \Gamma_1) \},
$$

$$
\Phi_b(\Delta, \Delta') = \{ \{x,y\} \mid (x \in \Gamma_1, y \in \Gamma'_2) \text{ or } (x \in \Gamma_2, y \in \Gamma'_1) \},
$$

and

$$\Psi(\Delta, \Delta') = \Phi_a(\Delta, \Delta') + \Phi_b(\Delta, \Delta')$$

are all sets of imprimitivity for $G'$.

**Proof**

Let $\pi \in G$ and assume that $\pi$ maps $\Delta$ and $\Delta'$ onto $\tilde{\Delta}$ and $\tilde{\Delta}'$, respectively. Then $\pi$ maps $\Psi(\Delta, \Delta')$ onto $\Psi(\tilde{\Delta}, \tilde{\Delta}')$. Furthermore, $\Phi_a(\Delta, \Delta')$ and $\Phi_b(\Delta, \Delta')$ are either mapped onto $\Phi_a(\tilde{\Delta}, \tilde{\Delta}')$ and $\Phi_b(\tilde{\Delta}, \tilde{\Delta}')$, respectively, or they are mapped onto $\Phi_b(\tilde{\Delta}, \tilde{\Delta}')$ and $\Phi_a(\tilde{\Delta}, \tilde{\Delta}')$, respectively. •

We remark without proof that the sets of imprimitivity defined by Lemma 12 are the only sets of imprimitivity for $G'$ if $G$ is a transitive Sylow 2-subgroup of a symmetric group.

Lemma 12 can be generalized to 2-groups of degree not a power of 2. Here one needs to consider separately the sets $\{x,y\}$ where $x$ and $y$ are in different trees in $F$. While the generalization is quite straightforward, it is cumbersome in formulation. Hence we will proceed differently and enlarge the permutation domain with additional points (left fixed by $G$) so that $G$ is represented isomorphically as a 2-group of degree a power of 2. Since at most $n-1$ additional points are needed, one verifies that this degree change does not adversely affect the asymptotic complexity of the algorithms to be described below. Note also that Lemma 11 and Corollary 12 remain intact.

Let $G$ be a 2-group of degree $2^m$, $\tilde{G}$ a Sylow 2-subgroup of $\text{Sym}(L^2)$ containing $G'$.

We assume that $G$ has the complete imprimitivity structure $F$, a complete binary tree
of height m. We describe an $O(n^2)$ method for constructing a complete imprimitivity
structure $\tilde{F}$ for $G'$, thereby determining $\tilde{G}$. It is not difficult to verify that $\tilde{F}$ consists
of binary trees with the heights $2m-2, 2m-3, \ldots, m-1$. Here the tree of height
$m-2+h, 1 \leq h \leq m$, has leaves labelled with pairs $\{x,y\}$ such that the nearest common
ancestor of $x$ and $y$ in $\tilde{F}$ is at distance $h$ from the leaves.

Consider the set $\Psi(\Delta_1, \Delta_2)$, where $\Delta_1$ and $\Delta_2$ are sets of imprimitivity in $F$ of equal
size. $\Delta_1$ is the leaf set of a subtree of $F$ rooted in the interior vertex $v_1$, and $\Delta_2$ of a
subtree rooted in $v_2$, hence we can represent $\Psi(\Delta_1, \Delta_2)$ by the pair $(v_1, v_2)$. Now

$$\Psi(\Delta_1, \Delta_2) = \Phi_a(\Delta_1, \Delta_2) + \Phi_b(\Delta_1, \Delta_2).$$

If $\Delta_1 = \Delta_{1,1} + \Delta_{1,2}$ and $\Delta_2 = \Delta_{2,1} + \Delta_{2,2}$, then

$$\Phi_a(\Delta_1, \Delta_2) = \Psi(\Delta_{1,1}, \Delta_{2,2}) + \Psi(\Delta_{1,2}, \Delta_{2,1})$$

and

$$\Phi_b(\Delta_1, \Delta_2) = \Psi(\Delta_{1,1}, \Delta_{2,1}) + \Psi(\Delta_{1,2}, \Delta_{2,2}).$$

So, if $v_{1,1}$ and $v_{1,2}$ are the sons of $v_1$ and $v_{2,1}$ and $v_{2,2}$ the sons of $v_2$, then we obtain the
following part of $\tilde{F}$ shown in Figure 5:

$$\begin{array}{c}
(v_{1,1}, v_{2,2}) & (v_{1,2}, v_{2,1}) & (v_{1,1}, v_{2,1}) & (v_{1,2}, v_{2,2}) \\
| & | & | & |
\downarrow & \downarrow & \downarrow & \downarrow \\
(v_{1,1}, v_2) & (v_{1,2}, v_2) & \end{array}$$

Figure 5

Here the intermediate vertices represent the sets $\Phi_a(\Delta_1, \Delta_2)$ and $\Phi_b(\Delta_1, \Delta_2)$ which could
have been labelled with the pairs of pairs $((v_{1,1}, v_{2,2}), (v_{1,2}, v_{2,1}))$ and
$((v_{1,1}, v_{2,1}), (v_{1,2}, v_{2,2}))$, respectively.

Figure 5 suggests a recursive procedure for constructing those subtrees of $\tilde{F}$
whose roots represent sets $\Psi(\Delta_1, \Delta_2)$, where $\Delta_1 + \Delta_2$ is again a set of imprimitivity in $F$.
We illustrate this in
**Example 5**

Let $G$ have degree $2^3$. We will construct the tree in $\tilde{F}$ of height $3-2+3 = 4$ which consists of all pairs $x,y$ such that the nearest common ancestor of $x$ and $y$ in $F$ is the root of $F$ (which is at distance 3 from the leaves). We assume a vertex labelling of $F$ as shown in Figure 6 below.

![Figure 6](image)

The root of the tree to be constructed is labelled $e,f$. The principle for constructing it recursively from this vertex pair is evident from Figure 7 below.

![Figure 7](image)

Now let $v$ be a vertex in $F$ at distance $h$ from the leaves, $v_1$ and $v_2$ its sons, and assume that $\Delta_1$ and $\Delta_2$ are the leaf sets of the subtrees rooted in $v_1$ and $v_2$, respectively. We have to explain how the set $\Psi(\Delta_1,\Delta_2)$ fits into larger sets of imprimitivity when $v$ is not the root. Here we observe that this set can be considered equivalent to the set $\Delta_1 + \Delta_2$ of imprimitivity for $G$, so, if we "equate" the vertex $v$ with the pair $(v_1,v_2)$ of its sons, then the tree in $\tilde{F}$ containing the subtree with root $(v_1,v_2)$ can be completed by attaching each of these subtrees to the vertex in $F$ with which the root pair has been equated. This simple idea is illustrated in
Example 6

Consider the group $G$ of Example 5. The tree in $\tilde{F}$ of height 3 is shown in Figure 8 below. Note that the root and its sons come from the tree $F$.

![Diagram](image)

Figure 8

The correctness of the construction is easily verified, and it is clear, moreover, that $\tilde{F}$ can be constructed in time proportional to its size, i.e., in $O(n^2)$ steps. Hence we obtain

**Proposition 5 (Hoffmann)**

Given a complete imprimitivity structure for the 2-group $G$ of degree $n = 2^m$, a complete imprimitivity structure for $G'$ can be found in $O(n^2)$ steps. Hence a representation for $G'$, a Sylow 2-subgroup of $\text{Sym}(L^2)$ containing $G'$, can be found in $O(n^3)$ steps.

We now describe how to obtain a representation of $\tilde{G}_S$ and how to determine from it a composition sequence for the group $\overline{G}$, the restriction of $\tilde{G}_S$ to $S$. It is crucial to construct a composition sequence for $\overline{G}$ directly since a composition sequence for $\tilde{G}_S$ might require $O(n^4)$ symbols.

As a first step, we label those leaves in the imprimitivity forest $\tilde{F}$ that are pairs in $S$. Clearly this can be done in $O(n^2 \log_2(n))$ steps. Next, we run the tree isomorphism algorithm separately for each tree in $\tilde{F}$. The object here is to label isomorphic subtrees of each tree $T$ and to rearrange those subtrees such that isomorphic subtrees are equal, i.e., an isomorphism exists which exchanges corresponding leaves in the
two subtrees. Since we deal with complete binary trees, it is clear that this part requires only $O(n^2)$ steps.

Having determined all isomorphic subtrees and having rearranged them appropriately, we now have an implicit representation for the groups $\tilde{G}_S$ and for $\tilde{D}$. In order to read out from this representation a composition series for $\tilde{D}$, we now have to augment the tree vertices with the following pointers and labels:

(a) Those leaves in $\tilde{F}$ which are pairs in $S$ are linked together into a list, from left to right.

(b) For each interior vertex $v$, a label is added to indicate whether the subtree rooted in $v$ contains a leaf belonging to $S$, and if so, a pointer to the leftmost such leaf in the subtree is also added.

Clearly this information can be gathered in $O(n^2)$ steps.

Let us denote the subtree rooted in the interior vertex $v$ with $T_v$, and the set of leaves in the subtree with $\Delta_v$. Let $v$ be an interior vertex of a tree in $\tilde{F}$ and assume that $u$ and $w$ are the sons of $v$. We associate with $v$ the subgroup $K(v)$ of $\tilde{G}_S$ which is the pointwise stabilizer of all points in $L^2$ which are not in $\Delta_v$. Then $K(v)$ setwise stabilizes the maximal sets of imprimitivity $\Delta_u$ and $\Delta_w$ iff $T_u$ and $T_w$ are not isomorphic. Hence, either $K(v) = K(u) \times K(w)$, and then $T_u$ and $T_w$ are not isomorphic, or $K(v) = \langle K(u) \times K(w), \pi \rangle$, and then $T_u$ and $T_w$ are isomorphic and $\pi$ exchanges the corresponding leaves of $T_u$ and $T_w$. Moreover, in that case the index of $K(u) \times K(w)$ in $K(v)$ is 2.

Now let $\Sigma_u$ and $\Sigma_w$ be composition sequences for $K(u)$ and $K(w)$, respectively. If $T_u$ and $T_w$ are not isomorphic, then their concatenation $\Sigma_u, \Sigma_w$ is a composition sequence for $K(v)$. Otherwise, $\Sigma_u, \Sigma_w, [\pi]$ is a composition sequence for $K(v)$. Hence a composition sequence for $\tilde{G}_S$ can be determined by a simple post-order traversal of the trees in $\tilde{F}$.

The determination of a composition sequence for $\tilde{D}$ is quite similar. Here we must construct only those generators $\pi$ which exchange the leaf sets $\Delta_u$ and $\Delta_w$ containing at least one leaf which is a pair in $S$. Since the corresponding interior vertices are marked, this situation is readily recognized. So we must modify the procedure described above by adjoining to the concatenated sequence $\Sigma_u, \Sigma_w$ the generator $\pi$ iff $T_u$ and $T_w$ are isomorphic and both $u$ and $w$ are marked vertices. Furthermore, $\pi$ is in
The computation of a composition sequence for $\overline{D}$ can be implemented in $O(n^2)$ steps: We note that we can construct $\overline{\pi}$ directly with the help of the pointers at the sons of the vertex considered and the pointers linking all leaves which are pairs in $S$. Now consider computing the composition sequence for $\overline{D}$. If $v$ is marked and has the isomorphic sons $u$ and $w$ (which are therefore also marked), then $\Delta_u$ and $\Delta_v$ contain an equal number of leaves which are pairs in $S$. There are $n$ such leaves total. Hence the number of marked vertices with isomorphic sons is at most $\frac{n}{2} + \frac{n}{4} + \cdots + \frac{n}{2^m} < n$. Therefore, no more than $n$ generators are constructed, each in at most $O(n)$ steps. Thus, a total of $O(n^2)$ steps suffice to construct a composition sequence for $\overline{D}$.

We now give the final steps for computing $H$, the largest subgroup of the given 2-group $G$ with the prescribed minimal system $S$ of imprimitivity.

Having shown how to obtain a composition sequence for $\overline{D}$, we must now determine a composition sequence for $B$, the largest 2-group in $\text{Sym}(L)$ with $S$ as a system of imprimitivity and homomorphic image $\overline{D}$ in $\text{Sym}(S')$. Let $S = \{(x_1,y_1), \ldots, (x_r,y_r)\}$ and let $\Sigma$ be a composition sequence for $\overline{D}$. (We now consider the elements in $\Sigma$ equivalently as permutations in $\text{Sym}(L)$). This is justifiable since $S$ is a partition. We claim that $\Lambda = [(x_1,y_1), \ldots, (x_r,y_r)]$, $\Sigma$ is a composition sequence and generates $B$.

It is clear that $\Lambda$ is a composition sequence. Let $C$ be the group generated by $\Lambda$. Surely the homomorphic image $\overline{C}$ is $\overline{D}$. Now the kernel of the homomorphism from $C$ to $\overline{C}$ is an elementary abelian 2-group of order $2^r$, where $r = \frac{n}{2}$, hence $B = C$ is the desired group.

By Proposition 2, we can find a complete imprimitivity structure for $B$ in $O(n^2 \cdot \log_2(n) \cdot \log_2^*(n))$ steps. Hence, by Corollary 9, the composition sequence for $B$ can be converted into an imprimitivity sequence for $B$ in $O(n^3)$ steps. Summarizing the results of the previous sections, we have now established

**Theorem 6**

Let $G < \text{Sym}(L)$ be a 2-group of even degree for which a complete imprimitivity structure and an imprimitivity sequence is given. Let $S$ be a partition of $L$ into blocks of size 2, and let $H$ be the largest subgroup of $G$ such that $S$ is a minimal system of imprimitivity for $H$. Then an imprimitivity sequence for $H$ can be found in $O(n^3)$ steps.
We now turn to the interpretation of the imprimitivity problem as an automorphism problem for certain graphs. For the remainder of the section, cone graph shall always mean binary cone graph. The leaf set of the cone graph X is the set of leaves of the underlying tree. The subgraph consisting of the leaf set and the cross edges will be called the leaf graph of X.

We reconsider Problem 2, the determination of Autv(X) of the cone graph X. Since we are ultimately interested in the application of this problem to trivalent graph isomorphism, we will only consider cone graphs whose leaf graphs have a particular structure.

We observe that the group of automorphisms of the (regular, binary) cone graph X which fix the root can be faithfully represented as a permutation group acting on the leaf set. Consequently, we may consider the problem of determining this group equivalently as the problem of intersecting the automorphism group of the leaf graph with the Sylow 2-subgroup of the symmetric group of the leaf set whose imprimitivity structure is the underlying tree. We expect that the difficulty of the automorphism problem is related to the graph-theoretic structure (and the consequent structure of the automorphism group) of the leaf graph.

Let us consider the automorphism problem for cone graphs in which the leaf graph has valence one, i.e., the leaf graph consists of isolated edges and isolated vertices. Such cone graphs are called simple since they have a simple graph-theoretic structure.

A set of two vertices can be considered an undirected edge, and we note that the imprimitivity problem is essentially the automorphism problem for a simple cone graph whose leaf graph does not have any isolated vertices. This suggests approaching the automorphism problem for simple cone graphs as follows:

Let X be a simple cone graph with leaf set L and underlying tree T, and let n be the number of leaves of T. Let G be the Sylow 2-subgroup of Sym(L) induced by the automorphisms of T. It is clear that we can compute an imprimitivity sequence for G in $O(n^2)$ steps. Let $J \subseteq L$ be the set of all leaves of X incident to a cross edge, $J' \subseteq L$ the set of all isolated vertices in the leaf graph. We compute $G_J$, the setwise stabilizer of J in G, using Proposition 4, and note that the sought automorphism group is a subgroup of $G_J$. We restrict $G_J$ to J obtaining a group A. Now the cross edges of X define a partition of J into blocks of size 2. We call this partition $S$ and solve the imprimitivity problem for $S$ and A using Theorem 6. Let H be the subgroup of A so obtained, and let
B be the restriction of \( \mathbf{G} \) to the set \( J' \). Then \( \mathbf{G} \cap \mathbf{H} \times \mathbf{B} \) is the desired automorphism group of the simple cone graph. Note that all groups are 2-groups for which we have (or can compute efficiently) imprimitivity sequences. In summary, we therefore have

**Proposition 6 (Hoffmann)**

Let \( \mathbf{X} \) be a simple cone graph with \( n \) vertices. Then an imprimitivity sequence (with respect to the imprimitivity structure defined by the underlying tree) for \( \text{Aut}_v(\mathbf{X}) \) can be determined in \( O(n^3) \) steps.

We remark without proof that the bound of Proposition 6 remains intact when the cross edges of \( \mathbf{X} \) are labelled.

### 4.3. Gadgets for Trivalent Graph Isomorphism

In order to apply Theorem 6 to the problem of determining \( \text{Aut}_e(\mathbf{X}) \), \( \mathbf{X} \) a connected trivalent graph, we need to devise a mechanism for reducing the problem of finding, in \( \text{Aut}_e(\mathbf{X}_k) \), the subgroup \( \mathbf{B} \) of all automorphisms which may be extended to an automorphism in \( \text{Aut}_e(\mathbf{X}_{k+1}) \), to determining \( \text{Aut}_v(\mathbf{Y}) \), where \( \mathbf{Y} \) is a simple cone graph. Roughly speaking, we will use a construction by which certain subgraphs are substituted which permit viewing the trivalent graph as a simple cone graph. This intuitive notion is imprecise in the sense that instead of subgraphs we really substitute certain 2-groups for points on which \( \text{Aut}_e(\mathbf{X}_k) \) acts (this is done via wreath products), with the effect that the determination of the subgroup \( \mathbf{B} \) reduces to a sequence of applications of Theorem 6, plus a few setwise stabilization and intersection operation.

To grasp the thrust of this approach, let us first consider cone graphs whose leaf graphs have valence 2. We plan to reduce this case to simple cone graphs by using isomorphism gadgets in an abstract sense. Let \( \mathbf{X} \) be a cone graph whose leaf graph has valence 2. We wish to construct a simple cone graph \( \mathbf{X}_2 \) from \( \mathbf{X} \) which has, roughly speaking, the same automorphism group as \( \mathbf{X} \).

Let \( T \) be the underlying tree of \( \mathbf{X} \) of height \( m \). The graph \( \mathbf{X}_2 \) will have an underlying tree \( T_2 \) of height \( m+1 \). Hence \( \mathbf{X}_2 \) has (about) twice as many vertices as \( \mathbf{X} \). We put the vertices \( v \) of \( T \) into 1-1 correspondence with the interior vertices \( v' \) of \( T_2 \) such that the tree structure is preserved. Now consider the cross edge \((u,v)\) of \( \mathbf{X} \), and let \( u_1, u_2 \) and \( v_1, v_2 \) be the sons of \( u' \) and \( v' \), respectively. Corresponding to the edge \((u,v)\), there will be exactly one cross edge \((u_i,v_j)\) in \( \mathbf{X}_2 \). Here the indices \( i \) and \( j \) must be chosen
such that $X_2$ has a leaf graph of valence 1. Since the leaf graph of $X$ has valence 2, this is always possible. Let us call the graph $X_2$ an extended simple cone graph of $X$. Note that $X$ does not uniquely determine $X_2$.

**Example 7**

Let $X$ be the cone graph shown in Figure 9 below. Its leaf graph has valence 2. Figure 10 shows an extended simple cone graph of $X$ obtained by the construction described above. Note that this graph has a leaf graph of valence 1.

The construction of the extended simple cone graph is a gadget construction in the following sense: Each valence 2 leaf $v$ in the leaf graph of $X$ has been replaced with the pair $v_1$, $v_2$, and the two edges incident to $v$ are connected (one each) to these new vertices. Now an automorphism of $X$ may locally interchange the two edges.
emanating from $v$. Hence the gadget $v_1, v_2$ must permit the transposition $(v_1, v_2)$. This is done by a wreath extension with $C_2$ of the underlying automorphism group of the tree as realized by the increase in the tree height.

The crucial property of the construction is summarized by the following

**Theorem 7 (Hoffmann)**

Let $X$ be a regular, binary cone graph with root $v$ whose leaf graph has valence 2, and let $X_2$ be an extended simple cone graph of $X$ constructed as described above. Then $\text{Aut}_v(X)$, the group of all automorphisms of $X$ fixing the root $v$, is the homomorphic image of $\text{Aut}_v(X_2)$, where the homomorphism is provided by the induced action of $\text{Aut}_v(X_2)$ on the interior vertices of $X_2$.

**Proof** In the following, we will identify the vertices of $X$ with the interior vertices of $X_2$. Hence the leaf set $L$ of $X$ may be considered the set of interior vertices of $X_2$ whose distance from the leaves of $X_2$ is 1. $L_2$ will be the leaf set of $X_2$.

Let $\alpha \in \text{Aut}_v(X_2)$, $\alpha$ its homorphic image. Since the brothers in $L_2$ form sets of imprimitivity for $\text{Aut}_v(X_2)$, it follows that $\alpha \in \text{Aut}(X)$.

Conversely, let $\alpha \in \text{Aut}(X)$, $\alpha$ an extension of $\alpha$ to the vertex set of $X_2$. Consider $w \in L$ and the corresponding brothers $w_1, w_2 \in L_2$. Let $(u, w)$ be a cross edge of $X$, $(u_1, w_1)$ the corresponding cross edge in $X_2$. If $\alpha$ maps $(u_1, w_1)$ to $(u_2, w_2)$, then $\alpha' = \alpha(w_1, w_2)$ maps the edge to $(u_1, w_1)$. Hence there exists a permutation $\pi$ consisting entirely of transpositions of brothers in $L_2$, such that $\beta = \alpha \pi$ is an automorphism of $X_2$. Since $\beta = \alpha$, it follows that to every $\alpha \in \text{Aut}_v(X)$ there exists an extension in $\text{Aut}_v(X_2)$.

We have reduced the automorphism problem for cone graphs whose leaf graphs have valence 2, to the automorphism problem of simple cone graphs. It remains to show how to obtain an imprimitivity sequence for $\text{Aut}_v(X)$ from an imprimitivity sequence for $\text{Aut}_v(X_2)$. Now it is trivial to compute the homomorphic image $\Sigma'$ of an imprimitivity sequence $\Sigma$ for $\text{Aut}_v(X_2)$. However, $\Sigma'$ may contain redundant generators. Hence we need the following

**Lemma 13**

Let $[\pi_m, ..., \pi_1]$ be a sequence of generators for the 2-group $G$ such that, for $1 \leq i \leq m$, the group $G^{(i)} = <\pi_m, ..., \pi_{i+1}>$ has index 1 or 2 in $G^{(i-1)}$. Then an imprimitivity sequence for $G$ can be determined in $O(m \cdot n^2)$ steps.

**Proof** By Proposition 2, we can find a complete imprimitivity structure for $G$ in less than $O(m \cdot n^2)$ steps. We will sift the generators in the sequence $\pi_m, ..., \pi_1$, as for
Proposition 3, but without forming or sifting pair products. We claim that the resulting table defines an imprimitivity sequence for $G$.

Assume inductively that the table has $r \leq m-i$ nonempty entries after having sifted $\pi_m, ..., \pi_{i+1}$, and that these entries constitute an imprimitivity sequence for $G^{(i)}$. Consider sifting $\pi_i$. If $\pi_i$ is redundant, then no new table entry results, the index of $G^{(i)}$ in $G^{(i-1)}$ is 1, and we now have an imprimitivity sequence for $G^{(i-1)}$. Otherwise, the new entry $\psi$ is made. Now the index of $G^{(i)}$ in $<G^{(i)}, \psi> = G^{(i-1)}$ is at least 2, hence it must be exactly 2 by the hypothesis of the lemma. Hence every pair product formed with $\psi$ and with existing table entries must be redundant, and so the new table defines an imprimitivity sequence for $G^{(i-1)}$.

As a consequence of Lemma 13 and Theorem 7, we obtain immediately

**Corollary 13**

Let $X$ be a regular, binary cone graph with root $v$ whose leaf graph has valence 2. If $X$ has $n$ leaves, then an imprimitivity sequence for $\text{Aut}_r(X)$ can be determined in $O(n^3)$ steps.

We remark that the corollary generalizes to cone graphs whose leaf graphs have valence 2 and whose cross edges are labelled.

Recall that the automorphism problem for cone graphs may be understood as an intersection problem of the automorphism group of the leaf graph with the 2-group $G$ arising from the automorphism group of the underlying tree. We observe that the solutions in Corollary 13 and Proposition 6 do not make use of the Sylow structure of $G$, hence we may speak of "abstract" cone graphs, that is, leaf graphs augmented by an underlying 2-group $G$ instead of the underlying complete binary tree. Thus we may generalize these results as follows:

**Corollary 14 (Hoffmann)**

Let $X = (V,E)$ be an edge labelled graph of valence 2, and let $G < \text{Sym}(V)$ be a 2-group with given imprimitivity sequence. Then an imprimitivity sequence for $\text{Aut}(X) \cap G$ may be determined in $O(n^3)$ steps, where $n$ is the number of vertices of $X$.

Reflecting on the gadget construction above, we also observe that there is really no need to replace a vertex $v$ with a pair $v_1, v_2$ if the valence of $v$ (in the leaf graph) is less than 2. However, we now must setwise stabilize in $G$ the set of all vertices of valence 0 and the set of all vertices of valence 1. The corresponding extension of the group $G$ to the new permutation domain is straightforward.
We now have developed all the essential ideas and the major tools needed for an $O(n^4)$ isomorphism test for trivalent graphs. This test follows the general approach of the other two isomorphism tests in this chapter but differs significantly in the details. In particular, note that it is not necessary to determine $\text{Aut}_e(X)$ for the purpose of testing isomorphism: If $h$ is the maximum distance of any vertex from the edge $e$, then knowing the homomorphic image $A_e(X)$ of $\text{Aut}_e(X)$ in $\text{Sym}(V_h)$ is sufficient. ($A_e(X)$ is the restriction of $\text{Aut}_e(X)$ to the set of vertices at distance $h$ from $e$). Consequently, we will determine the groups $A_e(X_k)$, where, for $0 \leq k \leq h$, $A_e(X_k)$ is the homomorphic image of $\text{Aut}_e(X_k)$ in $\text{Sym}(V_k)$, and $A_e(X_{h+1}) = A_e(X)$.

Recall Sections 1 and 2. We develop an algorithm for determining the group $B$ of all those permutations in $A_e(X_k)$ which can be extended to an automorphism in $\text{Aut}_e(X_k+1)$. More precisely, given a complete imprimitivity structure and an imprimitivity sequence for $A_e(X_k)$, we will determine an imprimitivity sequence for $B$ in $O(n_k^2)$ steps. Here $n_k$ is the cardinality of $V_k$, the set of all vertices at distance $k$ from $e$.

For every edge type, we examine the subgraph spanned by these edges and study how it contributes to the definition of $B$. We plan to compute, for each edge type $t$, the subgroup $B(t)$ consisting of those permutations in $A_e(X_k)$ which could be extended to an automorphism in $\text{Aut}_e(X_k+1)$, provided $E_k$ contained only edges of type $t$. Having so determined up to seven subgroups of $A_e(X_k)$, each arising from a particular edge type, we obtain the group $B$ as the intersection of these groups. Note that the correctness of this strategy follows from Lemmata 1 and 2.

*Type* $t_{0,2}$: The subgraph spanned by these edges has valence 2. Hence the group $B^{(0,2)}$ can be determined in $O(n_k^2)$ steps using Corollary 14.

*Type* $t_{1,1}$: The subset of the vertices in $V_k$ incident to an edge of this type has to be stabilized in $A_e(V_k)$. By Proposition 4, $B^{(1,1)}$ is found in $O(n_k^2)$ steps.

*Type* $t_{2,1}$: Proceed as for type $t_{1,1}$.

*Type* $t_{1,2}$: Let $v_1, v_2$ be the two vertices in $V_k$ incident to a family of this type. We may replace the family by the cross edge $(v_1, v_2)$. (Note that these new cross edges are distinguished from the "normal" cross edges of type $t_{0,2}$ since we consider a single edge type at a time). The subgraph so obtained has valence 2, hence $B^{(1,2)}$ can be determined using Corollary 14.

*Type* $t_{2,2}$: We proceed as for type $t_{1,2}$ and replace each family by a single cross edge. Note that these cross edges are isolated. Hence we can apply Corollary 14.
**Type t_{2,3}:** We replace each family of this type by the three cross edges \((v_1,v_2), (v_2,v_3),\) and \((v_3,v_1).\) Here \(v_1, v_2, \) and \(v_3\) are the three vertices in \(V_k\) incident to the family. Note that the subgraph spanned by these new cross edges consists of isolated cycles of length 3, hence is a graph of valence 2. Corollary 14 applies.

We have now considered all edge types except for type \(t_{1,3}.\) This case is more complicated and we consider it now in detail. First, we replace each family of type \(t_{1,3}\) with a cycle of three special cross edges, just as for type \(t_{2,3}.\) Since the subgraph spanned by these cross edges may have valence 4, however, we need to further analyze its structure.

Consider a cycle with vertices \(u, v,\) and \(w\) obtained from a family of edges of type \(t_{1,3}.\) Assuming a suitable extension of the permutation domain of \(A_e(X_k),\) the imprimitivity structure of this group is a complete binary tree. Since the cycle \([u,v,w,u]\) has odd length, it follows that the height of the nearest common ancestor (in the imprimitivity tree) of one of the three pairs \((u,v),\) \((u,w),\) and \((v,w)\) must differ from the height of the nearest common ancestor for the other two pairs (see Figure 11). Consequently, one of the three cycle edges is distinguished from the other two. We call the

![Figure 11](image)

distinguished edge the \(b\)-edge of the cycle, and call the other two edges \(a\)-edges. It is clear that an automorphism cannot map a \(b\)-edge to an \(a\)-edge, hence \(B(1,3) = B(1,3-a) \cap B(1,3-b),\) where \(B(1,3-a)\) is the subgroup of \(A_e(X_k)\) arising by only considering \(a\)-edges, and \(B(1,3-b)\) the subgroup arising by only considering \(b\)-edges.

Considering the \(a\)-edges obtained from families of type \(t_{1,3},\) we note that these edges can be paired since each family results in exactly two \(a\)-edges in our
construction. Furthermore, the a-edges can be oriented, from the vertex incident to both edges of a pair towards the b-edge of the same family. These two observations are crucial for analyzing the local action of $B^{(1,3-a)}$ on a-edges incident to a common vertex, and are used in the design of suitable isomorphism gadgets coping with valences higher than 2.

Subject to the pairing and orientation of the a-edges, the group $B^{(1,3-a)}$ is equal to $B^{(1,3)}$, since the mapping of a directed a-edge pair entails the correct mapping of the b-edge belonging to the same family. This observation saves constructing $B^{(1,3-b)}$ and one group intersection.

We now describe how to obtain $B^{(1,3-a)}$ (assuming that the a-edges have been paired and oriented). First, we reduce the essential valence to 1 by observing the edge pairing, then we account fully for the orientation of the a-edges.

In the subgraph spanned by the a-edges, a vertex can have valence up to four. So we classify the vertices according to their valence and setwise stabilize each of the five classes in $A_4(X_k)$, thereby obtaining the 2-group $D$.

We wish to extend $D$ to a larger 2-group acting on a larger permutation domain by forming wreath products with various 2-groups. These 2-groups should be the gadgets reducing the valence of the subgraph spanned by the a-edges to one.

For vertices of valence 2, the wreath product is with the cyclic group $C_2$ of order 2. This is exactly as for Corollary 13.

Consider the vertices of valence 3. Due to the implicitly present b-edge, a vertex of valence 3 is always incident to a pair and a single a-edge. Because of the pairing, the single a-edge cannot be mapped to one of the pair edges, hence our gadget is a Sylow 2-subgroup of $S_3$, the symmetric group of degree 3, as shown graphically in Figure 12.
Finally, consider the valence 4 vertices. Here the vertex must be incident to two pairs of $a$-edges. It is clear that each pair is a set of imprimitivity for the action of $B^{(1,3-a)}$ on the $a$-edges, hence the required gadget is a Sylow 2-subgroup of $S_4$ as shown in Figure 13.

The Valence 4 Gadget

With this gadget construction we obtain a new 2-group $D'$ of degree at most 4 times the degree of $D$. $D'$ reduces the essential valence of the subgraph of $a$-edges to one, but we still need to fully account for the orientation of the $a$-edges. We do this now by setwise stabilizing, in the group $D'$, the class of all vertices incident to the origin of an $a$-edge. To the resulting 2-group $D''$ we may apply Corollary 14 and so obtain $B^{(1,3-a)} = B^{(1,3)}$. It is clear that the application of the corollary dominates the timing, and in summary we have now established

**Theorem 8 (Hoffmann)**

Given a complete imprimitivity structure and an imprimitivity sequence for $A_e(X_k)$, an imprimitivity sequence for the subgroup $B$, consisting of those permutations which may be extended to an automorphism in $\text{Aut}_e(X_{k+1})$, can be determined in $O(n_k^3)$ steps, where $n_k$ is the number of vertices at distance $k$ from the edge $e$.

We now construct $A_e(X_{k+1})$ from $B < A_e(X_k)$ and $E_k$ in two stages: First, $B$ is lifted to a homomorphic image $\overline{B}$ in $\text{Sym}(V_{k+1})$. By Lemmata 1 and 2 and the definition of $B$ this is always possible. Second, we adjoin generators for the group $C$ consisting of those permutations of $V_{k+1}$ which may be extended to an automorphism of $X_{k+1}$ fixing every vertex not in $V_{k+1}$.

By Corollary 2, the group $C$ is generated by transpositions $(u,v)$, where $u$ and $v$ are vertices in $V_{k+1}$ incident to a family of type $t_{2,3}$, $t_{2,2}$, or $t_{2,1}$. Clearly $C$ is elementary abelian. An imprimitivity sequence $[(u_1,v_1), ..., (u_r,v_r)]$ for $C$ can be found in
Let \([\pi_1, \ldots, \pi_s]\) be the imprimitivity sequence determined for \(B\). We construct a sequence \([\bar{\pi}_1, \ldots, \bar{\pi}_s]\) of permutations generating \(B\), where \(\bar{\pi}_i\) is obtained from \(\pi_i\), \(1 \leq i \leq s\). We assume a fixed but arbitrary enumeration of the vertices in \(V_{k+1}\) and explain how to determine the \(\bar{\pi}_i\).

Let \(\pi\) be the next generator of \(B\). For each \(v \in V_{k+1}\) determine its ancestry \(U\) in \(V_k\), and let \(U'\) be the image of \(U\) under \(\pi\). By Lemma 1, \(U'\) is an ancestry and the family types for \(U\) and \(U'\) are equal. If \(w \in V_{k+1}\) has the same ancestry \(U\) as \(v\), then there are exactly two vertices \(v', w' \in V_{k+1}\) whose ancestry is \(U'\). In this case we set \(v^\pi = v'\) and \(w^\pi = w'\), assuming, without loss of generality, that \(v\) precedes \(w\) and \(v'\) precedes \(w'\) in the enumeration of \(V_{k+1}\). If there is no other vertex \(w \in V_{k+1}\) whose ancestry is \(U\), then there is exactly one \(v' \in V_{k+1}\) with ancestry \(U'\). In this case we define \(v^\pi = v'\).

It is clear that this construction determines a permutation \(\bar{\pi} \in \text{Sym}(V_{k+1})\) which is unique due to the fixed enumeration of \(V_{k+1}\). Moreover, given \(\pi\) there is an automorphism in \(\text{Aut}_e(X_{k+1})\) whose restriction to \(V_{k+1}\) is \(\bar{\pi}\). Hence, if \(\bar{B} = \langle \bar{\pi}_1, \ldots, \bar{\pi}_s \rangle\), then \([\bar{\pi}_1, \ldots, \bar{\pi}_s]\) contains a composition sequence for \(\bar{B}\). Note that \([\bar{\pi}_1, \ldots, \bar{\pi}_s]\) can be determined in \(O(n_k^2 + n_{k+1}^2)\) steps.

Finally, the sequence \([(u_1, v_1), \ldots, (u_r, v_r), \bar{\pi}_1, \ldots, \bar{\pi}_s]\) contains a composition sequence for \(A_e(X_{k+1})\). Note that \(r + s \leq n_k + n_{k+1}\). Hence, by Proposition 2 and Lemma 13, we can determine a complete imprimitivity structure and an imprimitivity sequence for \(A_e(X_{k+1})\) in \(O(n_k^3 + n_{k+1}^3)\) steps. Consequently, we have established

**Theorem 9 (Hoffmann)**

Given a complete imprimitivity structure and an imprimitivity sequence for \(A_e(X_k)\), then a complete imprimitivity structure and an imprimitivity sequence for \(A_e(X_{k+1})\) can be determined in \(O(n_k^3 + n_{k+1}^3)\) steps, where \(n_k\) is the cardinality of \(V_k\).

**Proof** Theorem 8 and the above discussion. 

We summarize the results derived above as follows:

**Corollary 15**

Let \(X\) be a connected, trivalent graph with \(n\) vertices, \(e\) an edge of \(X\), and \(h\) the maximum distance in \(X\) of any vertex from \(e\). Then \(A_e(X)\) can be determined in \(O(n^3)\) steps.

**Proof** This follows from a straightforward induction on \(h\) using Theorem 8.
**COROLLARY 16 (Hoffmann)**

Let $X$ be a trivalent graph with $n$ vertices. Then isomorphism of $X$ can be tested in $O(n^4)$ steps.

**Proof** $X$ is split into connected components which are classified into isomorphism classes. An elementary computation establishes the time bound with the help of Corollary 15. $
$ 

5. **Notes and References**

Miller [1979] has investigated the problem of reducing the valence of a graph while preserving isomorphism. He defines an isomorphism $m$-gadget as a graph $Z$ of valence at most $m$ with a distinguished set $\Gamma$ of $m+1$ vertices of valence at most $m-1$. $Z$ has to have the additional property that the setwise stabilizer of $\Gamma$ in $\text{Aut}(Z)$ acts on $\Gamma$ as the symmetric group $\text{Sym}(\Gamma)$. Given a graph of valence $m+1$, one first replaces every vertex $v$ of valence $m+1$ with a copy of $Z$, connecting the $m+1$ edges formerly incident to $v$ to the vertices in $\Gamma$. Then, the gadget edges are labelled by adding tails of sufficient length to each. It is not hard to see that two different ways of replacing every vertex of valence $m+1$ must result in two isomorphic graphs, i.e., the construction must preserve isomorphism. Miller [1979] proved that the only possible gadget is the 4-gadget discovered by Carter [1977].

Theorem 1 is due to Tutte [1947]. Tutte's formulation states that the order of the stabilizer of every vertex in a trivalent graph is either $2^r$ or $3 \cdot 2^r$. A variant has been proved in Babai and Lovász [1973]. The basic approach to the trivalent case is due to Furst, Hopcroft, and Luks [1980a], except for the reduction to the setwise stabilizer in 2-groups by a classification of the edges in $E_k$, which is due to Luks [1980]. Furst, Hopcroft, and Luks [1980a] contains sufficient techniques to derive the equivalent of Corollary 4. This was noted independently by us and by M. Furst.

The methods of Section 3 are due to Luks [1980]. Recall that we partition the set $W$ consisting of all nonempty subsets of $V_k$ of size at most 3 into up to 9 blocks. Luks [1980] uses only 8 blocks. This seeming discrepancy is explained by the fact that for $i > 0$ we may consider $t_{i,1}$, $t_{i,2}$, and $t_{i,3}$ to be the same label, since subsets of $V_k$ of different cardinality must lie in different orbits of $G$. Thus, only six blocks are required.
The material of Section 4 is due to Hoffmann [1981b]. Propositions 1 and 2 are the result of joint work with C. Sims. Lemmata 8 and 9 are hinted at in Luks [1980]. The isomorphism test takes the same general approach as the subexponential algorithm of Furst, Hopcroft, and Luks [1980a]. The previous best bound for trivalent graph isomorphism testing seems to have been $O(n^6)$ as announced in Luks [1980].