New Computational Methods for Analysing Finitely-Presented Groups

Georgina Liversidge

A thesis submitted in fulfillment of the requirements for the degree of Doctor of Philosophy in Mathematics, the University of Auckland, 2023.

Supervised by Marston Conder
Co-Supervised by Gabriel Verret

Department of Mathematics
University of Auckland
New Zealand

Acknowledgements:

Words cannot express my gratitude to my supervisor Prof. Marston Conder for his astounding knowledge, his inspirational expertise and his well-tested patience. I'm extremely grateful to my co-supervisor Dr Gabriel Verret and to my advisory committee members Assoc. Prof. Jeroen Schillewaert and Dr Tanya Evans for all their guidance, encouragement and friendship. I thank also the University of Auckland for providing financial support in the form of a scholarship, without which this endeavour would not have been possible.

I would like to extend my sincere thanks to Prof. George Havas, Prof. Maxim Vsemirnov and Dr Brent Everitt for their correspondence, advice and expertise. Special thanks to Prof. Vivien Kirk for her moral support and practical advice during particularly trying personal times. I am also grateful to my fellow PhD candidates, Dr Emily Kendall and Dr Samuel Dobson, for their friendship and support.

Finally, I'd like to acknowledge my mum, Linda Liversidge, and my aunt, Karen Roper, for their astoundingly unconditional love and support.

Abstract:

In this thesis, we describe three new and/or improved methods for the analysis of finitely-presented groups and show their usefulness in a variety of contexts.

The first procedure creates a labelled coset graph which can be used for rewriting and also for finding expressions for subgroup elements in terms of given generators. We use this procedure to find nice generating sets for torsion-free subgroups of finite index in ordinary triangle groups, with implications for the study of regular maps and automorphism groups of compact Riemann surfaces.

Our second procedure is an improvement of the package `PEACE` by Havas and Ramsay. This package uses coset enumeration to find a proof for subgroup inclusion, by way of a proof word (which is a sequence of the elements of the supergroup and various brackets indicating two methods of simplification, the equality of which proves the inclusion). We use this procedure to find new generating sets and presentations for the special linear group $\mathrm{SL}(3, \mathbb{Z})$.

Finally, we give a version of the low-index subgroups algorithm with added capabilities for finding specific types of subgroups by way of avoiding the inclusion of specified words. We use this algorithm to find torsion-free subgroups of Coxeter groups, with implications for the construction of hyperbolic manifolds.

Contents

Chapter 1

Introduction

The overall goal of this PhD thesis project was to develop new and/or improved methods for dealing with finitely-presented groups. We aimed to develop methods that may be helpful in a variety of contexts and showcase some applications.

We succeeded in developing procedures in the MAGMA programming language developed by Bosma, Cannon and Playoust [5], with applications in three different contexts: the construction and analysis of regular maps, hyperbolic manifolds of small volume, and new presentations and generating sets for special linear groups over the integers. While these areas are quite different, they are tied together not only by the role of finitely-presented groups but also by some common themes such as torsion-free subgroups and finding alternative generating sets.

A finitely-presented group $G = \langle X \mid R \rangle$ is a group generated by a finite set X subject to a finite set R of relators, which are words on the

elements of $X \cup X^{-1}$, all equal to the identity element of G, and are enough to define G. Finite presentations give a succinct way to specify a group, but many properties of the group can be difficult to determine from a presentation. One such property is the finiteness of the group. Even the trivial group can be defined by arbitrarily long presentations, and the proof of the infamous 'Word Problem' attributed to Novikov [35] and Boone [4] states that there is no algorithm that can determine if a given word on the generators in an arbitrary finitely-presented group is equal to the identity element.

Nevertheless, there are several methods that can be effective for investigating the structure of finitely-presented groups, such as the Reidemeister-Schreier rewriting process, Tietze transformations, coset enumeration and the Low Index Subgroups algorithm (see [24]).

In this thesis project, we aimed to develop methods which may be helpful in instances where such methods do not exist, or tend to fail, or require substantial computing resources (memory and/or time).

Many methods, including those we have developed, apply to finite-index subgroups of finitely-presented groups. The existence of certain subgroups can tell us many things. For example, the presence of a finite-index finite subgroup can help determine the order of a group through Lagrange's theorem, and the presence of a subgroup with infinite abelianisation proves that the group itself is infinite. All of the procedures we have developed rely on coset enumeration in some way

or another, and hence we consider only finite-index subgroups.

Our first procedure, which we have called `CosGraphLabs`, uses the theory behind the Reidemeister-Schreier rewriting process and Tietze transformations to create a labelled coset graph which can be used for rewriting, and also for finding expressions for subgroup elements in terms of given generators. Our testing shows that `CosGraphLabs` outperforms the functions called `!` and `Rewrite` already available in Magma in terms of processing time.

We demonstrate the use of `CosGraphLabs` in finding 'nice' generating sets for torsion-free normal subgroups of ordinary triangle groups, and expressions for the conjugates of those generators by the generators of the parent group. This extends (to the best of the author's knowledge, for the first time) the work of Leech in the 1960s [26] to other triangle groups, and has implications for the study of regular maps and large automorphism groups of compact Riemann surfaces. Having such 'nice' generating sets is very useful for constructing covers of such maps with abelian covering groups, for example.

The full code and results of the latter application can be found at `github.com/GLiversidge/CosGraphLabs`.

Next, `PEACE` is a package developed by Havas and Ramsay [22, 23], which we have rewritten with added functionality and usability and called `PEACEv2`. `PEACE` and `PEACEv2` use coset enumeration to give 'proof words' for element inclusion in subgroups. Unlike `CosGraphLabs`,

`PEACEv2` can be used with index 1 subgroups and thus can be used to confirm alternative generating sets for a group. To the best of the author's knowledge there are no other packages which produce proofs for subgroup inclusion.

We demonstrate the use of `PEACEv2` in finding new generating pairs for the special linear group $\mathrm{SL}(3,\mathbb{Z})$ and the corresponding new presentations. These presentations are, to the best of the author's knowledge, the first known two-generator presentations for $\mathrm{SL}(3,\mathbb{Z})$. Furthermore, we believe that (up to equivalence) only two presentations for $\mathrm{SL}(3,\mathbb{Z})$ were previously known.

The full code and results can be found at
`github.com/GLiversidge/PEACEv2`.

Our final method, which we call `LowIndAlg`, is a variation of the Low Index Subgroups algorithm, designed for finding specific subgroups by avoiding the inclusion of given words. To the best of the author's knowledge, this is the first procedure usable within Magma with such capabilities. Furthermore, the use of this procedure greatly outperforms previously available procedures in Magma in particular applications, with time reductions by a factor of over 400.

A similar function is available as an option within the GAP function `LowIndexSubgroupsFpGroup`, but our testing shows that `LowIndAlg` often greatly outperforms this function in terms of processing time. To the best of the author's knowledge, no such function is available in Magma.

We demonstrate the use of `LowIndAlg` in finding torsion-free subgroups of Coxeter groups, with implications for the construction of hyperbolic manifolds. This extends the work by Milnor, Lorimer and Everitt [33, 27, 15] and corrects work by Lorimer [28], and fills the remaining gaps in Milnor's table. It also determines (up to equivalence) *all* of the hyperbolic 3-manifolds of smallest possible volume constructible from each rank 4 Coxeter group $[p, q, r]$ with $p, q, r > 2$, $1/p + 1/q \geq 1/2$ and $1/q + 1/r \geq 1/2$.

The full code and results can be found at
`github.com/GLiversidge/LowIndAlg`.

Chapter 2

Further Background

In this chapter we give some further background on topics mentioned in Chapter 1 or needed later, including preliminaries on Tietze transformations and the Reidemeister-Schreier rewriting process, Schreier coset graphs, coset enumeration, the Low Index Subgroups algorithm, manifolds and surfaces, orientably-regular maps, manifolds of small volume and generators for special linear groups.

2.1 Preliminaries

A group homomorphism $\theta : G \to H$ is called *smooth* if its kernel $K = \ker \theta$ is torsion-free. In particular, this means that the order of every torsion element in G is preserved under the homomorphism θ. In that case, the image of G under θ is called a *smooth quotient* of G. For many finitely presented groups, such as triangle groups and some Coxeter groups, all that is required for smoothness is that the orders of the generators and their pairwise products (when those orders are

finite) are preserved. For example, if $G = \langle\, x, y \,|\, x^3, y^4, (xy)^5 \,\rangle$ then the quotient G/K is smooth if and only if the cosets Kx, Ky and Kxy have respective orders 3, 4 and 5 in G/K.

Given a set X of symbols (sometimes called an 'alphabet'), we may associate with X an adjunct set $X^{-1} = \{x^{-1} : x \in X\}$, and then define a *word* in or on $X \cup X^{-1}$ (or sometimes more simply, a word in or on X) to be any expression of the form $x_1^{e_1} x_2^{e_2} \dots x_m^{e_m}$ where $x_i \in X$ and $e_i = \pm 1$ for $1 \leq i \leq m$. Also we call m the *length* of this word, and we say that such a word is *reduced* if it contains no sub-expression of the form $x_i^{e_i} x_j^{-e_i}$ with $x_i = x_j$.

Next, the *free group* on X is the set of all reduced words on X, endowed with multiplication defined by concatenation of words followed by reduction (elimination of sub-expressions of the form $x_i^{e_i} x_j^{-e_i}$ with $x_i = x_j$). This group is denoted by $F(X)$. Its identity element is the empty word (of length 0). The group $F(X)$ is generated by X, and we say it has presentation $\langle\, X \,|\, \emptyset \,\rangle$, or simply $\langle\, X \,|\, \,\rangle$, with the '$\emptyset$' (or blank space) indicating that no relations hold in $F(X)$ other than those which are consequences of the group axioms. For example, if $X = \{x\}$ then $F(X)$ is the infinite cyclic group, with element set $\{x^m : m \in \mathbb{Z}\}$.

More generally, let R be a set of words on $X \cup X^{-1}$. Then the normal closure of R in $F(X)$ is the normal subgroup $N(R) = \langle\, R \,\rangle^{F(X)}$ generated by the words in R and all their conjugates in $F(X)$. For example, the normal closure of $\{x\}$ in $F(x, y)$ is the subgroup $N(\{x\}) =$

$\langle\, x^w \,|\, w \in F(x,y) \,\rangle$, where x^w denotes the conjugate $w^{-1}xw$ of x by w.

The factor group $F(X)/N(R)$ is then denoted as the group with presentation $\langle\, X \,|\, R \,\rangle$. The elements of X are usually referred to as the generators for this group G, and the elements of R are the *relators*. A *relation* in G is an equation of the form $r = 1$ where $r \in R$, or of the form $u = v$ where the equation $uv^{-1} = 1$ is deducible from the fact that the relators are all trivial in $G = F(X)/N(R)$.

The presentation $\langle\, X \,|\, R \,\rangle$ is said to be finite if both X and R are finite, and a group G is said to be *finitely-presented* if it admits a finite presentation (or equivalently, is isomorphic to the group with presentation $\langle\, X \,|\, R \,\rangle$ for some finite X and R).

2.2 Tietze transformations

Tietze transformations are a family of procedures that involve adding and/or removing generators and relators to or from a finite presentation, without changing the group described. If $G = \langle\, X \,|\, R \,\rangle$ then the following are Tietze transformations on G:

- T1 (Adding a relator): If $r \in N(R)$ then $G = \langle\, X \,|\, R \cup \{r\} \,\rangle$. This means that the word r can be added as a relator of G.
- T2 (Removing a relator): If $r \in R$ and $r \in N(R \backslash \{r\})$ then $G = \langle\, X \,|\, R \backslash \{r\} \,\rangle$. This means that the word r can be removed from the relators of G.

- T3 (Adding a generator): If w is a word on $X \cup X^{-1}$ and y is a symbol not in $X \cup X^{-1}$ then $G \cong \langle\, X \cup \{y\} \,|\, R \cup \{y^{-1}w\} \,\rangle$. This means that a symbol y representing the word w can be added to the generators of G when also adding $y^{-1}w$ is added to the relators of G.

- T4 (Removing a generator): If $y \in X$ and w is a word on $(X \cup X^{-1})\backslash\{y, y^{-1}\}$ such that $y^{-1}w \in R$ is the only relator containing y then $G \cong \langle\, X\backslash\{y\} \,|\, R\backslash\{y^{-1}w\} \,\rangle$. This means that the symbol y can be removed from the generators of G when also removing $y^{-1}w$ from the relators of G.

Example 2.2.1 Let $G = \langle\, a, b, c \,|\, a^2, abc, c^{-1}a^{-1}b^{-1}, (ab)^4 \,\rangle$.
We can show that $G \cong \langle\, a, b, d \,|\, a^2, aba^{-1}b^{-1}, d^{-1}ab, d^4 \,\rangle$ using Tietze transformations.
First we have

$$aba^{-1}b^{-1} = (abc)(c^{-1}a^{-1}b^{-1}) \in N(\{a^2, abc, c^{-1}a^{-1}b^{-1}, (ab)^4\})$$

so by T1 we can add $aba^{-1}b^{-1}$ as a relator of G, giving

$$G = \langle\, a, b, c \,|\, a^2, abc, c^{-1}a^{-1}b^{-1}, (ab)^4, aba^{-1}b^{-1} \,\rangle.$$

Conversely, we have
$c^{-1}a^{-1}b^{-1} = (abc)^{-1}(aba^{-1}b^{-1}) \in N(\{a^2, abc, (ab)^4, aba^{-1}b^{-1}\})$, so by T2 we can remove $c^{-1}a^{-1}b^{-1}$ from the relators of G, giving

$$G = \langle\, a, b, c \,|\, a^2, abc, (ab)^4, aba^{-1}b^{-1} \,\rangle.$$

Next, by T4 we may remove the symbol c, along with the relator abc, giving

$$G \cong \langle\, a, b \,|\, a^2, (ab)^4, aba^{-1}b^{-1} \,\rangle.$$

Then we can use T3 to add the symbol d representing the word ab, which gives

$$G \cong \langle\, a, b, d \,|\, a^2, (ab)^4, aba^{-1}b^{-1}, d^{-1}ab \,\rangle.$$

Now we use T1 to add

$$\begin{aligned} d^4 &= (ab)^4(d^{-1}ab)^{-1}((d^{-1}ab)^{-1})^d((d^{-1}ab)^{-1})^{d^2}((d^{-1}ab)^{-1})^{d^3} \\ &\in N(\{a^2, (ab)^4, aba^{-1}b^{-1}, d^{-1}ab\}). \end{aligned}$$

This gives

$$G \cong \langle\, a, b, d \,|\, a^2, (ab)^4, aba^{-1}b^{-1}, d^{-1}ab, d^4 \,\rangle.$$

Finally we use T2 to remove the relator

$$\begin{aligned} (ab)^4 &= (d^{-1}ab)^{d^{-1}}(d^{-1}ab)^{d^{-2}}(d^{-1}ab)^{d^{-3}}(d^{-1}ab)^{d^{-4}}d^4 \\ &\in N(\{a^2, aba^{-1}b^{-1}, d^{-1}ab, d^4\}). \end{aligned}$$

This gives

$$G \cong \langle\, a, b, d \,|\, a^2, aba^{-1}b^{-1}, d^{-1}ab, d^4 \,\rangle.$$

Note that the combination of the last two operations is equivalent to using the equality $d = ab$ in the relator $(ab)^4$.

When manipulating sets of relators the three following short cuts can be used to replace relators.

Theorem 2.2.1 *Let $G = \langle X \mid R \rangle$. Then Tietze transformations T1 and T2 can be used to replace any relator $r \in R$ by a conjugate r^w of r by any word w on $X \cup X^{-1}$.*

Proof. Let w be a word on $X \cup X^{-1}$. Then $r^w \in N(R)$ so by T1,

$$G = \langle X \mid R \cup \{r^w\} \rangle$$

Next $(r^w)^{w^{-1}} = w(w^{-1}rw)w^{-1} = r \in N((R \cup \{r^w\})\backslash\{r\}$, so by T2,

$$G = \langle X \mid (R \cup \{r^w\})\backslash\{r\}) \rangle.$$

□

Theorem 2.2.2 *Let $G = \langle X \mid R \rangle$ and let r_1 and r_2 be two relators in R. Then Tietze transformations T1 and T2 can be used to replace r_2 by the relator r_1r_2.*

Proof. First $r_1r_2 \in N(R)$, so by T1

$$G = \langle X \mid R \cup \{r_1r_2\} \rangle.$$

Next, $r_2 = (r_1^{-1})(r_1r_2) \in N((R \cup \{r_1r_2\})\backslash\{r_2\}$. Hence, by T2,

$$G = \langle X \mid R \cup \{r_1r_2\})\backslash\{r_2\} \rangle.$$

□

Theorem 2.2.3 *Let $G = \langle X \mid R \rangle$ and suppose that $yw^{-1} \in R$ for some element $y \in X$ and some word w on $X \cup X^{-1}$ and that r is a relator in R containing w such that $r = r_1 w r_2$ for some words r_1, r_2 on $X \cup X^{-1}$. Then Tietze transformations T1 and T2 can be used to replace r by the relator $r_1 y r_2$.*

Proof. First $r_1 y r_2 = (yw^{-1})^{r_1^{-1}} r \in N(R)$, so by T1

$$G = \langle X \mid R \cup \{r_1 y r_2\} \rangle.$$

Next, $r = ((yw^{-1})^{-1})^{r_1^{-1}}(r_1 y r_2) \in N((R \cup \{r_1 y r_2\}) \backslash \{r\}$ and so by T2,

$$G = \langle X \mid R \cup \{r_1 y r_2\}) \backslash \{r\} \rangle.$$

This process may be repeated to remove multiple appearances of w.

□

2.3 Schreier transversals and generators

Given a group G and a subgroup $H \leq G$, a subset $U \subseteq G$ is called a (right) *transversal* for H in G if the right cosets Hu for $u \in U$ are distinct and their union is G. In other words, a transversal for H in G is a set of representatives of the (right) cosets of H in G. For example, if $G = \langle x, y \mid xyx^{-1}y^{-1}, y^2 \rangle$ and H is the subgroup generated by x, then both $\{1, y\}$ and $\{x, y\}$ are transversals for H in G.

Let $G = \langle X \mid R \rangle$ and let $H \leq G$. A *Schreier transversal* for H in G is a transversal U with the property that if $u = x_1^{e_1} x_2^{e_2} \ldots x_{n-1}^{e_{n-1}} x_n^{e_n}$ is a

word on $X \cup X^{-1}$ that lies in U then also $x_1^{e_1} x_2^{e_2} \dots x_{n-1}^{e_{n-1}}$ lies in U. For example, if $G = \langle\, x, y \,|\, xyx^{-1}y^{-1}, y^2 \,\rangle$ and H is the subgroup generated by x, then the aforementioned $\{1, y\}$ is a Schreier transversal for H in G, but $\{x, y\}$ is not.

The *Reidemeister-Schreier rewriting process* is a procedure for finding a presentation for a subgroup H of finite index in a finitely presented group G. A description of the process can be found in Chapter 4 of [24], along with the following theorem on which the process is based.

Theorem 2.3.1 *Let $G = \langle\, X \,|\, R \,\rangle$ be a finitely presented group. Let U be a Schreier transversal for a subgroup H of finite index in G. For $g \in G$ let $\overline{g}$ denote the unique element $u \in U$ such that $Hg = Hu$. Then H is generated by the set*

$$B = \{ux(\overline{ux})^{-1} \,|\, u \in U, x \in X, ux \notin U\}.$$

Moreover, every element of the form uru^{-1} with $r \in R$ and $u \in U$ can be expressed as a word in the elements of B, and if S is the set of resulting expressions, then $\langle\, B \,|\, S \,\rangle$ is a presentation for H.

The (non-trivial) elements of the form $ux(\overline{ux})^{-1}$ with $u \in U$, $x \in X$ and $ux \notin U$ are called *Schreier generators* for the subgroup H (with regard to U).

2.4 Schreier coset graphs

A *Schreier coset graph* is a digraph associated with a group G, a generating set X for G, and a subgroup $H \leq G$. The vertices of the graph are the right cosets of H, and each vertex Hg is incident with directed edges (Hg, Hgx) for all $x \in X$. These directed edges correspond to multiplication of each coset Hg by each generator of G, and each such edge is labelled by the corresponding generator. (The associated reverse edges correspond to multiplication by the inverse of each generator of G.) In essence, the Schreier coset graph for H in G provides a graphical representation of the coset table for H in G.

For Example, if $G = \langle\, x, y \,|\, x^6, y^2, xyx^{-1}y^{-1} \,\rangle$ and H is generated by x^3 then the Schreier coset graph for H in G is given in Figure 2.1.

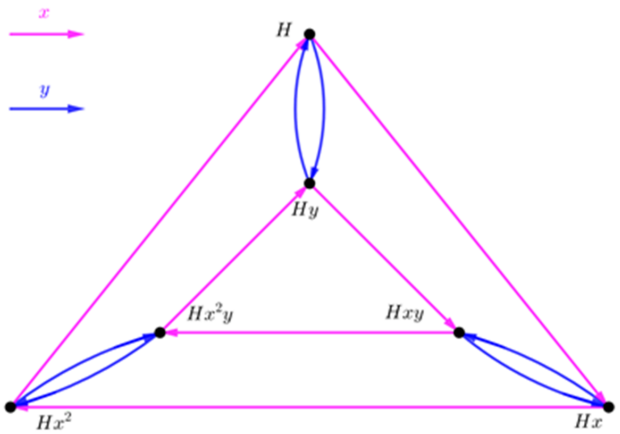

Figure 2.1: A Schreier coset graph

In [10], Conder showed how a Schreier coset graph can be used to illustrate the Reidemeister-Schreier rewriting process for finding a presentation of a given subgroup H of finite index in G. Here we give formal proofs for Conder's observations.

Theorem 2.4.1 *A spanning tree in the undirected form of the Schreier coset graph for a finite index subgroup H of $G = \langle X \mid R \rangle$ corresponds to a Schreier transversal for H.*

Proof. Let H be a finite index subgroup in the group $G = \langle X \mid R \rangle$. We can obtain a Schreier transversal U from a spanning tree in the undirected form of the Schreier coset graph for H in G as follows.

First, we include the identity element in U as a representative of the trivial coset H. Then for each other vertex v, we include the product of the generators and their inverses labelling the edges traversed on the unique path in the spanning tree from the vertex corresponding to the coset H to the vertex v. By the definition of a Schreier coset graph, this product is an element of the corresponding coset. Since there is exactly one such path to each coset of H, we see that $G = \cup_{u \in U} Hu$ and that for each $u, u' \in U$, if $Hu = Hu'$ then $u = u'$. Hence U is a transversal for H in G. Furthermore, if $u = x_1^{e_1} x_2^{e_2} \dots x_{n-1}^{e_{n-1}} x_n^{e_n} \in U$ then the path $(H, Hx_1^{e_1}, Hx_1^{e_1}x_2^{e_2}, \dots Hx_1^{e_1}x_2^{e_2} \dots x_{n-1}^{e_{n-1}}, Hu)$ is the unique path from H to Hu in the spanning tree, and so $(H, Hx_1^{e_1}, Hx_1^{e_1}x_2^{e_2}, \dots Hx_1^{e_1}x_2^{e_2} \dots x_{n-1}^{e_{n-1}})$ is the unique path from H to $Hx_1^{e_1}x_2^{e_2} \dots x_{n-1}^{e_{n-1}}$ in the spanning tree, and therefore $x_1^{e_1}x_2^{e_2} \dots x_{n-1}^{e_{n-1}} \in U$. Hence U is a Schreier transversal for H in G.

□

Theorem 2.4.2 *Let H be a finite-index subgroup of $G = \langle X \mid R \rangle$ and let Γ be the Schreier coset graph for H in G. Let T be the set of edges in a spanning tree of the undirected form of Γ and let U be the corresponding Schreier transversal. The edges of Γ which are not in T correspond to the Schreier generators of H, namely the elements of the set $B = \{ux(\overline{ux})^{-1} \mid u \in U, x \in X, ux \notin U\}$.*

Proof. Each directed edge (Hu, Hux) of the Schreier coset graph goes from the vertex Hu to the vertex $Hux = H\overline{ux}$, and gives rise to an element $ux(\overline{ux})^{-1} \in H$, which is either trivial (when $ux \in U$ and so $\overline{ux} = ux$, and the given edge lies in the spanning tree T), or a Schreier generator in the set B (when the edge $\{Hu, Hux\}$ lies outside T).

□

Next, let $G = \langle X \mid R \rangle$, H, U, Γ and T be as above. Label all edges of Γ represented in T with the identity element 1_H, and label every other directed edge (Hu, Hux) of Γ with the corresponding Schreier generator $ux(\overline{ux})^{-1}$.

If $g = \overline{ux}$, then $\overline{gx^{-1}} = \overline{uxx^{-1}} = \overline{u} = u$, and hence also the reverse arc $(Hux, Hu) = (Hg, Hgx^{-1})$ can be associated with the pair (g, x^{-1}), because $gx^{-1}(\overline{gx^{-1}})^{-1} = \overline{ux}x^{-1}u^{-1}$, which is the inverse of $ux(\overline{ux})^{-1}$.

Theorem 2.4.3 *Let G, H, U, Γ, T, B and the labelling of edges of Γ be as above. Then the following hold:*

(a) *If w is any word on $X \cup X^{-1}$, and $u \in U$, and W is the walk in Γ defined by w from the vertex u, then the concatenation of the labels on the walk W is equal to $uw\overline{uw}^{-1}$.*

(b) *If a word w on $X \cup X^{-1}$ defines a closed walk W in Γ from the vertex 1_G to itself, then the concatenation of the labels on the walk W is equal to w.*

(c) *If $r \in R$ and $u \in U$, then the element uru^{-1} may be written as a word in the elements of B by applying the relator r to vertex of Γ corresponding to u, and concatenating the labels of the edges traversed.*

(d) *If the word obtained by applying the relator $r \in R$ to the vertex corresponding to $u \in U$ is wb for some $b \in B$, then the Schreier generator b may be replaced by the word w^{-1}, and the associated edge of Γ may be labelled as such.*

Proof. Part (a) is easily seen to be true by induction on the length of w, because if we extend such a walk W by adding a further edge from Hw to Hwx (where $x \in X$), then we concatenate $uw\overline{uw}^{-1}$ with $\overline{uw}x(\overline{uwx})^{-1}$ to get $uw\overline{uw}^{-1}\,\overline{uw}x(\overline{uwx})^{-1} = uwx(\overline{uwx})^{-1}$, and the analogous property holds if we replace x by x^{-1}.

For part (b), by part (a) the labels of the given closed walk concatenate to $1_G w\overline{1_G w}^{-1} = w\overline{w}^{-1}$. But since this walk is closed we must have $\overline{w} = 1_G$, and hence the concatenation of the labels on this closed walk is equal to w.

For part (c), first note that the concatenation of any labels must give a word in the elements of B. Now let r be given by the word $x_1^{e_1}x_2^{e_2}\ldots x_{n-1}^{e_{n-1}}x_n^{e_n}$ on $X \cup X^{-1}$. Then the concatenation of the labels

of the edges traversed by the application of r to u is equal to $ur\overline{ur}^{-1}$, but $r \in R$ and hence $\overline{ur} = \overline{u} = u$. Thus $ur\overline{ur}^{-1} = uru^{-1}$, and hence the concatenation is uru^{-1}, as required.

Finally, for part (d), observe that wb is a relator for H. By Theorem 2.2.3, every appearance of b in other relators may be replaced by w^{-1}, and then we may use Tietze transformations to remove the generator b and the relator wb. Also re-labelling the edge of Γ associated with b with the word w^{-1} will cause the concatenation of the labels of any walk containing this edge to have the element b replaced by w^{-1}, and hence in particular, the concatenation of the labels of the closed walk corresponding to r from the vertex u will be $ww^{-1} = 1_H$.

□

`CosGraphLabs` applies parts (b) and (d) of the above theorem multiple times to find labels for the edges associated with the Schreier generators, replacing the elements of B with a chosen set of generators for H.

Example 2.4.1 Let $G = \langle R, S \,|\, R^8, S^8, (RS)^2 \rangle$. Then Figure 2.2 shows a labelled Schreier coset graph for the subgroup H generated by

$$\{R^3S^{-1}, R^2S^{-1}R, RS^{-1}R^2, S^{-1}R^3\},$$

with generators labelled w, x, y and z, respectively. The spanning tree

used corresponds to the Schreier transversal

$$U = \{1_G, R, S, R^{-1}, S^{-1}, R^2, RS, RS^{-1}\},$$

and its edges have all been labelled 1_H. (Note that the labelling of the cosets $H, HR, \ldots, HR^7$ corresponds to the Schreier transversal $\{1_G, R, \ldots, R^7\}$.)

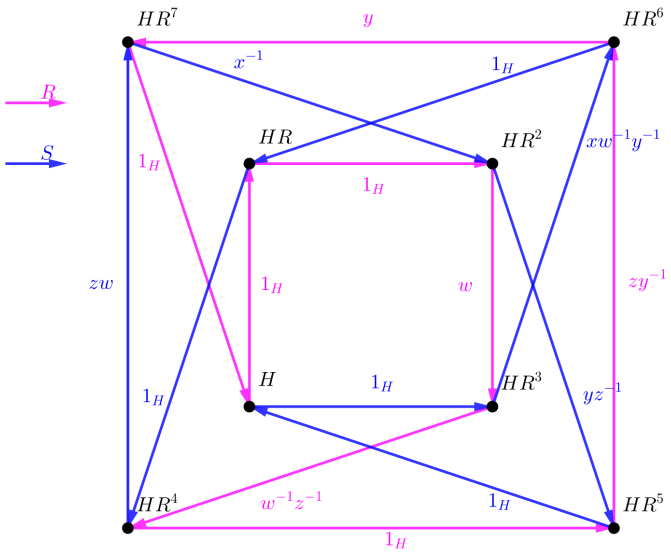

Figure 2.2: A labelled Schreier coset graph.

Next, note that the concatenation of the labels on the closed walk defined by each generator of H from the vertex H to itself reduces to the corresponding element w, x, y or z.

Also note that the concatenation of the labels on the edges of the

closed walks defined by the relators R^8 and $(RS)^2$ from any vertex reduces to 1_H, since the corresponding relators uru^{-1} of H have been used to label an edge as per Theorem 2.4.3 (d).

Applying the relators of G to each vertex in the graph gives us a single relator $xw^{-1}y^{-1}zwx^{-1}yz^{-1}$ for H. The concatenation of the labels on the closed walk defined by S^8 from each vertex is simply a conjugate of the concatenation of the labels on the closed walk defined by S^8 from another vertex. This happens because this walk is in fact a Hamilton cycle. By Theorem 2.2.1 only one of these words needs to be included as a relator.

The labelled graph not only provides an alternative to the MAGMA function `Rewrite`, but also can be used to express any element of H given in terms of the generators of G as a word in the chosen generators for H.

At the time of writing `CosGraphLabs`, we were unaware of the capability of MAGMA to do the latter via its function `!`, but our testing shows the expressions found via `CosGraphLabs` tends to take less processing time that those found using the inbuilt function in MAGMA, and in some instances quite significantly so. Furthermore, rewriting subgroups with `CosGraphLabs` was at least as fast as the MAGMA function `Rewrite`, and in some instances quite significantly faster.

2.5 Coset enumeration

Coset enumeration is a process for counting the number of cosets of a finite-index subgroup H in a finitely-generated group G and determining the associated coset table.
While there is no formal algorithm for coset enumeration, there are implementations that work quite well for most cases of interest. An early implementation of coset enumeration was given in 1936 by Todd and Coxeter [42].

Let $G = \langle X \mid R \rangle$ and suppose $H = \langle Y \rangle$ is a finite index subgroup of G. Briefly, the procedure involves creating several tables concurrently.

- A coset table, which has a column for each $x \in X \cup X^{-1}$ and a row for each coset. The (i, j)th entry of the coset table gives the number of the coset obtained by right multiplication of the coset numbered i by the element of $X \cup X^{-1}$ used for column j.

- Generator tables. For each $y \in Y$ we have a table with a single row. If y is given by the word $x_1 x_2 \dots x_n$ in the generators of G, then the table for y will have $n+1$ entries. The first and last entry are 1, and for $1 \leq i < n$ the $(i+1)$th entry is the number for the coset corresponding to the action of x_i on the coset given in the ith entry.

- Relator tables. For each $r \in R$ we have a table with a row for each coset. If r is given by the word $x_1 x_2 \dots x_n$ in the generators of G, then the table for r will have $n+1$ entries. The first and last entry

of each row are equal to the row number, and for $1 \leq i < n$ then the $(i+1)$th entry is the number for the coset corresponding to the action of x_i on the coset in the ith entry.

At each step of the procedure a coset B is created, defined as the coset given by the action of some generator $x \in X \cup X^{-1}$ on some previously defined coset A, filling the first gap in the coset table. Then the other tables are updated according to the table rules.

When all entries of a particular row are filled we update the coset table so that it is consistent with the table rules being satisfied. If an entry of the coset table where we wish to put a coset B is already filled with a coset A then we have a coincidence. By swapping the roles of A and B if necessary, let us suppose that $A < B$. Then each appearance of B in any table is replaced by A, and if the (i, B)th entry of the coset table has been filled, and the (i, A)th entry has not, then this value is copied to the (i, A)th entry. If both the (i, B)th entry and the (i, A)th entry have been filled then we have another coincidence between those two entries. Finally we remove the row B from the coset table and each relator table and then rename each coset greater than B to make the row numbers consistent with the coset names.

The procedure is complete when all tables have been filled.

Various improvements have been made over the last few decades, most notably those of Felsch [16] and HLT (Haselgrove, Leech and Trotter) [25], who suggested defining new cosets strategically to fill

the generator tables or relator tables respectively.

Computer implementations of coset enumeration called `ACE` (Advanced coset enumerator)[21] and `PACE` (Parallel Advanced coset enumerator) [37] were developed in 1999 and 2000 respectively by Havas and Ramsay, who subsequently developed `PEACE` in 2000 with the addition of 'proof extraction' (see [22, 23, 38]). Our implementation is based on that of Havas and Ramsay (with their permission), with a focus on increased usability.

The process involves tracking the definitions of new cosets, and the generator or relator used in any deductions made when filling the coset table. Records are maintained even for cosets which become inactive due to a coincidence, as well as recording the generator or relator which led to the coincidence and the equivalent coset. These records contain all the information necessary to 'explain' each entry of the coset table. The `PEACE` package uses this information to create *proof words*, a sequence of generators of the supergroup G, as well as round brackets surrounding some relators of G and square brackets surrounding some generators of the subgroup H.

The proof is obtained by the equivalence of two methods of simplification, one ignoring all brackets and removing sequential inverse pairs, and the other removing round brackets and their contents, and then removing only inverse pairs outside square brackets. It is easy to see that the resulting words must be equal, since only words which are equal to the identity in the supergroup are removed. Also it is

easy to see that the first method of simplification will give a word in the generators of the supergroup, but what is not obvious is that the second method of simplification (when applied to proof words created by the program) gives a word in the generators of the subgroup. This will be explained in Chapter 4, but the implication is that the words can be used to prove the inclusion of an element of the supergroup in the subgroup, as well as giving a word for this element in the given generators of the subgroup.

Our version is written as a function to be used in MAGMA. We removed the option of changing the enumeration style, and instead use a hybrid of the HLT and Felsch methods. We also removed the option of trying equivalent presentations (by rearranging the relators in a variety of ways), as they are unlikely to make much difference to the number of cosets created with this style of enumeration. Havas and Ramsay built the proof table after coset enumeration, but we build the proof table as we perform enumeration. Our version has an additional proof word table, which we build from the proof table on an 'as-required' basis. This increases the speed of production of proof words, especially when multiple proof words are needed, because sections of the word are likely to have been found already. Finally, we have also developed methods to reduce the length of proof words.

2.6 The Low Index Subgroups algorithm

The Low Index Subgroups algorithm (due to Sims [40]) uses the concept of partial coset enumeration to find subgroups up to a given index.

Let $G = \langle X \mid R \rangle$. To find subgroups of G up to index n, we begin by enumerating a certain number of cosets (greater than n), using a coset enumeration procedure, with no generator tables. Next we traverse a search tree, with branches corresponding to the inclusion of subgroup generators obtained through the forced coincidence of cosets or obtained through filling gaps in the coset table with existing cosets. After each branch point, enumeration is continued before moving to the next branch point. The branch ends when complete coset collapse occurs, that is, when the coset table reduces to that of the trivial subgroup, or reduces to a partial or complete coset table for a subgroup that is conjugate to one found already. Each complete coset table found in the search corresponds to a subgroup of G, or to a conjugacy class of such subgroups when the conjugacy test is applied.

Variations of this algorithm have been developed for finding specific types of subgroups, such as the 'lowx' package written by Peter Dobcsányi for finding only normal subgroups of up to the given index. A much more effective procedure for the latter was developed by Firth [17] (and his PhD supervisor Holt) by systematically considering the possible composition series for the finite quotient G/K of the group by the normal subgroup K. The Low Index Subgroups and Firth's Low Index Normal Subgroups algorithms are implemented

in MAGMA and GAP. The package lowx is available for download at `github.com/pdobsan/lowx`.

Our variation `LowIndAlg` is an implementation of the Low Index Subgroups algorithm, with the additional feature of an option to input a list of 'bad words'. This is a list of words on the generators of the given group G, which are not allowed to occur as elements in any of the subgroups produced (or in any of their conjugates). This feature allows the user to search for subgroups of small index with particular properties, such as torsion-free subgroups.

We use an HLT-Felsch hybrid style of enumeration to find subgroups of a specific index (rather than in a range of indices). Specifically, if n is the desired index, we enumerate up to n cosets. We create branches first by forcing the coincidence of cosets and continuing enumeration, and then by systematically considering possible ways to fill the coset table without coincidence.

At the time of writing `LowIndAlg` we were unaware of the similar procedure available as an option within the `LowIndexSubgroupsFpGroup` function in GAP, but we found that `LowIndAlg` often greatly outperforms `LowIndexSubgroupsFpGroup` in terms of processing time. Also, we believe no such procedure is available in MAGMA.

2.7 Manifolds and surfaces

An *n-manifold* is a topological space in which each point has a neighbourhood that is homeomorphic to n-dimensional Euclidean space. A 2-manifold is called a *surface*. Examples of surfaces include a plane in 3-dimensional Euclidean space, the boundary of a 3-dimensional solid such as a sphere or a torus, as well as the Klein bottle and Möbius strip.

A surface is called *non-orientable* if there exists a walk from one point on the surface to the same point on the other side of the surface without crossing any boundaries of the surface, and otherwise the surface is *orientable*. The sphere and torus are both orientable, while the Möbius strip and Klein bottle are both non-orientable.

The *genus* of a surface is the largest number of non-intersecting simple closed curves that can be drawn on the surface without separating it. The sphere has genus zero, the torus and Möbius strip have genus one and the Klein bottle has genus two.

2.8 Orientably-regular maps

A *map* is a 2-cell embedding of a connected graph or multigraph into a surface, where 2-cell means the embedding breaks up the surface into simply connected regions called the *faces* of the map. For example, the embedding of a planar graph into the plane without edge-crossings is a map. In this sense, maps can be seen as a generalisation of plane maps to other surfaces.

A map is called *orientable* if the underlying surface on which the graph is embedded is orientable, and *non-orientable* if the surface is non-orientable. The *genus* g of a map is the genus of the surface in which the map is embedded. The genus of a map is related to the *Euler characteristic* χ of the surface, with $\chi = 2 - 2g$ for orientable surfaces and $\chi = 2 - g$ for non-orientable surfaces.

The Euler characteristic is also related to the map by Euler's formula $\chi = V - E + F$, where V is the number of vertices, E is the number of edges and F is the number of faces of the map.

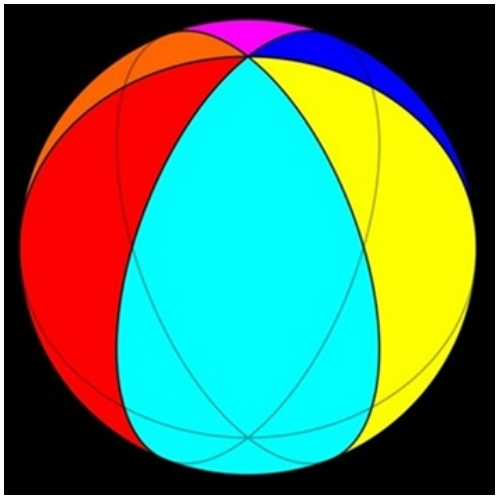

Figure 2.3: A polar map on the sphere (genus 0) with $|V| = 2$, $|E| = 6 = |F|$ (copied from [44]).

An *automorphism* of a map is a bijection from the map to itself tak-

ing vertices to vertices, edges to edges and faces to faces, and preserving the incidences between these objects. For example, automorphisms of the map in Figure 2.3 include swapping the two vertices without moving the edges of faces, or rotating the edges and faces one place around to the right. Swapping the red and yellow faces without moving other faces would not be an automorphism since that does not preserve the incidence of the edge between the red and orange face with those two faces.

A *flag* of a map is a triangle in its barycentric subdivision (obtained by adding a line from a central point P of each face to each of the vertices of that face, and a line from P to the midpoint of each edge of that face). Accordingly, a flag can be identified as a triple consisting of a vertex v, half of an edge e incident with v and a triangular section of a face f incident with both v and e. In all but some degenerate cases, a flag can be thought of more simply as an incident vertex-edge-face triple. An *arc* of a map is an incident vertex-edge pair.

A map is called *regular* if its automorphism group is transitive on its flags. An orientable map is called *orientably-regular* if its group of orientation-preserving automorphisms is transitive on its arcs. If an orientably-regular map has an automorphism a that reverses an edge e but preserves an incident face then the map is called *reflexible*, and otherwise the map is called *chiral*. Chiral maps occur in pairs, with each map in the pair being like a mirror image of the other. Orientably-regular maps which are reflexible are regular orientable maps. Chiral maps are orientably-regular, but are not regular.

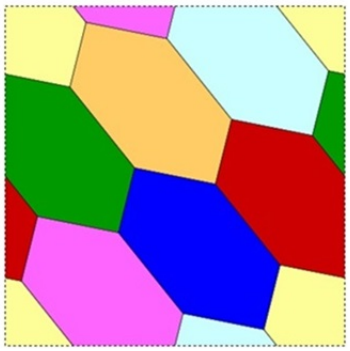

Figure 2.4: A chiral map of type $\{6, 3\}$ on the torus (copied from [14]).

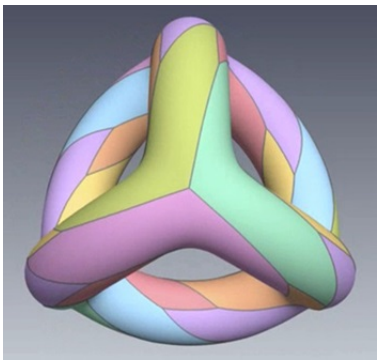

Figure 2.5: The Klein map (a regular map of genus 3 with type $\{7, 3\}$) [45].

Regular and orientably-regular maps have an automorphism group which acts transitively on vertices, edges and faces. It follows that all faces have the same number of edges, say p, and all vertices have the same degree, say q. The pair $\{p, q\}$ is called the *type* of the map, and despite the use of curly brackets, is considered as an ordered pair.

Regular maps are closely related to triangle groups. The *full* or *extended triangle group* $\Delta(\ell, m, n)$ is the group with the presentation

$$\langle\, a, b, c \,|\, a^2, b^2, c^2, (ac)^\ell, (ab)^m, (bc)^n \,\rangle.$$

The *ordinary triangle group* or *von Dyck group* $D(\ell, m, n)$ is the group with presentation

$$\langle\, x, y \,|\, x^\ell, y^m, (xy)^n \,\rangle.$$

Now let $G = \langle\, a, b, c \,\rangle$ be any finite smooth quotient of the full triangle group $\Delta(2, p, q)$, generated by elements a, b, c of order 2 such that ac, ab and bc have orders 2, p and q, where $p, q \geq 2$. Then a map $M = M(a, b, c)$ may be constructed from G by taking the vertices, edges and faces as the right cosets in G of the subgroups $V = \langle\, b, c \,\rangle$, $E = \langle\, a, c \,\rangle$ and $F = \langle\, a, b \,\rangle$, respectively, and with incidence defined by non-empty intersection of these cosets. Then M is a regular map, with type $\{p, q\}$ and with G as its automorphism group (acting by right multiplication on cosets). Also M is orientable if the subgroup $\langle\, ab, bc \,\rangle$ of G has index 2, and non-orientable if it has index 1.

Similarly if $H = \langle\, x, y \,\rangle$ is a finite smooth quotient of the ordinary triangle group $D(2, p, q)$, generated by elements x and y of orders 2 and p such that xy has order q, then an orientably-regular map M of type $\{p, q\}$ can be constructed from H in the analogous way by taking right cosets of $V = \langle\, R \,\rangle$, $E = \langle\, RS \,\rangle$ and $F = \langle\, S \,\rangle$ where $R = y$

and $S = (xy)^{-1} = y^{-1}x$, and with H preserving orientation and acting transitively on the arcs of M.

In this case H is called the *rotation* group of the map. If the map is reflexible then H has index 2 in the full automorphism group, while for a chiral map H is the full automorphism group of the map.

Lists by Conder of all regular orientable maps and all orientably-regular but chiral maps of genus 2 to 301 are available at `www.math.auckland.ac.nz/~conder`. For each map in the list, the following is included:

- the type of the map,
- the order of the full automorphism group of the map,
- defining relators for the full automorphism group of the map,
- the vertex multiplicity (the number of edges incident with a given pair of adjacent vertices),
- the face multiplicity (the number of edges incident with a given pair of incident faces).

For each such map of genus g from 2 to 12 we have found a 'nice' set of $2g$ generators for the torsion-free normal subgroup N in the associated ordinary triangle group D, such that the rotation group is isomorphic to D/N. We have also determined the effect of conjugation by the generators of the triangle group D on the nice generators of N. By *nice* we mean that this information about conjugation of the

chosen generators of N by generators of D can be expressed succinctly. See Chapter 3.

It is well known that the minimum number of generators for the torsion-free normal subgroup N is $2g$, where g is the genus of the orientably-regular map M – or equivalently, for the orientable surface S carrying M, upon which the finite quotient group D/N acts – and moreover, that N is isomorphic to the fundamental group of the surface S, and has a presentation of the form

$$\langle\, a_1, b_1, \ldots, a_g, b_g \,|\, [a_1, b_1] \ldots [a_g, b_g] = 1 \,\rangle.$$

These facts about a canonical presentation for N were explained in [26], but without detailed justification. A proof of the first part of this may be found in [2], and the rest is quite classical and may be found in just about any good textbook on algebraic topology, such as [18] or [30].

Such generating sets (and the effect of the generators of the parent group by conjugation on them) can be used to determine important aspects of the structure of the fundamental group, as used many times in work on regular maps and actions of triangle groups on Riemann surfaces (see for example [3, 6, 9, 29]). Moreover, finding such generating sets extends the pioneering work by Leech [26] for the case of the ordinary (2,3,7) triangle group, which was critical to the determination in [9] of all Hurwitz groups (conformal automorphism groups of compact Riemann surfaces of maximum order) for genus 2 to 11905.

2.9 Manifolds of small volume

The study of small-volume 3-manifolds dates back to the early twentieth century. The smallest volume closed orientable hyperbolic 3-manifold is the Weeks-Matveev-Fomenko manifold, discovered independently by Weeks in [48] and Matveev and Fomenko in [31]; see [19]. The smallest volume non-compact 3-manifold is the (non-orientable) Gieseking manifold, discovered by Gieseking in [20] 1912, see [1]. The smallest volume non-compact orientable manifold is the figure 8 knot complement and a 'sibling' of that one; see [8].

For $n \geq 3$, an n-manifold $\mathcal{M}$ can be constructed from a torsion-free finite-index subgroup Λ of a $[k_1, \ldots, k_n]$ Coxeter group Γ, by considering features of the natural permutation representation of Γ on the right cosets of Λ. Furthermore, various properties of $\mathcal{M}$ may be computed from the natural permutation representation of the Γ on the right cosets of the Λ and from the parameters $k_1, \ldots, k_n$. The $[k_1, \ldots, k_n]$ Coxeter group is the finitely-presented group generated by $n+1$ involutions $x_1, \ldots, x_{n+1}$ with $(x_i x_{i+1})^{k_i} = 1$ for $1 \leq i \leq n$ and $(x_i x_{j+1})^2 = 1$ for $1 \leq i < j \leq n$.

For example, the manifold $\mathcal{M}$ is orientable if and only if Λ is contained in the index 2 subgroup of Γ generated by the products $x_i x_j$ of the canonical generators of Γ, or equivalently, if and only if every one of the generators of Λ is a word of even length in the canonical generators x_i of Γ. For further details (and many examples), the reader may refer to [15, 27, 33, 39].

Various small volume 3-manifolds were constructed by Milnor [33], Lorimer [28] and Everitt [15], by finding torsion-free subgroups of minimum possible index in $[p, q, r]$ Coxeter groups with $1/p + 1/q \geq 1/2$ and $1/q + 1/r \geq 1/2$. There are 15 such Coxeter groups: four spherical, one Euclidean and ten hyperbolic. In the spherical cases, the group is finite and hence there are no non-trivial torsion-free subgroups. In the hyperbolic cases, Milnor found single examples for all triples $[p, q, r]$ except [3,5,3], [4,3,5], [5,3,6] and [6,3,6], and gave incomplete information about them in a table, copied (with some minor simplifications) below.

p, q, r	$V_3(\Sigma_{p,q,r})$	L.C.M	least $\#(\Gamma/\Pi)$	$V_3(H^3/\Pi)$
$3, 5, 3$	0.0390503	120	?	
$4, 3, 5$	0.035885	240	?	
$5, 3, 5$	0.0933255	120	120	11.199064
$3, 3, 6$	0.0422892	24	24	1.0149416
$4, 3, 6$	0.1057231	48	48	5.074708
$3, 4, 4$	0.0763305	48	48	3.66386
$5, 3, 6$	0.1715017	120	?	
$3, 6, 3$	0.1691569	12	12	2.029883
$6, 3, 6$	0.2537354	24	?	
$4, 4, 4$	0.2289914	16	16	3.66386

Table 2.1: Milnor's table of hyperbolic 3-manifolds of small volume

Some spaces in the table have since been filled. Everitt found examples for [3,5,3] and [5,3,6]. Lorimer attempted to deal with the [4,3,5] case, but due to poor health he made the mistake of looking for subgroups of the wrong index (120 instead of 240). The case of the [6,3,6] Coxeter group was also left open.

Two facts help us to find the minimum possible index of torsion-free subgroups. First, let m be the lowest common multiple of the orders of all finite subgroups of a given group G. Then by considering the natural action of such finite subgroups on right cosets of a given subgroup of finite index in G, it is easy to see that every finite subgroup must act semi-regularly on the cosets, and hence every finite-index torsion-free subgroup of G must have index km for some $k \in \mathbb{N}$. Secondly, every finite subgroup of an infinite $[k_1, \ldots, k_n]$ Coxeter group is conjugate to a finite subgroup of $H_i = \langle\, x_1, x_2, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n \,\rangle$ for some i in the range $1 \leq i \leq n+1$; see [15, 7].

The question of uniqueness of minimum-volume manifolds in [33] has been answered negatively (for example see [15]), but the classification up to isometry of such minimum-volume hyperbolic manifolds has remained open. This question is equivalent to finding the isomorphism classes of minimum-index torsion-free subgroups of the associated Coxeter groups; see [36], [34].

We use these facts to find all isomorphism classes of minimum finite-index torsion-free subgroups of the 11 non-spherical Coxeter groups described above, filling the last two holes in Milnor's table. See Chapter 5.

2.10 Generators for special linear groups

The *special linear group* $\mathrm{SL}(n, \mathbb{Z})$ is the group of all $n \times n$ matrices with integer entries and determinant 1, with the group operation of matrix multiplication. This group is generated by the transvections T_{ij} for $1 \leq i \neq j \leq n$, where T_{ij} is the $n \times n$ identity matrix with an extra 1 as its (i, j)th entry. One presentation for $\mathrm{SL}(n, \mathbb{Z})$ on these generators is given in [32] by the relations

- $[T_{ij}, T_{k\ell}] = 1$ for $i \neq \ell$ and $j \neq k$,
- $[T_{ij}, T_{jk}] = T_{ik}$ for i, j, k all distinct, and
- $(T_{12}T_{21}^{-1}T_{12})^4 = 1$.

Other generating sets are known, but not many presentations are known.

Trott showed in [43] that $\mathrm{SL}(n, \mathbb{Z})$ can be generated by an element of order n and an element of infinite order, for all odd n. Tamburini, Wilson, Vsemirnov and others showed that $\mathrm{SL}(n, \mathbb{Z})$ can be generated by an element of order 2 and an element of order 3 whenever $n \geq 5$, and cannot be generated by such elements for $n < 5$ (see [41, 46, 47]).

As far as we are aware, until 2022 there were (up to equivalence) only two known presentations for the group $\mathrm{SL}(n, \mathbb{Z})$: the one above in terms of the $n^2 - n$ transvections, subject to the Steinberg relations (see [32]), and another on three generators obtained with the help of the former (see [13]).

Note that group presentations on a small number of generators are not just aesthetically pleasing but can also be very helpful in analysing the structure of the group and its homomorphic images. Some important and well-known computational procedures such as the `LowIndexSubgroups` algorithm and the `LowIndexNormalSubgroups` procedure work much faster when the number of generators (and relators) is small.

The 3-generator presentation for $\mathrm{SL}(3, \mathbb{Z})$ given in [13] is as follows:

$$\langle\, x, y, z \mid x^3, y^3, z^2, (xz)^3, (yz)^3, (xy)^6, (x^{-1}zxy)^2, (y^{-1}zyx)^2 \,\rangle.$$

We used this presentation (and `PEACEv2`) to find several new 2-generator presentations for $\mathrm{SL}(3, \mathbb{Z})$. We believe that these new presentations are the only known finite presentations for $\mathrm{SL}(3, \mathbb{Z})$ on two generators.

This work led to the discovery of other new generating pairs for $\mathrm{SL}(3, \mathbb{Z})$ and new generating pairs for $\mathrm{SL}(n, \mathbb{Z})$ for all $n > 3$ (see [12]), but the latter work falls outside of the scope of this thesis.

Chapter 3

CosGraphLabs

3.1 CosGraphLabs algorithm

CosGraphLabs is a MAGMA function which takes a finitely-presented group $G = \langle X \mid R \rangle$ and a set $\{y_1, \ldots, y_n\}$ of elements of G which generate a finite index subgroup H and returns an array L of labels for the edges of the Schreier coset graph for H in G as words in the given generators of H. This labelled Schreier coset graph L can then be used to find a presentation for H, and also for any element $h \in H$ given by a word in the generators of G, the graph L can be used to find a word for h in the generators of H.

3.1.1 Definitions

Throughout the algorithm, group and subgroup generators are stored as integers, and words in these generators are stored as arrays of the

corresponding integers, with negative values used to indicate inverses. The identity element is stored as an empty array. Since each coset of H in G corresponds to a vertex in the Schreier coset graph for H in G, the same integer value will be used to refer to the coset and it corresponding vertex.

Let R be the set of relators of G, given as words in $X \cup X^{-1}$, and let R_c be the set containing all cyclic rearrangements of r and their inverses, for each $r \in R$. For example if $R = \{x^2y\}$ then

$$R_c = \{x^2y, xyx, yx^2, y^{-1}x^{-2}, x^{-2}y^{-1}, x^{-1}y^{-1}x^{-1}\}.$$

We define L to be the array in which we will store the labels we have assigned to the edges in the coset graph. The labels are words in the elements of H, and the word $L[v, g]$ at coordinates (v, g) gives the label of the directed edge from the vertex v to the vertex obtained by multiplication by the generator $g \in G$. Recall that this is equal to $u_v g(\overline{u_v g})^{-1}$, where u_v is the element in the Schreier transversal that lies in the coset represented by the vertex v. This coordinate system is the same as the coordinates in the coset table for H in G.

We start with $L[v, g] = 0$ for all v and all g, to signify that these edges have not yet been labelled, and call these unlabelled edges *zero edges*. Whenever we label an edge $L[v, g] = w$ we must also label its reverse as $L[vg, g^{-1}] = w^{-1}$. This can be assumed throughout the algorithm.

Example 3.1.1 Let $G = \langle\, x, y \,|\, x^6, y^2, xyx^{-1}y^{-1} \,\rangle$ and H be the subgroup generated by x^3, with coset table as given below in Table 3.1.

	x	y	x^{-1}	y^{-1}
1	2	3	4	3
2	4	5	1	5
3	5	1	6	1
4	1	6	2	6
5	6	2	3	2
6	3	4	5	4

Table 3.1: Coset table for H in G

Then at some point during the algorithm, L might be as shown in Table 3.2. At such a point, the graph has five (unlabelled) zero edges, namely $[3, x]$, $[4, y]$, $[5, x]$, $[5, y]$, and $[6, y]$. Six of the edges have been labelled with the identity element, namely $[1, x]$, $[1, y]$, $[2, y]$, $[3, y]$, $[4, x]$, and $[6, x]$. Recall that the identity element is represented by an empty sequence, and so these entries are left blank in Table 3.2. The final edge $[2, x]$ has been labelled with the single generator of H, and is indicated by the entries 1 ($= L[2, x]$) and -1 in Table 3.2.

	x	y	x^{-1}	y^{-1}
1				
2	1			0
3	0			
4		0	-1	0
5	0	0	0	
6		0	0	0

Table 3.2: L

If H had more generators we would likely see longer labels such as "1,2,3" and its inverse "-3,-2,-1", where 1,2, and 3 refer to three different generators of H. In subsequent sections we shall refer to the labels as their intended values rather than sequences of integers.

3.1.2 Walks in the Schreier coset graph

Any word in the generators of G defines a walk in the Schreier coset graph from each vertex, by taking the edges corresponding to successive letters of the word. Relators of G trace out a closed walk from every vertex, and elements of H trace out a closed walk from coset 1.

By Theorem 2.4.3 (b), in order to get a presentation on the given generators $y_1, \ldots, y_h$ of H, the concatenation of the labels of the edges on the closed walk defined by each y_i from coset 1 must reduce to y_i or to an element which is equal to y_i in G. We say that the generator y_i is *labelled* once all edges on the closed defined by y_i from coset

1 to itself have been labelled, and otherwise, we say that y_i is *unlabelled*.

Example 3.1.2 In the Schreier coset graph corresponding to the coset table in Table 3.1, the generator of x^3 defines the closed walk ($[1, x]$, $[2, x]$, $[4, x]$) from coset 1. If we call this generator a, then the concatenation of the labels on this walk in Table 3.2 would be $1_H a 1_H$, which is equal to a as required. The generator a is labelled in this table. Suppose $b = yxyx^{-1}$ were a second generator of H. Then this generator would define the closed walk ($[1, y]$, $[3, x]$, $[5, y]$, $[2, x^{-1}]$) from coset 1. The edges $[3, x]$ and $[5, y]$ are both zero edges, so this generator would be unlabelled.

3.1.3 Labelling edges

The first step of the algorithm is to label the edges of a spanning tree of the graph using the label 1_H on each edge. As we add edges to the spanning tree we record the vertices which have been connected to it in a list T, which is initialised to contain the vertex associated with coset 1. We systematically add edges as follows; for each $v_1 \in T$, if there is an edge $[v_1, g]$ from coset v_1 to coset v_2, and $v_2 \notin T$, then we set $L[v_1, g]$ to 1_H, and add v_2 to T. Note that any spanning tree may be used.

Example 3.1.3 In the Schreier coset graph corresponding to the coset table in Table 3.1, we start with $T = \{1\}$, and first consider the edge $[1, x]$. This edge goes from coset 1 to coset 2, which is not yet in T,

so we label the edge 1_H, and then T becomes $\{1, 2\}$. Next we consider the edge $[1, y]$, and so forth.

Once all vertices have been added to T, the edges we have labelled 1_H in the process will be the spanning tree.

Next, we use relations and H-generators to label as many other edges of the Schreier coset graph as possible. By Theorem 2.4.3 (c), the concatenation of the labels on the closed walk defined by each relator from each coset must be equal to the identity. We may use Theorem 2.4.3 (d) to label an edge if it is the only zero edge of a closed walk defined by a relator from a particular coset.

Example 3.1.4 Let x^3 be a relator of G which defines the closed walk $([2, x],\ [3, x],\ [4, x])$ from coset 2. If $[3, x]$ is a zero edge and $[2, x]$ and $[4, x]$ are labelled a and bc respectively, then $[3, x]$ can be labelled $a^{-1}c^{-1}b^{-1}$, which makes the concatenation of the labels on this walk reduce to the identity.

Similarly, if the closed walk defined by a H-generator y_i from coset 1 has only one zero edge, then we may use Theorem 2.4.3 (b) to conclude that the concatenation of the labels of this walk must reduce to an element of G equal to y_i, in order to label the edge. Note that H-generators are applied only to coset 1, since the walk defined by some generator y from some coset v will be closed only if $Hvy = Hv$, that is, only if y lies in the conjugate $v^{-1}Hv$, which is not assumed.

Example 3.1.5 Let $b = xyx^2$ be a generator of H which defines the closed walk ($[1, x]$, $[2, y]$, $[3, x]$, $[4, x]$) from coset 1. If the first three edges are labelled 1_H, a, and ab, respectively, and the edge $[4, x]$ is a zero edge, then we label it $b^{-1}a^{-2}b$ so that the concatenation of the labels on this walk reduces to b.

Since H-generators are applied only to coset 1, we simply keep track of which H-generators are unlabelled, to avoid redundant processing. For each unlabelled H-generator y_i, if the closed walk defined by y_i from coset 1 has a single zero edge, then we label this edge so that the concatenation of the labels of the edges of the closed walk will reduce to y_i.

On the other hand, relators are applied to every coset, and for those we track 'new entries'. For each new entry in L at coordinates $[v, g]$, we apply each relator r in the list R_c of all cyclic rotations and inverses of relators in R having first letter g. If the walk defined by r from coset v has a single zero edge, then we label this edge in such a way that the concatenation of the labels of the edges of this walk will reduce to the identity.

Example 3.1.6 Let $G = \langle\, x, y \,|\, x^6, y^2, xyx^{-1}y^{-1} \,\rangle$ and suppose $[1, x]$ is a new entry. Then we apply the relators x^6, $xyx^{-1}y^{-1}$ and $xy^{-1}x^{-1}y$ to coset 1. The closed walk defined by each of these relators from coset 1 will contain the newly labelled edge $[1, x]$ and hence their application may yield a different result to any previous applications.

By applying only the appropriate relators to new entries, we ensure that every relator is applied to every vertex every time a zero-edge of the related closed walk is labelled, while avoiding the reapplication of relators to vertices from which the labels of associated walk remain unaltered. This increases efficiency and ensures that the application of relators will complete (although the graph might still have some zero edges upon completion).

3.1.4 Temporary generators

A *temporary generator* is a label given to a zero edge temporarily, to represent the corresponding Schreier generator. In many instances, simply applying H-generators and relators does not produce a labelled Schreier coset graph. In these cases, we take a copy L_2 of L to experiment with different combinations of zero edges labelled with temporary generators. We then apply relators and H-generators until a label for a zero edge which does not contain any of the temporary generators is found, or no more labels of L_2 can be found.

We have three procedures for selecting a set of zero edges to label with temporary generators. In each procedure, we start with the smallest possible number of temporary generators and increase this number until a new label is found, or until all sets are exhausted, before moving on to the next type of set selection.

In our first procedure, we select edges to 'fill' unlabelled H-generators. For each unlabelled H-generator y_i, if the walk defined by y_i from coset

1 has $M+1$ zero edges then we label M of them with temporary generators, and then the application of y_i provides the label of the final zero edge.

Example 3.1.7 Let $a = x^3y$ be a generator of H which defines the walk $([1,x],\ [2,x],\ [3,x],\ [4,y])$ from coset 1. If $[1,x]$ and $[2,x]$ are both labelled with the identity then when $M = 1$ we would label the edge $[3,x]$ with a temporary generator, say t. Now the application of a to coset 1 allows for the edge $[4,y]$ to be labelled $t^{-1}a$ so that the concatenation of the labels on this walk reduces to a.

In our second procedure, we select edges to 'fill' relators. For each relator $r \in R$ and each coset c, if the walk defined by r from coset c has $M+1$ zero edges then we label M of them with temporary generators, and then the application of r to c provides the label of the final zero edge.

Example 3.1.8 Let $r = x^2y^2$ be a relator of G which defines the closed walk $[5,x],\ [3,x],\ [7,y],\ [4,y]$ from coset 5. If the edge $[5,x]$ is labelled aba and the remaining three edges are zero edges, then at $M = 2$ we label $[3,x]$ and $[7,y]$ with temporary generators, say t_1 and t_2. The application of r allows the final edge of the walk to be labelled $t_2^{-1}t_1^{-1}a^{-1}b^{-1}a^{-1}$.

In our third procedure, we systematically go through each possible set of M zero edges.

When using temporary generators, the application of relators and H-generators must follow a slightly different procedure. We still apply all unlabelled generators and apply relators to new entries, but the actions taken as a result of these applications will vary, with each application falling into one of seven categories.

First, if there is more than one zero edge in the walks defined by the relator or generator then we take no action as in the main algorithm.

There are two categories involving closed walks with a single zero edge. As in the main algorithm, we find the label such that the concatenation of the labels on the edges of the walk would reduce to the generator associated with the walk, or the identity in the case of a walk associated with a relator, and the choice of action depends on the label we find. If the label contains temporary generators then it is assigned to the zero edge in the copied labels, L_2, and then we continue applying generators and relators using this set of labels. If the label does not contain any of the temporary generators then the label is assigned to the corresponding zero edge in L, and then we discard L_2 and return to the main algorithm.

Example 3.1.9 Let $r = x^3$ be a relator of G which defines the closed walk $[3, x]$, $[10, x]$, $[7, x]$ from coset 3. If $[3, x]$ and $[10, x]$ are labelled abc and d respectively, then the label we find is $w = d^{-1}c^{-1}b^{-1}a^{-1}$. If none of a, b, c, d is a temporary generator then we label the edge $[7, x]$ with w in L and discard L_2. If (say) d is a temporary generator then

we label $[7, x]$ with w in L_2, and leave L untouched.

The last four categories involve closed walks with no zero edges. In these instances the concatenation of the labels for the edges of the walk forms a relator for H, either as a word on its own if the walk is associated with a relator of G, or the word formed when it is concatenated with the inverse of the associated generator. Our action again depends on the presence of temporary generators in this word.

Example 3.1.10 Let $a = x^4$ be a generator of H which defines the closed walk $([1, x], [2, x], [4, x], [8, x])$ from coset 1. Suppose in L_2 these edges have been labelled 1_H, t_1, t_2 and $t_2^{-1}t_1^{-1}a$, respectively, where t_1 and t_2 are temporary generators. Next, let $r = y^2$ be a relator of G which defines the closed walk $[5, y]$, $[6, y]$ from coset 5. If the labels of these edges are the words w_1, w_2 respectively, then the word $w = w_1w_2$ is a relator of H. The next step depends on the presence of t_1 and t_2 in w.

If the word contains no temporary generators then no action is taken. If only one temporary generator t_i appears in the word and it appears only once in the word, then there is a conjugate of the word or its inverse having the form wt_i^{-1}, and since this is a relator of H, the generator t_i can be removed from the presentation by Theorem 2.2.3. To achieve this, we label the zero-edge associated with t_i with the label w in L. Then we discard L_2 and return to the main algorithm.

If the word contains more than one temporary generator and each temporary generator appears more than once then no action is taken. On the other hand, if at least one temporary generator (say t_i) appears only once then there is a conjugate of the word, or its inverse, having the form wt_i^{-1}, where w does contain temporary generators, but does not contain t_i. Then in this case, again t_i can be removed from the presentation, but we do this only in L_2. Every appearance of t_i or t_i^{-1} in a label or labels in L_2 is replaced with w or w^{-1}, respectively. If t_i or t_i^{-1} is replaced in a label then we check to see if this new label contains any temporary generators. If the label does not contain any temporary generators then we assign the label to the corresponding edge in L, and then we discard L_2 and return to the main algorithm. If the label does contain temporary generators then we update the label in L_2, and continue replacing appearances of t_i and t_i^{-1}. Once all appearances of t_i and t_i^{-1} have been removed from the labels in L_2 we return to applying relators and generators.

Example 3.1.11 Continuing our previous example, if neither t_1 nor t_2 are in w then no action is taken, but if $w_1 = t_1$, and w_2 contains neither t_1 nor t_2, then in L we may label the edge $[2, x]$ with the word w_2^{-1} and discard L_2. If $w = t_1abt_2abt_1t_2$, then both temporary generators appear twice and so no action is taken. If $w = t_1at_2a^{-1}t_2^{-1}$, then t_1 appears only once and may be removed from L_2, with any appearances of t_1 being replaced by $t_2at_2^{-1}a^{-1}$ and appearances of t_1^{-1} being replaced by $at_2a^{-1}t_2^{-1}$.

Once all new entries of L_2 have been processed and any remaining

unlabelled generators have been applied to coset 1 (with no new labels created), we discard L_2 and move on to the next set of zero edges. If M reaches the maximum value for each selection type, then the incomplete labels are returned.

Note that maximum values for each selection type are calculated as follows:

- The maximum value for filling unlabelled generators is equal to the maximum length of the given generators.
- The maximum value for filling relators is equal to the maximum length of relators in G.
- The maximum value for systematic sets is equal to the number of unlabelled edges.

These values ensure that all possible sets are explored while guaranteeing that the function `CosGraphLabs` will terminate. Note that while the function will terminate, it is not guaranteed to return a Schreier coset graph without zero-edges. In other words, the labels on the coset graph are not guaranteed to be complete. If the labels are incomplete then they may not be used for rewriting, but they can provide words for a particular element h of H, provided that the walk defined by h from coset 1 does not traverse any zero edges.

3.1.5 Using the labelled Schreier coset graph

To get a presentation for H with generators $\{y_1, \ldots, y_n\}$, we use L to obtain the relations for H. For each $r \in R$ and each coset v, the con-

catenation of the labels of the walk defined by r from v gives a relator for H. For each generator y_i of H, the labels of the walk defined by y_i (when given as a word in the generators of G) from the coset 1 reduce to a word w_i for y_i in the generators of H, and we can add $w_i y_i^{-1}$ to the relators for H.

Example 3.1.12 If $r \in R$ defines the closed walk $[3, a]$, $[4, b]$, $[6, b^2]$ from coset 3 and these edges are labelled xy, xy^2 and $y^{-1}z^2$ respectively, then the word $xyxyz^2$ is a relator of H. If $y = abc$ is a generator of H, and defines the closed walk $[1, a]$, $[2, b]$, $[5, c]$ from coset 1, and these edges are labelled 1_H, x^2y, x^2, respectively, then the word $x^2yx^2y^{-1}$ is a relator of H.

To see that this gives a presentation, first note that by Theorems 2.2.1 and 2.2.2, the edges of the spanning tree correspond to the elements of a Schreier transversal U and all other edges correspond to elements of the associated Schreier generators B. As previously stated the labels of the edges are obtained by the application of Theorems 2.2.4 and 2.2.6 along with Tietze transformations and each is equal to the Schreier generator associated with that edge. By Theorem 2.3.1 the relators required to form a presentation for H on the Schreier generators B is the set S of expressions in the elements of B for the elements $\{ur^{-1}u \mid u \in U, r \in R\}$. By Theorem 2.2.5 the expression for the element uru^{-1} may be obtained by the concatenation of the labels on the closed walk defined by r from the vertex w. The additional relations of H given by $w_i y_i^{-1}$ for each y_i are required to account for any use of T3 Tietze transformations performed based on Theorem 2.2.4 (that

the concatenation of the labels on the walk defined by y_i are equal to y_i).

The set of relators found in this way often results in more relators than are required to define a presentation for H. We exclude some of the obviously redundant relators found, by simply not adding them to the list of relators. The concatenation of the labels on the closed walk defined by a relator from some coset may reduce to the identity, and thus does not need to be added. Finally some relators may be cyclic rearrangements of earlier relators or their inverses, and these too can be ignored.

The presentation given may be very long, in which case application of the MAGMA function SimplifyLength to H can be helpful.

To convert a word w given in the generators of G to a word w' in the generators of H, take the concatenation of the labels of the edges in the walk defined by w from coset 1. If this walk is not closed, then $w \notin H$, and in that case no word w' is possible.

3.2 Application

For a given orientably-regular map of type $\{p, q\}$, let D be the associated ordinary triangle group, that is, the group given by the presentation

$$D = \langle\, R, S \,|\, (RS)^2, R^p, S^q \,\rangle.$$

Let N be the torsion-free normal subgroup of D such that the rotation group of the map is isomorphic to D/N. We use Conder's list of regular orientable maps and list of orientably-regular but chiral maps to find nice generating sets for N for such maps of genus 2 to 12.

Recall that such generating sets (and the effect of the generators of the parent group on them) can be used to determine important aspects of the structure of the fundamental group; see for example [3, 6, 9, 29, 26].

For each such map of genus `g`, we let `G` be the associated ordinary triangle group, let `ind` be the order of the rotation group and let `r` be the set of defining relators (as given in Conder's list) which do not contain the letter T (indicating an orientation-reversing automorphism for regular maps) and are not relators of `G`. This set `r` is then a set of defining relators for the rotation group of the map.

Let the set `niceGs` be the set of generators found. To establish that the set `niceGs` generates the intended subgroup we show that the subgroup generated is normal, torsion-free, of the correct index and

contains the required elements. To be considered 'nice' the set must have the minimum possible number of generators. The set should be comprised of conjugates of elements of `r` by elements of `G` if possible, and the conjugates of the elements of `niceGs` by the generators of `G` as words in the elements of `niceGs` should be reasonably short.

The set `niceGs` is found by trial and error. In most instances experimenting with different sets of conjugates (by words in the generators of `G`) of the elements in r results in a set of nice generators relatively easily. If this proves to be more difficult, a generating set (although probably not a nice one) may be found by using the LowIndexNormalSubgroups algorithm. Then we experiment with sets involving one or more of these generators along with various conjugates of elements of `r`.

To show that the subgroup generated by `niceGs` is normal, we may simply use the MAGMA function `IsNormal` or show that the conjugate of each element in `niceGs` by each of the generators of `G` is in the subgroup. The latter can be shown using the labelled Schreier coset graph of the subgroup generated by `niceGs` in the group `G`.

To show that the subgroup generated by `niceGs` is torsion-free we consider the action of `G` on the cosets of this subgroup by multiplication. If the permutations induced by RS R and S have orders 2, p, and q, respectively, then the subgroup is torsion-free. (Note that those permutations will be semi-regular, because the given subgroup is normal in `G`.)

The subgroup generated by `niceGs` must contain all the elements of `r`, since these are the relators of the rotation group. We try to use the elements of `r` to build the set, so often they appear in `niceGs`. If there are elements of `r` which are not in `niceGs` then we can use the labelled Schreier coset graph to establish their inclusion. The supplementary function `getLabs(L,G, niceGs, r)` gives the words for the elements of `r` in the elements of `niceGs` as obtained from the labelled Schreier coset graph `L`. Note that the given `G` and `niceGs` must be the same as those used by `CosGraphLabs` to obtain `L`.

The index required can be calculated from the order of the full automorphism group of the map, which is given in Conder's list. If the map is regular the rotation group has index 2 in the full automorphism group, and hence has order equal to half the order of the full automorphism group. If the map is chiral then the full automorphism group is the rotation group, and hence the order is the same. The index of the subgroup generated by `niceGs` must be equal to the order of the rotation group.

Recall that the minimum possible number of generators is `2*g`, that is, twice the genus of the corresponding map.

The creation of the labelled Schreier coset graph enables us to ascertain if the set `niceGs` will give rise to reasonably short words (in the elements of `niceGs` for conjugates of the elements of `niceGs` by the generators of `G`. The longer the algorithm `CosGraphLabs` takes to complete the more likely it is that the results will contain longer words.

Hence, if the algorithm takes more than a few seconds, we try a new set of generators. Once the algorithm completes, the required words are easily obtained using the supplementary function `getConjLabs(L,G, niceGs, s)`, where `L` is the labels as returned by `CosGrapgLabs` and `s` is the set of elements of `G` by which we wish to conjugate the elements of `niceGs`. Note that the given `G` and `niceGs` must be the same as those used by `CosGraphLabs` to obtain `L`.

To perform the tasks in MAGMA we first run the following tests on the set `niceGs`:

```
ind eq Index(G, sub<G|niceGs>); //correct index
IsNormal(G, sub<G|niceGs>); //normal
f,Q:=CosetAction(G, sub<G|niceGs>);
[Order(f(w)): w in [R*S, R, S]] eq [2,p,q]; //torsion-free
#niceGs eq 2*g; //correct number of generators
```

If each of these tests return true then we find the required labels, using the following code:

```
L:=CosGraphLabs(G, niceGs);//graph labels
getConjLabs(L,G, niceGs, [R,S,R*S]);//conjugate labels
getLabs(L,G, niceGs, r);//defining relator labels
```

If the function `CosGraphLabs` takes more than a few seconds we try a different set of `niceGs`. If the function `CosGraphLabs` completes in reasonable time and if each of the labels returned by `getConjLabs` and `getLabs` are no more than a few lines long then we declare the set

`niceGs` a nice generating set.

Finally, we note that the results of the computations we have carried out for the maps of genus 2 to 12 can be checked independently in a least two different ways, as follows:

1. Adding the conjugacy relations to the canonical presentation for the triangle group and the using the `Simplify` function to give a group that is verifiably isomorphic to the given triangle group using the `SearchForIsomorphism` function, and/or

2. Verifying that the conjugacy relations are satisfied in all finite quotients of the given triangle group of up to a specified order, and/or in all transitive permutation representations of the given triangle group on up to a specified number of points.

3.3 Results

In the following tables we give the nice generating sets that we found for each regular orientable map of genus 2 and 3, as given in Conder's list of regular orientable maps (see `www.math.auckland.ac.nz/~conder`). For each such map, in the first row of the corresponding table, we list the following:

- the 'name', as given in Conder's list,
- the 'type' of the map, given in the form $\{p, q\}$, where p is the order of R and q is the order of S (while RS has order 2),
- the 'order' of the full automorphism group of the map, and
- the 'index' of the normal subgroup N in the ordinary triangle group `G`.

The subsequent rows correspond to the generators in the nice generating set for N. For each generator we list the following:

- the 'generator', as a word in R and S and the label we have given it, in the form u_i,
- the 'R-conjugate' of this generator u_i,
- the 'S-conjugate' of u_i, and
- the 'RS-conjugate' of u_i.

R2.1	Type $\{3, 8\}$	Order 96	Index 48
Generator	R-conjugate	S-conjugate	RS-conjugate
$(RS^{-3})^2 = u_1$	u_4	u_2	u_1^{-1}
$(S^{-1}RS^{-2})^2 = u_2$	$u_3^{-1}u_4$	u_3	$u_4^{-1}u_1^{-1}$
$(S^{-2}RS^{-1})^2 = u_3$	$u_3^{-1}u_2u_1^{-1}$	u_4	$u_4^{-1}u_3u_2^{-1}$
$(S^{-3}R)^2 = u_4$	$u_4^{-1}u_1^{-1}$	u_1^{-1}	$u_1u_2^{-1}$

Table 3.3: Nice generating set for R2.1

R2.2	Type $\{4, 6\}$	Order 48	Index 24
Generator	R-conjugate	S-conjugate	RS-conjugate
$(RS^{-1})^2 = u_1$	u_2	u_2	u_3
$(S^{-1}R)^2 = u_2$	u_1^{-1}	u_3	u_2^{-1}
$(S^{-2}RS)^2 = u_3$	$u_2^{-1}u_4^{-1}$	u_4	u_1
$(S^{-3}RS^2)^2 = u_4$	$u_2^{-1}u_3u_2u_1$	$u_1^{-1}u_3^{-1}$	$u_1u_4u_2u_1^{-1}$

Table 3.4: Nice generating set for R2.2

R2.3	Type $\{4, 8\}$	Order 32	Index 16
Generator	R-conjugate	S-conjugate	RS-conjugate
$S^{-2}R^2S^{-2} = u_1$	$u_2^{-1}u_1u_2^{-1}u_3u_4^{-1}u_1^{-1}$ u_2	u_2	$u_3^{-1}u_1u_4$
$S^{-3}R^2S^{-1} = u_2$	$u_2^{-1}u_1u_3$	u_3	$u_3^{-1}u_2u_4$
$S^{-4}R^2 = u_3$	$u_2^{-1}u_1u_4$	u_4	$u_4^{-1}u_1^{-1}u_2u_3^{-1}u_4$
$S^{-5}R^2S = u_4$	$u_2^{-1}u_1u_4u_3^{-1}u_2$	u_1^{-1}	$u_4^{-1}u_1^{-1}u_2$

Table 3.5: Nice generating set for R2.3

R2.4	Type $\{5, 10\}$	Order 20	Index 10
Generator	R-conjugate	S-conjugate	RS-conjugate
$SR^2S = u_1$	$u_2u_4u_3^{-1}u_1^{-1}$	u_2	u_1^{-1}
$R^2S^2 = u_2$	u_1^{-1}	u_3	u_2^{-1}
$S^{-1}R^2S^3 = u_3$	$u_1u_2^{-1}u_1^{-1}$	u_4	$u_2u_4u_3^{-1}u_1^{-1}u_4^{-1}u_2^{-1}$ u_1
$S^{-2}R^2S^4 = u_4$	$u_1u_2u_4u_3^{-1}u_1^{-1}u_4^{-1}$ u_2^{-1}	$u_3^{-1}u_1^{-1}u_2u_4$	$u_2u_1^{-1}u_4^{-1}u_2^{-1}u_1$

Table 3.6: Nice generating set for R2.4

R2.5	Type $\{6, 6\}$	Order 24	Index 12
Generator	R-conjugate	S-conjugate	RS-conjugate
$SR^2S = u_1$	$u_1u_2u_3u_4u_3^{-1}u_1^{-1}$	u_2	u_1^{-1}
$R^2S^2 = u_2$	u_1^{-1}	u_3	u_2^{-1}
$S^{-1}R^2S^3 = u_3$	$u_1u_2^{-1}u_1^{-1}$	u_4	$u_2u_3^{-1}u_2^{-1}$
$S^{-2}R^2S^4 = u_4$	$u_1u_2u_3^{-1}u_2^{-1}u_1^{-1}$	$u_3^{-1}u_1^{-1}$	$u_2u_3u_4^{-1}u_3^{-1}u_2^{-1}$

Table 3.7: Nice generating set for R2.5

R2.6	Type $\{8, 8\}$	Order 16	Index 8
Generator	R-conjugate	S-conjugate	RS-conjugate
$R^3S^{-1} = u_1$	u_2	u_4	$u_4u_2^{-1}u_3u_4^{-1}u_1^{-1}u_2u_3^{-1}$
$R^2S^{-1}R = u_2$	u_3	$u_4u_2^{-1}u_3u_4^{-1}u_1^{-1}u_2u_3^{-1}$	$u_4u_2^{-1}u_4^{-1}$
$RS^{-1}R^2 = u_3$	u_4	$u_4u_2^{-1}u_4^{-1}$	$u_4u_3^{-1}u_4^{-1}$
$S^{-1}R^3 = u_4$	$u_2^{-1}u_3u_4^{-1}u_1^{-1}u_2u_3{}^{-1}u_4$	$u_4u_3^{-1}u_4^{-1}$	u_4^{-1}

Table 3.8: Nice generating set for R2.6

R3.1	Type $\{3, 7\}$	Order 336	Index 168
Generator	R-conjugate	S-conjugate	RS-conjugate
$(RS^{-2})^4 = u_1$	u_3	u_2	u_4
$(S^{-1}RS^{-1})^4 = u_2$	u_1^{-1}	u_3	u_2^{-1}
$(S^{-2}R)^4 = u_3$	u_2^{-1}	u_4	u_3^{-1}
$(S^{-3}RS)^4 = u_4$	$u_2^{-1}u_4^{-1}u_6^{-1}u_1^{-1}u_3^{-1}u_5^{-1}$	u_5	u_1
$(S^{-4}RS^2)^4 = u_5$	$u_3^{-1}u_6^{-1}$	u_6	$u_1u_5u_2u_6u_3$
$(S^{-5}RS^3)^4 = u_6$	$u_1u_6u_4u_2$	$u_2^{-1}u_4^{-1}u_6^{-1}u_1^{-1}u_3^{-1}u_5^{-1}$	$u_4^{-1}u_6^{-1}u_1^{-1}$

Table 3.9: Nice generating set for R3.1

R3.2	Type$\{3,8\}$	Order 192	Index 96
Generator	R-conjugate	S-conjugate	RS-conjugate
$(SR^{-1}S)^3 = u_1$	$u_2u_4u_6$	u_2	u_1^{-1}
$(R^{-1}S^2)^3 = u_2$	u_1^{-1}	u_3	u_2^{-1}
$(S^{-1}R^{-1}S^3)^3 = u_3$	$u_5^{-1}u_3^{-1}u_1^{-1}$	u_4	$u_6^{-1}u_4^{-1}u_2^{-1}$
$(S^{-2}R^{-1}S^4)^3 = u_4$	$u_1u_4u_5^{-1}u_3^{-1}u_1^{-1}$	u_5	$u_1u_3u_5u_4^{-1}u_1^{-1}$ $u_6^{-1}u_3^{-1}$
$(S^{-3}R^{-1}S^5)^3 = u_5$	$u_1u_3u_2^{-1}$	u_6	$u_2u_4u_3^{-1}$
$(S^{-4}R^{-1}S^6)^3 = u_6$	$u_1u_3u_5u_4^{-1}u_2^{-1}$	$u_5^{-1}u_3^{-1}u_1^{-1}$	$u_2u_4u_6u_5^{-1}u_3^{-1}$

Table 3.10: Nice generating set for R3.2

R3.3	Type $\{3,12\}$	Order 96	Index 48
Generator	R-conjugate	S-conjugate	RS-conjugate
$SRS^{-2}RS^3 = u_1$	$u_2u_1^{-1}$	u_2	$u_3u_2^{-1}$
$RS^{-2}RS^4 = u_2$	u_1^{-1}	u_3	u_2^{-1}
$S^{-1}RS^{-2}RS^5 = u_3$	$u_1u_5u_6^{-1}u_3^{-1}u_1^{-1}$	u_4	$u_1u_2^{-1}$
$S^{-2}RS^{-2}RS^6 = u_4$	$u_1u_5u_4^{-1}u_1^{-1}$	u_5	$u_2u_6u_5^{-1}u_2^{-1}$
$S^{-3}RS^{-2}RS^7 = u_5$	$u_1u_4^{-1}u_1^{-1}$	u_6	$u_2u_5^{-1}u_2^{-1}$
$S^{-4}RS^{-2}RS^8 = u_6$	$u_1u_3u_4^{-1}u_1^{-1}$	$u_4^{-1}u_1^{-1}u_2u_6$	$u_2u_4u_5^{-1}u_2^{-1}$

Table 3.11: Nice generating set for R3.3

R3.4	Type $\{4, 6\}$	Order 96	Index 48
Generator	R-conjugate	S-conjugate	RS-conjugate
$(RS^{-2})^2 = u_1$	u_3	u_2	u_1^{-1}
$(S^{-1}RS^{-1})^2 = u_2$	$u_2^{-1}u_6u_3$	u_3	$u_4^{-1}u_3^{-1}u_1^{-1}$
$(S^{-2}R)^2 = u_3$	u_4	u_1^{-1}	u_5
$(R^{-1}S^{-2}R^2)^2 = u_4$	$u_4^{-1}u_3^{-1}u_1^{-1}$	u_5	$u_5^{-1}u_1u_2^{-1}$
$(RS^{-1}R^2S)^2 = u_5$	u_2	u_6	u_3
$(S^{-1}RS^{-1}R^2S^2)^2 = u_6$	$u_2^{-1}u_5^{-1}u_4^{-1}$	$u_3u_4^{-1}u_3^{-1}$	$u_3^{-1}u_6^{-1}u_5^{-1}$

Table 3.12: Nice generating set for R3.4

R3.5	Type $\{4, 8\}$	Order 64	Index 32
Generator	R-conjugate	S-conjugate	RS-conjugate
$SRS^{-1}RS^2 = u_1$	$u_1u_3u_5u_4^{-1}u_1^{-1}$	u_2	$u_2u_4u_6u_5^{-1}u_2^{-1}$
$RS^{-1}RS^3 = u_2$	u_1^{-1}	u_3	u_2^{-1}
$S^{-1}RS^{-1}RS^4 = u_3$	$u_6^{-1}u_3^{-1}u_1^{-1}$	u_4	$u_1u_3u_5u_4^{-1}u_2^{-1}$
$S^{-2}RS^{-1}RS^5 = u_4$	$u_1u_3^{-1}u_1^{-1}$	u_5	$u_2u_4^{-1}u_2^{-1}$
$S^{-3}RS^{-1}RS^6 = u_5$	$u_1u_3u_6u_1u_4u_5^{-1}u_3^{-1}$ $u_1^{-1}u_2u_3^{-1}u_1^{-1}$	u_6	$u_2u_1^{-1}u_6^{-1}u_4^{-1}u_2^{-1}$
$S^{-4}RS^{-1}RS^7 = u_6$	$u_1u_3u_5^{-1}u_3^{-1}u_1^{-1}$	$u_5^{-1}u_3^{-1}u_1^{-1}$	$u_2u_4u_6^{-1}u_4^{-1}u_2^{-1}$

Table 3.13: Nice generating set for R3.5

R3.6	Type $\{4, 8\}$	Order 64	Index 32
Generator	R-conjugate	S-conjugate	RS-conjugate
$(RS^{-1})^2 = u_1$	u_2	u_2	u_3
$(S^{-1}R)^2 = u_2$	u_1^{-1}	u_3	u_2^{-1}
$(S^{-2}RS)^2 = u_3$	$u_2^{-1}u_4^{-1}u_6^{-1}$	u_4	u_1
$(S^{-3}RS^2)^2 = u_4$	$u_2^{-1}u_4^{-1}u_5u_4u_3$ u_2u_1	u_5	$u_1u_6u_4u_2u_1^{-1}$
$(S^{-4}RS^3)^2 = u_5$	$u_2^{-1}u_4^{-1}u_6u_4u_2$	u_6	$u_3^{-1}u_5^{-1}u_1^{-1}$
$(S^{-5}RS^4)^2 = u_6$	$u_2^{-1}u_4^{-1}u_5^{-1}u_4u_2$	$u_1^{-1}u_3^{-1}u_5^{-1}$	$u_3^{-1}u_5^{-1}u_6^{-1}u_5u_3$

Table 3.14: Nice generating set for R3.6

R3.7	Type $\{4, 12\}$	Order 48	Index 24
Generator	R-conjugate	S-conjugate	RS-conjugate
$S^{-3}R^2S^{-3} = u_1$	$u_4^{-1}u_5u_6^{-1}u_1^{-1}u_2u_3^{-1}$ $u_4u_5^{-1}u_6u_1u_6u_5^{-1}u_4$	u_2	$u_4^{-1}u_3u_2^{-1}u_3u_4^{-1}u_5$ $u_6^{-1}u_1^{-1}u_2u_3^{-1}u_4$
$S^{-4}R^2S^{-2} = u_2$	$u_4^{-1}u_5u_6^{-1}u_1^{-1}u_2$ $u_3^{-1}u_4u_5^{-1}u_6u_5^{-1}u_4$	u_3	$u_4^{-1}u_3u_2^{-1}u_1u_2^{-1}u_3$ $u_4^{-1}u_5u_6^{-1}u_1^{-1}u_2$ $u_3^{-1}u_4$
$S^{-5}R^2S^{-1} = u_3$	$u_4^{-1}u_5u_6^{-1}u_1^{-1}u_2u_3^{-1}$ $u_4u_5^{-1}u_4$	u_4	$u_4^{-1}u_3u_2^{-1}u_1u_5$
$S^{-6}R^2 = u_4$	$u_4^{-1}u_5u_6^{-1}u_1^{-1}u_2$ $u_3^{-1}u_4$	u_5	$u_4^{-1}u_3u_2^{-1}u_1u_6$
$S^{-7}R^2S = u_5$	$u_4^{-1}u_5u_6^{-1}u_1^{-1}u_2$	u_6	$u_4^{-1}u_3u_2^{-1}u_1u_6u_5^{-1}$ u_4
$S^{-8}R^2S^2 = u_6$	$u_4^{-1}u_5u_6^{-1}u_1^{-1}u_3$	$u_2^{-1}u_3u_4^{-1}u_5u_6^{-1}$ $u_1^{-1}u_2u_3^{-1}u_4u_5^{-1}u_6$	$u_4^{-1}u_3u_2^{-1}u_1u_6u_5^{-1}$ $u_4u_3^{-1}u_4$

Table 3.15: Nice generating set R3.7

R3.8	Type $\{6,6\}$	Order 48	Index 24
Generator	R-conjugate	S-conjugate	RS-conjugate
$S^{-1}R^3S^{-2} = u_1$	u_3	u_2	u_4
$S^{-2}R^3S^{-1} = u_2$	$u_2^{-1}u_5^{-1}u_4^{-1}$	$u_2u_3u_2^{-1}u_5^{-1}u_4^{-1}u_6$ u_4	$u_4^{-1}u_6^{-1}u_4$
$R^{-1}S^{-1}R^3S^{-2}R = u_3$	u_6	u_4	$u_4^{-1}u_3^{-1}u_1^{-1}$
$R^4S^{-2}RS = u_4$	$u_4^{-1}u_6^{-1}u_4u_5u_2u_3^{-1}$ u_2^{-1}	u_5	u_1
$S^{-1}R^4S^{-2}RS^2 = u_5$	$u_1^{-1}u_5u_2$	$u_2u_3^{-1}u_2^{-1}$	$u_2^{-1}u_5^{-1}u_4^{-1}u_6u_4$
$R^{-2}S^{-1}R^3S^{-2}R^2 = u_6$	$u_4^{-1}u_6^{-1}u_4u_5u_2u_3^{-1}$ $u_2^{-1}u_1^{-1}u_2u_3u_2^{-1}u_5^{-1}$ $u_4^{-1}u_6u_4$	$u_4^{-1}u_3^{-1}u_1^{-1}$	$u_1u_2^{-1}u_1^{-1}$

Table 3.16: Nice generating set for R3.8

R3.9	Type $\{7,14\}$	Order 28	Index 14
Generator	R-conjugate	S-conjugate	RS-conjugate
$SR^2S = u_1$	$u_2u_4u_6u_5^{-1}u_3^{-1}u_1^{-1}$	u_2	u_1^{-1}
$R^2S^2 = u_2$	u_1^{-1}	u_3	u_2^{-1}
$S^{-1}R^2S^3 = u_3$	$u_1u_2^{-1}u_1^{-1}$	u_4	$u_2u_3^{-1}u_2^{-1}$
$S^{-2}R^2S^4 = u_4$	$u_1u_2u_3^{-1}u_2^{-1}u_1^{-1}$	u_5	$u_2u_3u_4^{-1}u_3^{-1}u_2^{-1}$
$S^{-3}R^2S^5 = u_5$	$u_1u_2u_3u_4^{-1}u_3^{-1}u_2^{-1}$ u_1^{-1}	u_6	$u_2u_3u_4u_5^{-1}u_4^{-1}u_3^{-1}$ u_2^{-1}
$S^{-4}R^2S^6 = u_6$	$u_1u_2u_3u_4u_5^{-1}u_4^{-1}$ $u_3^{-1}u_2^{-1}u_1^{-1}$	$u_5^{-1}u_3^{-1}u_1^{-1}u_2u_4u_6$	$u_2u_3u_4u_3^{-1}u_1^{-1}u_6^{-1}$ $u_4^{-1}u_2^{-1}u_1$

Table 3.17: Nice generating set for R3.9

3.10	Type$\{8,8\}$	Order 32	Index 16
Generator	R-conjugate	S-conjugate	RS-conjugate
$S^{-1}R^2S^{-1}=u_1$	$u_1^{-1}u_3^{-1}u_2$	u_2	$u_2^{-1}u_4^{-1}u_3$
$S^{-2}R^2=u_2$	$u_1^{-1}u_3^{-1}u_5^{-1}u_4u_2$	u_3	$u_2^{-1}u_4^{-1}u_6^{-1}u_5u_3$
$S^{-3}R^2S=u_3$	$u_1^{-1}u_3^{-1}u_5^{-1}u_4u_1$	u_4	$u_2^{-1}u_4^{-1}u_6^{-1}u_5u_2$
$S^{-4}R^2S^2=u_4$	$u_1^{-1}u_3^{-1}u_5^{-1}u_4u_1$ $u_6u_5u_3u_1$	u_5	$u_2^{-1}u_4^{-1}u_6^{-1}u_5u_2$ $u_1^{-1}u_3^{-1}u_5^{-1}u_6u_4u_2$
$S^{-5}R^2S^3=u_5$	$u_1^{-1}u_3^{-1}u_5^{-1}u_4u_1$ $u_6u_3u_4^{-1}u_5u_3u_1$	u_6	$u_2^{-1}u_4^{-1}u_6^{-1}u_5u_2$ $u_1^{-1}u_3^{-1}u_5^{-1}u_4u_5^{-1}$ $u_6u_4u_2$
$S^{-6}R^2S^4=u_6$	$u_1^{-1}u_3^{-1}u_5^{-1}u_4u_1u_6$ $u_3u_2^{-1}u_4^{-1}u_6^{-1}u_1^{-1}$ $u_4^{-1}u_5u_3u_1$	$u_1^{-1}u_3^{-1}u_5^{-1}$	$u_2^{-1}u_4^{-1}u_6^{-1}u_5u_2$ $u_1^{-1}u_3^{-1}u_5^{-1}u_4u_1$ $u_2^{-1}u_5^{-1}u_6u_4u_2$

Table 3.18: Nice generating set for R3.10

R3.11	Type $\{8,8\}$	Order 32	Index 16
Generator	R-conjugate	S-conjugate	RS-conjugate
$SR^2S=u_1$	$u_1u_2u_3u_4u_5u_6u_5^{-1}$ $u_3^{-1}u_1^{-1}$	u_2	u_1^{-1}
$R^2S^2=u_2$	u_1^{-1}	u_3	u_2^{-1}
$S^{-1}R^2S^3=u_3$	$u_1u_2^{-1}u_1^{-1}$	u_4	$u_2u_3^{-1}u_2^{-1}$
$S^{-2}R^2S^4=u_4$	$u_1u_2u_3^{-1}u_2^{-1}u_1^{-1}$	u_5	$u_2u_3u_4^{-1}u_3^{-1}u_2^{-1}$
$S^{-3}R^2S^5=u_5$	$u_1u_2u_3u_4^{-1}u_3^{-1}$ $u_2^{-1}u_1^{-1}$	u_6	$u_2u_3u_4u_5^{-1}u_4^{-1}$ $u_3^{-1}u_2^{-1}$
$S^{-4}R^2S^6=u_6$	$u_1u_2u_3u_4u_5^{-1}u_4^{-1}$ $u_3^{-1}u_2^{-1}u_1^{-1}$	$u_5^{-1}u_3^{-1}u_1^{-1}$	$u_2u_3u_4u_5u_6^{-1}u_5^{-1}$ $u_4^{-1}u_3^{-1}u_2^{-1}$

Table 3.19: Nice generating set for R3.11

R3.12	Type $\{12, 12\}$	Order 24	Index 12
Generator	R-conjugate	S-conjugate	RS-conjugate
$R^5S^{-1} = u_1$	$u_2^{-1}u_3^{-1}u_4^{-1}u_5^{-1}u_6$ $u_5u_4u_3u_2$	u_2	$u_2u_1^{-1}u_2^{-1}$
$S^{-1}R^5 = u_2$	u_1^{-1}	u_3	u_2^{-1}
$S^{-2}R^5S = u_3$	$u_1^{-1}u_2^{-1}u_1$	u_4	$u_2^{-1}u_3^{-1}u_2$
$S^{-3}R^5S^2 = u_4$	$u_1^{-1}u_2^{-2}u_3^{-1}u_2^2u_1$	u_5	$u_2^{-1}u_3^{-2}u_4^{-1}u_3^2u_2$
$S^{-4}R^5S^3 = u_5$	$u_1^{-1}u_2^{-2}u_3^{-2}u_4^{-1}u_3^2$ $u_2^2u_1$	u_6	$u_2u_1u_2^{-1}u_3^{-1}u_4^{-1}u_5^{-1}$ $u_6^{-1}u_1^{-1}u_2^{-1}u_3^{-1}u_4^{-1}$ $u_5^{-1}u_6u_5u_4^2u_3^2u_2$
$S^{-5}R^5S^4 = u_6$	$u_2^{-1}u_3^{-1}u_4^{-1}u_5^{-1}u_6^{-1}$ $u_1^{-1}u_2^{-1}u_3^{-1}u_4^{-1}u_5^{-1}$ $u_6u_5u_4^2u_3^2u_2^2u_1$	$u_6u_5u_4u_3u_2u_1^{-1}$ $u_2^{-1}u_3^{-1}u_4^{-1}u_5^{-1}u_6^{-1}$	$u_2u_1u_2^{-1}u_3^{-1}u_4^{-1}u_5^{-1}$ $u_6^{-1}u_1^{-1}u_2^{-1}u_3^{-1}u_4^{-1}$ $u_5^{-1}u_6^{-1}u_5u_4u_3u_2u_1$ $u_6u_5u_4u_3u_2u_1^{-1}u_2^{-1}$

Table 3.20: Nice generating set for R3.12

The results for all regular orientable maps and orientably-regular (but chiral) maps with genus from 2 to 12 can be found at `github.com/GLiversidge/CosGraphLabs`.

3.4 Comparable algorithms

3.4.1 Rewriting functionality

In order to determine isomorphisms among the torsion-free subgroups of $[p, q, r]$ Coxeter groups found using `LowIndAlg`, as will be explained in Chapter 5, we needed to find presentations for those subgroups. For this reason, we compared the functionality of `CosGraphLabs` against that of the MAGMA function `Rewrite`, by measuring the time taken for finding such presentations, and more specifically for finding presentations for representatives of the conjugacy classes of the minimum index torsion-free subgroups of the Coxeter groups [3,3,6], [3,4,4], [3,5,3], [3,6,3], [4,3,4], [4,3,6], [4,4,4] and [6,3,6]. We did not compare times for rewriting the minimum index torsion-free subgroups of the remaining Coxeter groups [4,3,5], [5,3,5], [5,3,6] simply because the MAGMA version was taking far too long for those.

Note that there are two versions of `Rewrite` available in MAGMA, differentiated only by their inputs. The capability of `CosGraphLabs` to rewrite a subgroup with specified generators is most similar to the call `Rewrite(G,~H)`, and so it is this version we have used in our time comparisons. The other version of the MAGMA function is obtained with the call `Rewrite(G,H)`. The latter version is significantly faster, but often results in a presentation on a set of generators different from those given.

CosGraphLabs was consistently faster or as fast as the Rewrite function. The maximum time taken to rewrite the subgroups for any one Coxeter group was 0.34 seconds for CosGraphLabs for the 11 subgroups of the group [4,3,6]. In stark contrast, the maximum time for Rewrite was 2272.42 seconds for the 7 subgroups of the Coxeter group [3,5,3].

We created the following procedure to obtain time data for the Rewrite procedure.

```
test_rewrite:=procedure(G, L)
    start:=Realtime();
    start1:=start;
    for Hgens in L do
        H:=sub<G|Hgens>;
        Rewrite(G,~H);
        print H;
        print "time taken=", Realtime()-start1;
        start1:=Realtime();
    end for;
    print "time taken=", Realtime()-start;
end procedure;
```

We used the following procedure to obtain time data for rewriting via `CosGraphLabs`.

```
test_CGL1:=procedure(G, L)
    start:=Realtime();
    for Hgens in L do
        reWrite2(G,Hgens);
    end for;
    print "time taken=", Realtime()-start;
end procedure;
```

Time Taken			
Coxeter Group	Number of subgroups	`Rewrite`	`CosGraphLabs`
3,3,6	1	0.13	0.02
3,4,4	13	7.21	0.24
3,5,3	7	2272.42	0.33
3,6,3	2	0.02	0.02
4,3,4	2	0.02	0.02
4,3,6	11	6.28	0.34
4,4,4	12	0.31	0.07
6,3,6	12	8.61	0.14
Total	60	2295.00	1.18

Table 3.21: Comparison of times for rewriting a subgroup on specified generators.

3.4.2 Functionality for obtaining words

We compared the functionality of `CosGraphLabs` for obtaining words against the MAGMA function `!` by measuring the time taken and the length of words found in the application described in Section 3.2. As can be seen in Table 3.22, the maximum time taken by `CosGraphLabs` to find the required information for any map in the table was 0.24 seconds (namely for the map R10.5), while in stark contrast the maximum time taken by the MAGMA function `!` was 4032.69 seconds (namely for the map R12.2). The average length of all words found for each map was comparable: 444 for `!`, and 454 for `CosGraphLabs`.

We created the following procedure to obtain time and length data for `!`:

```
mag_test:=procedure(G,niceGs,R, S, r)
        start:=Realtime();
        sum:=0; max:=1; n:=0;
        H:=sub<G| niceGs>;
        for g in niceGs do
            n:=#(H ! g);
            if n gt max then
                max:=n;
            end if;
            sum+:=n;
            n:=#(H ! (g^S));
            if n gt max then
                max:=n;
```

```
            end if;
            sum+:=n;
            n:=#(H ! (g^R));
            if n gt max then
                max:=n;
            end if;
            sum+:=n;
            n:=#(H ! (g^(R*S)));
            if n gt max then
                max:=n;
            end if;
            sum+:=n;
        end for;
        for g in r do
            n:=#(H ! g);
            if n gt max then
                max:=n;
            end if;
            sum+:=n;
        end for;
        print "max length:", max;
        print "total length:", sum;
        print "time taken", Realtime()-start;
    end procedure;
```

To obtain time and length data for the same process using `CosGraphLabs` we added print statements for time and length to `CosgraphLabs` and

supplementary procedures and used the following code.

```
L:=CosGraphLabs(G, niceGs);
    getConjLabs(L,G, niceGs, [R,S,R*S]);
    getLabs(L,G, niceGs, r);
```

Map	!		CosGraphLabs	
Name	Total length	Time taken	Total length	Time taken
R2.1	25	0	21	0.01
R2.2	25	0	21	0.01
R2.3	47	0	39	0
R2.4	26	0	38	0
R2.5	36	0	32	0
R2.6	39	0	41	0
R3.1	49	0.01	39	0.02
R3.2	53	0	49	0.1
R3.3	51	0	45	0.01
R3.4	39	0	33	0
R3.5	67	0	61	0
R3.6	53	0	47	0
R3.7	111	0	119	0
R3.8	71	0	63	0.01
R3.9	76	0	70	0
R3.10	115	0	109	0
R3.11	76	0	70	0.01
R3.12	215	0.01	129	0

Map	!		CosGraphLabs	
Name	Total Length	Time taken	Total Length	Time taken
R4.1	83	0	67	0.02
R4.2	88	0.01	82	0
R4.3	73	0	65	0.02
R4.4	90	0.01	82	0.01
R4.5	167	0	155	0.01
R4.6	131	0.01	127	0.01
R4.7	64	0	56	0.01
R4.8	195	0.01	163	0.01
R4.9	150	0.01	130	0.01
R4.10	132	0	170	0.01
R4.11	132	0.01	124	0
R4.12	455	0.08	255	0.03
R5.1	135	0	163	0.04
R5.2	77	0	67	0.02
R5.3	67	0.02	59	0.03
R5.4	122	0.01	100	0.02
R5.5	160	0	178	0.01
R5.6	193	0.01	133	0.01
R5.7	135	0.04	125	0.01
R5.8	191	0.01	167	0.01
R5.9	77	0.01	89	0.01
R5.10	118	0	108	0.01
R5.11	310	0.01	314	0.01

Map	!		CosGraphLabs	
Name	Total Length	Time taken	Total Length	Time taken
R5.12	100	0.01	90	0
R5.13	104	0	94	0.01
R5.14	204	0.01	194	0.02
R5.15	204	0	194	0
R5.16	236	0	230	0.02
R6.1	229	0	243	0.02
R6.2	93	0.04	81	0.01
R6.3	123	0	109	0.01
R6.4	188	0.56	176	0.01
R6.5	221	0.09	209	0.01
R6.6	145	0	148	0.01
R6.7	141	0	128	0.01
R6.8	160	0	128	0.01
R6.9	226	0	214	0.01
R6.10	416	0	404	0.02
R6.11	292	0	286	0.02
R6.12	298	0	286	0.01
R6.13	344	0.01	334	0.01
R7.1	93	0.02	79	0.05
R7.2	305	0.04	219	0.02
R7.3	297	0.1	337	0.02
R7.4	249	3.86	235	0.01
R7.5	493	0.03	479	0.01

Map	!		CosGraphLabs	
Name	Total Length	Time taken	Total Length	Time taken
R7.6	452	0.38	630	0.04
R7.7	546	0.01	614	0.07
R7.8	543	0	505	0.02
R7.9	396	0.01	388	0.05
R7.10	396	0	382	0.02
R7.11	506	0	462	0.07
R7.12	360	0.01	346	0.04
C7.1	216	0.01	290	0.01
C7.2	189	0	177	0.02
R8.1	137	0.02	121	0.04
R8.2	109	0.01	93	0.02
R8.3	318	39.38	302	0.01
R8.4	677	0.03	673	0.02
R8.5	229	0.03	213	0.01
R8.6	840	0	654	0.02
R8.7	250	0.05	232	0.01
R8.8	348	0	438	0.04
R8.9	516	0.01	506	0.07
R8.10	516	0	500	0.02
R8.11	490	0.01	682	0.04
C8.1	706	0.02	764	0.02
R9.1	260	0.02	232	0.04
R9.2	147	0.01	129	0.04

Map	!		CosGraphLabs	
Name	Total Length	Time taken	Total Length	Time taken
R9.3	415	0.02	417	0.04
R9.4	270	0.01	266	0.03
R9.5	1283	0.27	1041	0.04
R9.6	427	0.01	441	0.02
R9.7	200	0.01	202	0.02
R9.8	222	0	204	0.02
R9.9	238	1.06	270	0.02
R9.10	443	2.63	469	0.05
R9.11	257	0.01	237	0.01
R9.12	395	439.64	377	0.03
R9.13	813	0.07	795	0.02
R9.14	415	0.02	561	0.04
R9.15	187	0	167	0.02
R9.16	225	0.01	229	0.02
R9.17	489	0.01	471	0.02
R9.18	642	0.01	630	0.02
R9.19	281	0.01	263	0.01
R9.20	371	0	399	0.02
R9.21	422	0.01	528	0.03
R9.22	700	0.01	1040	0.02
R9.23	994	0.02	788	0.07
R9.24	927	0.01	909	0.05
R9.25	889	32.12	783	0.03

Map	!		CosGraphLabs	
Name	Total Length	Time taken	Total Length	Time taken
R9.26	492	0.01	474	0.03
R9.27	592	0.01	574	0.03
R9.28	267	0	249	0.02
R9.29	917	0	1653	0.1
R9.30	652	0.02	640	0.09
R9.31	652	0.01	634	0
R9.32	708	0	698	0.04
R10.1	264	0.02	242	0.04
R10.2	453	0.01	447	0.03
R10.3	521	0.07	441	0.03
R10.4	319	0	303	0.03
R10.5	373	0.01	511	0.24
R10.6	225	0.02	205	0.04
R10.7	275	0.01	279	0.03
R10.8	455	0.01	463	0.04
R10.9	507	0.13	487	0.03
R10.10	523	0.09	967	0.05
R10.11	460	3504.62	460	0.03
R10.12	1045	0.1	1041	0.03
R10.13	369	0	349	0.02
R10.14	1077	0.02	867	0.06
R10.15	510	0.01	476	0.03
R10.16	385	0	365	0.02

Map	!		CosGraphLabs	
Name	Total Length	Time taken	Total Length	Time taken
R10.17	333	0.38	313	0.01
R10.18	1147	82.09	1033	0.16
R10.19	1065	709.58	1089	0.03
R10.20	343	0	349	0.05
R10.21	910	20.37	840	0.05
R10.22	804	0.02	790	0.13
R10.23	804	0.01	784	0.01
R10.24	852	0.01	844	0.05
C10.1	279	0.03	309	0.04
C10.2	965	0.29	795	0.03
C10.3	696	0.02	730	0.04
R11.1	463	0.25	483	0.04
R11.2	443	0	421	0.09
R11.3	49	2737.55	551	0.03
R11.4	1213	0.19	1191	0.03
R11.5	578	0.01	350	0.02
R11.6	364	0	344	0.02
R11.7	473	0	451	0.03
R11.8	891	0.16	687	0.05
R11.9	436	7.67	426	0.02
R11.10	494	0	536	0.02
R11.11	972	0.04	956	0.16
R11.12	972	0.01	950	0.03

Map	!		CosGraphLabs	
Name	Total Length	Time taken	Total Length	Time taken
R11.13	944	0.01	1004	0.08
R11.14	1040	0.01	1028	0.07
C11.1	418	0.03	514	0.03
C11.2	688	0.02	528	0.02
C11.3	1249	0.05	1143	0.1
C11.4	346	0.02	314	0.01
C11.5	890	0.01	778	0.03
C11.6	3253	0.08	3867	0.18
R12.1	407	0.46	381	0.03
R12.2	53	4032.69	650	0.04
R12.3	1493	0.26	1489	0.04
R12.4	453	5.06	429	0.02
R12.5	440	0.05	416	0.02
R12.6	525	0.01	449	0.05
R12.7	744	0.01	720	0.14
R12.8	553	0.01	533	0.02
R12.9	1156	0.05	1138	0.21
R12.10	1156	0.02	1132	0.03
R12.11	1216	0.02	1206	0.08
C12.1	1814	0.06	2182	0.05
C12.2	1457	0.13	2309	0.05

Table 3.22: Time and length comparison data

Chapter 4

PEACEv2

4.1 The PEACEv2 Procedure

PEACEv2 takes a finitely presented group $G = \langle X \mid R \rangle$ and a set $\gamma = \{g_1, \ldots, g_n\}$ of elements of G which is known to generate a subgroup H of finite index in G, and returns a list L of information which is used by the ProveWord algorithm to create proof words. *Proof words* are sequences of generators of G as well as square and round brackets which can be used to show that a word w in the generators of G can be written as an expression in the elements of H.

4.1.1 Definitions

In the algorithm, the generators of G and the cosets of H in G are assigned integer values. A coset may be referenced by more than one integer, with the different integers corresponding to different definitions of the coset. The coset H can always be referenced by the integer 1.

The list L contains the following information:

- T: The coset table for H in G, with $T[c, g]$ being the coset resulting from right multiplication of the coset c by the generator g, and $T[c, g^{-1}]$ being the coset resulting from right multiplication of the coset c by the inverse of the generator g.

- A: The auxiliary table. In PEACEv2, cosets cannot be renamed as they are in many implementations of coset enumeration because the redundant cosets are used in the formation of proof words. There are two consequences of this: the row of the coset table which corresponds to the coset c might not be the cth row, and there might be multiple cosets which are equal to cg for each generator g. These cosets will be equivalent, but will have different definitions. We use an auxiliary table to keep track of information relating to definitions, redundancies and positions in T. More specifically:

  - $A[c, 1]$ gives the row of T which corresponds to the coset c, or 0 if the row has been removed.
  - $A[c, 2]$ is 0 if c is active, or is the integer representing the equivalent coset when c is made redundant.
  - $A[c, 3]$ gives the integer representing the coset from which c was defined or zero if c is coset 1.
  - $A[c, 4]$ gives the integer representing the generator used in the definition of c or zero if c is coset 1.

- P: The proof table. Each time an entry of T is changed, the reason for the change is recorded in P. If $[c, g]$ are the coordinates of the

entry changed, then the reason is recorded in the corresponding entry of P. Note that $P[c, g]$ may contain multiple entries, each corresponding to a different but equivalent coset. The format of entries in P will be described later in this chapter.

- Q: The redundancy proof table. If $A[c, 2] = c_2 \neq 0$, then $Q[c]$ gives the reason for the redundancy $c = c_2$. The format of entries in Q will be described later in this chapter.

- W: The proof word table. Each entry of $W[c, g]$ gives the proof word for $cg = c_2$, for some coset c_2. These entries correspond to the entries in $P[c, g]$.

- V: The redundancy proof word table. If $A[c, 2] = c_2 \neq 0$, then $V[c]$ will give the proof word for the equivalence of the cosets c and c_2.

While building the coset table, `PEACEv2` tracks which subgroup generator or group relator led to each deduction and coincidence, and records this information in P or Q, respectively. The algorithm ProveWord translates this information into proof words, and stores each word found in W and V.

4.1.2 Proof words

The sequence of characters in a proof word can be partitioned into three types of *proof items*.

- *Type 1* proof items are the generators of G which are outside any set of brackets.

- *Type 2* proof items are the sequences of the generators of G enclosed in round brackets. These sequences of generators of G are either an element of the relator set R, the inverse of an element of R, or a cyclic rotation of an element of R or its inverse.

- *Type 3* proof items are the sequence of the generators of G enclosed in square brackets. These sequences are the given generators of H or their inverses.

A deduction about how the given word w can be written as an expression in terms of the generators of H can be obtained from the proof word via the equality of two different methods of simplification, one giving an expression in the generators of G, and the other giving an expression in the generators of the subgroup H.

In the first simplification method s_1, we remove all brackets (but not their contents), so that we have a sequence of type 1 items. Then we remove all consecutive inverse pairs (of the form gg^{-1}), giving a word in the generators of the group G.

In the second method s_2, we remove all type 2 items. We keep type 1 and 3 proof items separate, removing consecutive inverse pairs if they are of the same type. In s_2, any sequence of type 1 proof items between two type 3 items will cancel out. The reason for this may not be immediately obvious so we will explain why in the next paragraph.

Type 1 proof items describe definitions of new cosets, and so we can think of the type 1 proof items as edges of a spanning tree in a

graph of all cosets created (even redundant ones). Type 2 and 3 proof items can be thought of as a loop from a coset to itself, with type 2 at all cosets but type 3 only at coset 1. Hence the type 1 proof items occurring between two type 3 items must describe a walk from coset 1 to itself. If the type 1 proof items did not cancel, it would imply the existence of a cycle in the spanning tree. Thus s_2 will always give some (possibly empty) sequence of type 3 items, which are (possibly) suffixed and (possibly) prefixed by type 1 items.

Example 4.1.1 Figure 4.1 shows a graphical representation of proof items. The points 1, 2, 3 and 4 are the cosets H, Ha, Ha^{-1} and Ha^2 respectively, where a is a generator of G. The blue cycles at each vertex correspond to a relator b of G and the pink cycle at vertex 1 corresponds to a generator c of H.

Suppose $b = a^6$ and $c = a^3$. Then the walk given by the edges $(c, a, a, b, a^{-1}, b^{-1}, a^{-1}c)$ gives the proof word

$$[a^3]aa(a^6)a^{-1}a^{-1}[a^3].$$

Simplification under s_1 gives

$$\begin{array}{cccccccc} & [a^3] & a & a & (a^6) & a^{-1} & a^{-1} & [a^3] \\ = & a^3 & a & a & a^6 & a^{-1} & a^{-1} & a^3 \\ = & a^{12} & & & & & & \end{array}$$

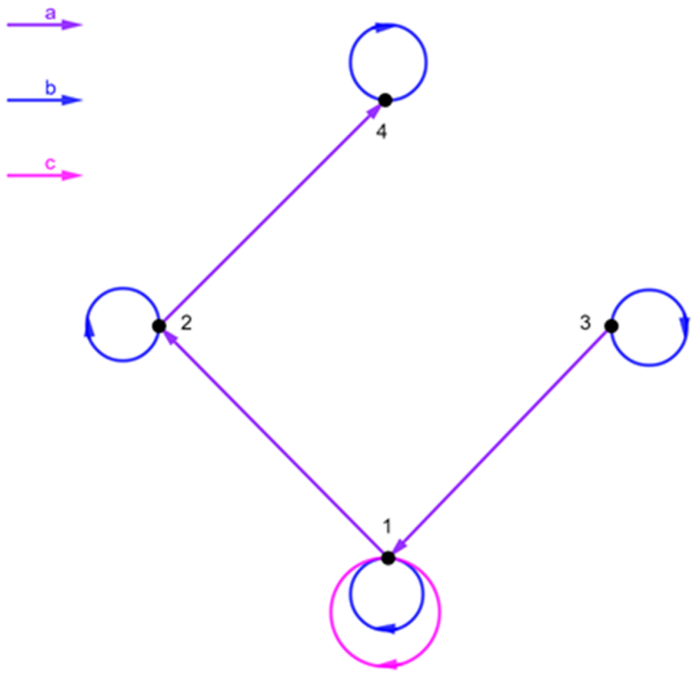

Figure 4.1: Graphical representation of proof items

Simplification under s_2 gives

$$\begin{array}{lllllll} & [a^3] & a & a & (a^6) & a^{-1} & a^{-1} & [a^3] \\ = & [a^3] & a & a & & a^{-1} & a^{-1} & [a^3] \\ = & [a^3][a^3] & & & & & & \end{array}$$

Using these two methods of simplification, we obtain the equality $s_1(w) = s_2(w)$, which can be rearranged to give a word in the generators of H on one side, and a word in the generators of G on the other. For example, the equations in Example 4.1.1 prove that $a^{12} = c^2$. Note that $c^2 = a^6 = 1_G$, so this result is quite trivial and is intended for

illustrative purposes only.

The intended purpose is to prove that a particular word in the generators of G gives an element of the subgroup H. We know from the coset table whether or not this is true, but the proof table of `PEACEv2` gives a justification.

4.1.3 Proof table entries

Every time we add a new entry to the coset table T, we must update the inverse entry, and record both entries in P. A proof table entry consists of two parts. The first is the integer representing the coset that the entry responds to, specifically the integer in the new entry in T. The second is a sequence of coset numbers separated by proof items or equal signs. This sequence tracks the reason for the new entry.

If we define a new coset c_2 as the result of multiplying c_1 by g, then we add proof table entries with an empty sequence to $P[c_1, g]$ and $P[c_2, g^{-1}]$ corresponding to c_2 and c_1 respectively, to signify that this is a definition, and not a deduction.

Suppose that tracing a word w from some coset c results in a single gap at index i, and hence gives the deduction $c_1 g = c_2$, for some cosets c_1 and c_2 and some generator g. The proof table entry of such a deduction is built as follows: First, we trace the end of w from the $(i+1)$th element to the last element, from coset c_2 to coset c. At each

step, we record the element of w used and the current coset number. Next, we record w^{-1}, using square brackets if w is a generator of H, and round brackets if w is a relator of G. Finally, we trace the start of w from the first element to the $(i-1)$th element, from coset c to coset c_1, recording generators and cosets traversed.

Example 4.1.2 If $w = xyz$ is a relator of G and $i = 2$, then in this instance the generator g involved in the deduction is equal to y and we obtain the entry "$c_2, z, c, (z^{-1}y^{-1}x^{-1}), c, x, c_1$" for $c_2y^{-1} = c_1$ and the entry "$c_1, x^{-1}, c, (xyz), c, z^{-1}, c_2$" for $c_1y = c_2$.

If tracing a word w from some coset c results in the primary coincidence $c_1 = c_2$ at index i, then the redundancy proof table entry of such a coincidence is built as follows: First, we trace the end of w from the ith element to the last element, from coset c_2 to coset c. At each step, we record the element of w used and the current coset. Next, we record w^{-1}, using square brackets if w is a generator of H and round brackets if w is a relator of G. Finally, we trace the start of w from the first element to the $(i-1)$th element, from coset c to coset c_1, recording the generators and cosets traversed.

Example 4.1.3 If $w = xy$ is a generator of H and $i = 2$, then in this instance the coset c must be equal to 1 and we obtain the entry "$c_2, y1, [y^{-1}x^{-1}], 1, x, c_1$". If $c_1 < c_2$, then we record this entry in $Q[c_2]$, and otherwise we record "$c_1, x^{-1}, 1, [xy], 1, y^{-1}, c_2$" in $Q[c_1]$.

If we have a secondary coincidence $c_1 = c_2$ resulting from two chains of coincidences "$x_1, =, \ldots, =, x_j, = c_1$" and "$y_1, =, \ldots, =, y_k, =, c_2$ and the proof entry w for $x_1 = y_1$, then the redundancy proof table entry of such a coincidence is built as follows: First, we add the inverse of the first chain: "$c_1, =, x_j, =, \ldots, =, x_1$". Then we add all but the first entry (x_1) and the last entry (y_1) of w. Then we add the second chain "$y_1, =, \ldots, =, y_k, =, c_2$".

Each proof table entry provides justification for the change in the coset table. The entries indicate partial proof words, given by the proof items between cosets. The equals signs and the generators of G are used to indicate a relationship between cosets which has already been established, either by definition or deduction. Note that the proof words formed by excluding the coset numbers and equality signs will reduce to the generator of G associated with the column of the coset table entry, or if the word is to justify a coincidence of cosets then it will reduce to the identity under s_1.

4.1.4 Processing coincidences

To process the coincidence of c_1 and c_2, we first build the proof table entry for the primary coincidence $c_1 = c_2$. We begin the coincidence list with the entry (c_1, c_2). For each pair (c_1', c_2') in the coincidence list, we obtain the non-redundant representative of each coset, namely r_1 and r_2, respectively, using column 2 of A. Suppose $r_1 < r_2$. Then non-zero entries from the row of T corresponding to r_2 need to be copied to the row corresponding to r_1 if the entry in the corresponding col-

umn is 0, while new coincidences may result if this entry is non-zero. Special care needs to be taken if the entry in either row is r_1 or r_2; in that case, the relevant column and corresponding inverse column of T are both updated to r_1 and coincidences $r_1 = x$ are added for each entry x in the relevant column or the inverse column in either row, for $x \notin \{0, r_1, r_2\}$. Any new coincidences obtained in this way are called *secondary coincidences*.

Example 4.1.4 If the row of T corresponding to coset r_1 is $[0, 1, 2, 3]$ and the row of T corresponding to coset r_2 is $[4, 0, 5, 6]$, then we have the secondary coincidences of cosets 2 and 5, and cosets 3 and 6, and the row of T corresponding to r_1 becomes $[4, 1, 2, 3]$. If the row of T corresponding to r_1 is $[1, r_2, 2, 3]$ and the row of T corresponding to r_2 is $[4, 5, 6, r_1]$, then we have the secondary coincidences of cosets 1 and 4, cosets r_1 and 5, cosets r_1 and 3, and cosets 2 and 6, and the row of T corresponding to r_1 becomes $[1, r_1, 2, r_1]$.

When an entry x is copied in the column corresponding to generator g, we add the proof table entry "$r_1, =, r_2, g, x$" to $P[r_1, g]$. For a secondary coincidence $x = y$ in the column corresponding to generator g, we use the proof entry "$x, g, r_1, =, r_2, g^{-1}, y$" to build a secondary coincidence redundancy proof table entry. Then we add (x, y) to the coincidence list.

Example 4.1.5 If the row of T corresponding to coset r_1 is $[0, 1, 2, 3]$ and the row of T corresponding to coset r_2 is $[4, 0, 5, 6]$, and if g is the

generator of G corresponding to column 1, then the proof table entry "$r_1, =, r_2, g, 4$" is added to $P[r_1, g]$. The sequence "$5, g, r_2, =, r_1, g^{-1}, 2$" is used to build the secondary coincidence entry for the coincidence of cosets 2 and 5.

Finally, we remove the row corresponding to r_2 from the coset table and update columns 1 and 2 of A accordingly. We repeat this process until there is nothing left in the coincidence list.

4.1.5 Applying relators and generators to cosets

To apply the relator r to the coset c, with parameter m, we first scan r forwards from c. If c_1 is the current coset, then for each k, if r_k is the kth element of r, then if the coset table entry corresponding to $[c_1, r_k]$ is non-zero, then set $c_1 = T[c_1, r_k]$. We repeat this process until a zero entry is reached. If no zero entry is reached and $c_1 \neq c$, then we have the coincidence $c_1 = c$ to process.

If a zero entry is found as the result of multiplication of c_1 by r_f, then we scan r backwards from c, with current coset c_2, until either a zero entry is found as the result of multiplication of c_2 by r_b^{-1} or until the fth element of r is reached. If a zero entry is found as the result of the multiplication of c_2 by r_f^{-1}, then to satisfy r we must have $c_1 r_f = c_2$. In that case, we update T accordingly and build the appropriate proof table entry.

If a zero entry is found at index $b \neq f$ and $m \geq b - f$, then we define a new coset c_3 with $c_3 r_b^{-1} = c_2$. In that case, we update A and T accordingly. If $T[c_1, r_f]$ is now defined, then we continue the forward scan, updating c_1 and f, and we set $c_2 = c_3$ and move to the next generator in r (that is, we subtract 1 from b). We repeat this process until $b = f$. Again we must have $c_1 r_f = c_2$, so we update accordingly and build the proof table entry.

If no zero entry is found, then some coset c_2 is reached with $c_2 r_f^{-1} = c_3$ already defined, and we have the coincidence $c_3 = c_1$ to process.

Example 4.1.6 If $h = x^2y^2$ is a generator of H we wish to apply, we start with $c_1 = 1$. Suppose $T[1, x] = 3$, $T[3, x] = 4$, and $T[4, y] = 5$. If $T[5, y] = 6$ then we have the coincidence of cosets 1 and 6 to process. If $T[5, y]$ is 0 then $f = 4$ and $c_1 = 5$. Next, we scan r^{-1} from coset 2, starting with $c_2 = 1$. If $T[1, y^{-1}] = 7$ then we have the coincidence of cosets 5 and 7 to process. If $T[1, y^{-1}] = 0$ then we update T so that $T[1, y^{-1}] = 5$, and add a proof table entry to correspond with coset 5 and with the sequence "$1, [y^{-2}x^{-2}], 1, x, 3, x, 4, y, 5$", to $P[1, y^{-1}]$.

Example 4.1.7 Let $r = yzyz^2$ be a relator of G we wish to apply to coset 2. If $T[2, y] = 3$ and $T[3, z] = 0$ then we have have $c_1 = 3$ and $f = 2$. Next, we let $c_2 = 2$ and scan r^{-1} from 2. If $T[2, z^{-1}] = 6$ and $T[6, z^{-1}] = 0$ then $c_2 = 6$ and $b = 4$. If $m = 1$ then $b - f = 2 > m$ so we do nothing. If $m \geq 2$ then we define a new coset c_3. If n is the number of cosets already defined then $c_3 = n+1$. We update T so that $T[6, z^{-1}] = n+1$ and add a proof table entry to $P[6, z^{-1}]$ to correspond

with the coset $n+1$ and with a empty proof table sequence. Next we set $c_2 = n+1$, $b = 3$ and consider $T[n+1, y^{-1}]$. Since this row of the coset table was added in the previous step this entry will be 0, so we add another coset $n+2$ and set $T[n+1, y^{-1}] = n+2$ and add a proof table entry with empty sequence to $P[n+1, y^{-1}]$. Now $c_2 = n+2$ and $b = 2 = f$. Finally we set $T[3, z] = n+2$ and add a proof table entry with the sequence "$3, y^{-1}, 2, (yzyz^2), 2, z^{-1}, 6, z^{-1}, n+1, y^{-1}, n+2$" to $P[3, z]$.

4.1.6 Simplifying proof words

To keep the length of proof words reasonably short, we have a few methods of simplification. First note that items can be inverses of each other only if they are of the same type. Given two proof words $w_1 \neq w_2$, if $s_1(w_1) = s_1(w_2)$ and $s_2(w_1) = s_2(w_2)$, then as proof words they are equivalent. We can simplify proof words by replacing sections of the word with shorter equivalent words. We do this using five methods:

1. Consecutive inverse items can be removed since they will be removed under both s_1 and s_2. For example the section $[xy][y^{-1}x]$ can be removed.

2. If two inverse items i_1 and i_2 of type 1 are separated by a single type 2 item i_3, and either i_1 is equal to the last element of i_3, or i_2 is equal to the first element of i_3, then moving i_1 and i_2 inside the brackets of i_3 gives a cyclic rearrangement i_4 of i_3. Since i_3 is a relator of G, so is any cyclic rearrangement of it. Hence the proof word section (i_1, i_3, i_2) can be replaced by i_4. For example

the section $x(yx)x^{-1}$ can be replaced by (xy).

3. If two inverse items of type 2 have the word w_1 between them, they may be removed if $s_1(w_1) = 1_G$. Under s_1 this section will reduce to the identity, while under s_2 the type 2 items will be removed anyway. Note that w_1 can contain any combination of item types. For example the section $(x^4)x(yz)x^{-1}[xz^{-1}](y^{-1}x^{-1})(x^{-4})$ may be replaced by $x(yz)x^{-1}[xz^{-1}](y^{-1}x^{-1})$.

4. If two consecutive type 3 items are the inverse of each other, then the square brackets can be removed. Under s_2 these items will be adjacent, and hence can be removed, and removing brackets does not affect the simplification under s_1. The word between the inverse pair may contain items of type 1 and/or 2. For example the section $[xy^{-1}]y(x^4)y^{-1}[yx^{-1}]$ may be replaced by $xy^{-1}y(x^4)y^{-1}yx^{-1} = x(x^4)x^{-1} = (x^4)$.

5. If there are two inverse items i_1 and i_2 of type 2, with no type 3 items between them, and the word formed by the type 1 generators between them reduces to 1_G, then the brackets may be removed from i_1 and i_2. Under s_2 this section will reduce to the identity, while under s_1 the brackets will be removed anyway. For example the section $(xy)z(x^4)z^{-1}(y^{-1}x^{-1})$ can be replaced by $xyz(x^4)z^{-1}y^{-1}x^{-1}$.

There are other ways of rearranging proof words which may result in a shorter equivalent word, but finding the shortest possible proof is infeasible, so we employ only the techniques described above in the

main algorithm.

Included in the code are two additional procedures for simplifying words, which may be useful if the proof word is particularly long.

The first procedure searches for sections w of the proof word which have the property that $s_1(w)$ is a relator of G and $s_2(w) = 1_G$. Such sections can be replaced by a single type 2 proof item equal to $s_1(w)$.

The second procedure can be used if there is a H-generator x with $x^n = 1_G$ as a relator of G. Any section w of the proof word which simplifies to x^n as an element of H under s_2 can be simplified by removing the square brackets of each type 3 item and adding n copies of x as a sequence of type 1 items, and then adding the type 2 item x^n. This may increase the length of the proof word but will reduce the length of the simplification under s_2.

Whenever we join two simplified proof words w_1 and w_2, we need only check for inverse pairs across the point of concatenation. By doing this, we greatly reduce processing time.

We carry out each method in the order listed. If an item is removed from a word, we must check across the point of concatenation of the two sides of that item. If an item is replaced, then we have two points of concatenation to check.

4.1.7 Building proof words from proof table entries

The tables W and V contain full proof words. The entries of W corresponding to the coset c_1 and the generator g are the proof words for $c_1 g = c_2$, for each c_2 described in P. The c_1th entry of the table V contains the proof word for $c_1 = c_2$, where c_2 is the representative of c_1 as given by $A[c_1, 2]$. These proof words are created using P and Q, along with entries already filled in W and V.

Filling the entirety of both tables can be quite time-consuming, and is not always required, so we fill in the entries as they are used. By storing each proof word that we find, we reduce processing time, since they are often used multiple times.

To create a proof word w for $c_1 g = c_2$ or $c_1 = c_2$, we first find the appropriate proof entry p from P or Q. If $|p| = 0$, then this is a definition, and so $w = g$. Note that redundancies cannot be definitions.

Otherwise, $|p| \geq 3$, and in that case we consider three consecutive elements of p at a time, say d_1, d_2, d_3. The first and third elements will be cosets (given as integers), and the middle element d_2 will be a string.

If d_2 is '=', then this section of p describes a coincidence, and so we get the proof word from V and add it to the end of w. Then we set $d_1 = d_3$, and consider the two subsequent elements of p.

If d_2 is a generator of G, then this section of p describes the deduction or definition $d_1 d_2 = d_3$. We find the appropriate word in W and add it to the end of w. Then we let $d_1 = d_3$, and consider the two subsequent elements of p.

Otherwise, d_2 will be a subgroup generator or a group relator. We add d_2 to the end of w. Then we set $d_1 = d_3$ and continue by considering the two subsequent entries of p.

We repeat until we reach the end of p. Whenever we find a proof word, we add it to the appropriate position in V or W. If at any point we find that we need an entry of W or V that has not yet been filled, then we create the required proof word, recursively building w.

Example 4.1.8 Suppose we wish to find the proof word w for $3x = 6$. We first consider the proof table entry of $P[3, x]$ associated with the coset 6. If this entry is an empty sequence, then $w = x$ and we are done. Otherwise suppose that this entry is the sequence "$3, x, 8, (z^2), 8, z^{-1}5, z^{-1}, 6$". If the entry of $W[3, x]$ associated with coset 8 is the word x then we start with $w = x$. Next we add the relator (z^2) of G to get $w = x(z^2)$. If the entry of $W[8, z^{-1}]$ associated with coset 5 is filled with the word z^{-1} then we add this to the end of w to get $w = x(z^2)z^{-1}$. Now suppose the entry of $W[5, z^{-1}]$ is not yet filled. Then we must find the proof word w_1 for $5z^{-1} = 6$.

Suppose the entry of $P[5, z^{-1}]$ associated with coset 6 is "$5, z^{-1}$, $9, =, 6$," and let the entry of $W[5, z^{-1}]$ associated with the coset 9 be z^{-1}. Then we start with $w_1 = z^{-1}$. Next we consider the proof table entry $V[9]$. If this is not yet filled we must find the proof word w_2 for $9 = 6$ by considering the redundancy proof table entry $Q[9]$. We then add this w_2 to the end of w_1 (checking for any simplifications) to obtain $w_1 = z^{-1}w_2$. Finally we add w_1 to the end of w to get $w = x(z^2)z^{-2}w_2$.

To get a proof word w_1 verifying that the word $h \in H$ given in terms of the generators of G is in H, we first let $c_1 = 1$. Let c_{i+1} be the entry of T corresponding to $[c_i, h_i]$, where h_i is the ith element of h. For each i, we retrieve/create the corresponding entry of W and add it to the end of w_1. If this process concludes at coset 1, then $h \in H$, while otherwise, h is in the coset reached.

4.1.8 Running PEACEv2

First, we apply each generator h of H to coset 1 with parameter $m = |h| + 1$. Next, we check each new entry of T. Suppose there is a new entry corresponding to coset c and generator g. If c is still active, then for each rotation r of relators of G having first element g, we apply r to coset c with parameter $m = 0$. If all new entries have been checked and there are no holes in T, then we are finished.

If all new entries have been checked and there are still holes in T, then we add a new coset by filling the first hole in the table. If the first

hole of T occurs in the row corresponding to coset c_1 in the column corresponding to generator g, then we add a new coset c_2 with $c_1 g = c_2$. We then update T, A and P accordingly, and return to checking new entries.

Like `PEACE`, this procedure is not guaranteed to complete in the sense that a complete coset table might not be obtained, and in fact this is not possible if the subgroup H is not of finite index. But, maximum values for the number of rows in the coset table and number of passes through the main loop are in place to avoid any infinite loops that may occur.

Note that the proof words obtained are in fact self justifying, so the soundness of each result is self evident.

4.1.9 Coset definitions

Columns 3 and 4 of A store the necessary information to find the definition for each coset c. The entry $A[c, 3]$ gives the integer representing the coset from which c was defined, and $A[c, 4]$ gives the integer representing the generator used in the definition. Recursively taking each generator used until we reach coset 1 then gives the definition of c. If H is the trivial subgroup, then finding the definition of each coset gives a representative for each element of G. Similar properties hold for normal subgroups and their quotients.

4.2 Application

Recall that given a finitely presented group G and a set $\gamma = \{g_1, \ldots, g_n\}$ of words in the generators of G, such that γ generates G, the PEACEv2 procedure can give a proof that each generator of G can be written as a word in the elements of γ. This can be useful when finding new presentations for groups from presentations already known.

Using the 3-generator presentation for $SL(3,\mathbb{Z})$ given by Conder, Robertson and Williams in [13], namely

$$SL(3,\mathbb{Z}) \cong \langle\, x, y, z \,|\, x^3, y^3, z^2, (xz)^3, (yz)^3, (Xzxy)^2, (Yzyx)^2, (xy)^6 \,\rangle,$$

we obtain several new 2-generator presentations for $SL(3,\mathbb{Z})$.

The proof words can be found by running the file PEACEv2.m from `github.com/GLiversidge/PEACEv2` then using the code below, with the appropriate elements of `G` for `u` and `v` and the desired word for `w`.

```
G<x,y,z>:=Group<x,y,z|x^3, y^3, z^2, (x*z)^3, (y*z)^3,
(x^-1*z*x*y)^2, (y^-1*z*y*x)^2, (x*y)^6>;
u:= z;
v:= x*y;
w:="y";
L:=PEACEv2(G, {u,v});
proveWrd(~L, w, ~pfwrd);
pfwrd;
```

The simplification of the word `pfwrd` with method s_2 gives an expression for `y` in terms of the elements `u` and `v`. We can either use the expression for `y` to solve for an expression for `x` in the elements `u` and `v`, or repeat the process using the following code.

```
w2:="x";
proveWrd(~L, w2, ~pfwrd2);
pfwrd2;
```

Note that we use the same variable `L`, and the formation of this proof word is likely to be faster since all partial proof words found in the creation of `pfwrd` have been stored in `L`.

Finally, we use Tietze transformations to add the new symbols $u = z$ and $v = xy$ and replace appearances of x, y and z in the relations with the appropriate word in terms of u and v. This can be by hand, or simply with the use of copy and paste.

These presentations led to further work on generating pairs for $SL(n, \mathbb{Z})$ for $n \geq 3$ (see [12]). Two of the main discoveries in [12] are as follows: $\mathrm{SL}(n, \mathbb{Z})$ is generated by two elements of infinite order, for all $n \geq 2$, and is generated by an element of order 2 and an element of infinite order, for all $n > 2$. This was achieved by completing partial work on this topic by previous authors, but using different approaches, which are beyond the scope of this project and so we do not give the details here.

4.3 Results

All presentations could potentially be reduced in length, but we give them as produced by PEACEv2 to illustrate the use of this package.

Note that in proof words we write $(w)^k$ only for w and k such that $(w)^k$ is a type 2 proof word item. Furthermore we use the case inverse notation $a^{-1} = A$ for compactness.

Letting $u = z$ and $v = xy$, we obtain the proof word:

$xYX(x^3)(XZ)^3(ZxYX)^2xyX[z][xy]X(z^2)x[z]x(z^2)X^2(x^3)(XZ)^3x$
$[YX][Z][xy](Y^3)y(yXzx)^2(zy)^3Y[Z]Y(Z^2)y[YX][Z]X(Z^2)x[YX][Z]$
$[xy](Y^3)yX(xyXz)^2(zx)^3(X^3)xYxYX(x^3)(XZ)^3(ZxYX)^2xyX[z][xy]X$
$(z^2)x[z]x(z^2)X^2(x^3)(XZ)^3x[YX][Z][xy](Y^3)y(yXzx)^2(zy)^3Y[Z]Y(Z^2)y$
$[YX][Z]X(Z^2)x[YX][Z][xy]Y^2X(xyXz)^2(zx)^3(X^3)xy^2.$

Simplification under s_1 is achieved by removing all brackets and cancelling adjacent inverse pairs. The result is y.

Simplification under s_2 is achieved by removing the contents of all () brackets and cancelling adjacent inverse pairs. The result is:

$$[z][xy][z][YX][Z][xy][Z][YX][Z][YX][Z][xy]$$
$$[z][xy][z][YX][Z][xy][Z][YX][Z][YX][Z][xy].$$

This shows that $y = (uvuVUvUVUVUv)^2$ and
$x = vy^{-1} = v(VuvuvuVuvUVU)^2$.

This gives the following presentation, with generators of orders 2 and 6:

$$\begin{aligned} SL(3,\mathbb{Z}) \cong \langle\, u, v \mid\, & u^2, v^6, (v(VuvuvuVuvUVU)^2)^3, (uvuvuVuvuVuV)^6, \\ & ((uvuvuVuvuV)^2uV)^3, ((vuVuvuVuVuv)^2u)^3, \\ & (Uv(VuvuvuVuvuVu)^4(vuVuvuVuVuvu)^2)^2 \,\rangle. \end{aligned}$$

If we let $u = xz$ and $v = yz$, then we obtain the following presentation, with generators of order 3:

$$\begin{aligned} SL(3,\mathbb{Z}) \cong \langle\, u, v \mid\, & u^3, v^3, ((uv)^3UvUVUvuvuVUVuvuVuVUV)^3, \\ & (uvuvUvUVUvuvUVUVuvuVuVUV)^2, \\ & (UVUVuvuVuVUVuvuvUvUVUvuvuVUVuvuVuVUV \\ & (uv)^3UvUVUvuv)^2, \\ & (VUVuvuVuVUVUVUvuvUvUVUvuvUVUVuvuVuV \\ & UVUvuvuvUvUVUvuvu)^2, \\ & (uVUVuvuVuVUV(uv)^3UvUVUvuvu)^6 \,\rangle. \end{aligned}$$

If we let $u = xz$ and $v = yx$, then we obtain the following presentation, with generators of orders 3 and 6:

$$\begin{aligned} SL(3,\mathbb{Z}) \cong \langle\, u, v \,|\, & u^3, v^6, (Vu)^4, (VUvUvuvUvuVuVuvu)^3, \\ & (vuVUvUv(UVuV)^2uvu)^3, (vu(VUvUv(UVuV)^2uvU)^2)^3, \\ & ((UVUvUv(UVuV)^2uv)^3uVUvUvuvUvuVuVuvUvU)^2, \\ & (UVUvUvuvUvuVuVuvUVuVUvUv(UVuV)^2uvUv)^2 \,\rangle. \end{aligned}$$

If we let $u = xy$ and $v = Xyz$, then we obtain the following presentation, with generators of orders 6 and 6:

$$\begin{aligned} SL(3,\mathbb{Z}) = \langle\, u, v \,|\, & u^6, (uvUvUV)^6, ((uVu)^2VUv)^3, \\ & ((vUvUVu)^3vUvU)^2, (v^2UvUVuvUvU^2(vUVuvU)^3)^3, \\ & ((vuVuVU)^2(vUvUVu)^3vUv)^2, ((vUv)^2UVu)^3, \\ & (uvU(vUVuvUvU^2)^2(vUVuvU)^3v(UvuVuV)^2)^2 \,\rangle. \end{aligned}$$

If we let $u = z$ and $v = xY$, then we obtain the following presentation, with generators of order 2 and infinite order:

$$\begin{aligned} SL(3,\mathbb{Z}) = \langle\, u, v \,|\, & u^2, (VUvuvUVUvuVuV)^6, \\ & (V(VUvuvUVUvuVuV)^2)^3, (u(VUvuvUVUvuVuV)^2)^3, \\ & (uV(VUvuvUVUvuVuV)^2)^3, (Vu(VUvuvUVUvuVuV)^2)^2, \\ & (vuV(VUvuvUVUvuVuV)^2)^2, ((VUvuvUVUvuVuV)^4V)^6 \,\rangle. \end{aligned}$$

If we let $u = x$ and $v = xYz$, then we obtain the following presentation, with generators of order 3 and infinite order:

$$\begin{aligned} SL(3,\mathbb{Z}) = \langle\, u, v \mid\, & u^3, ((vUV^2uVuv)^2uvUV^2UVUvUv^2uVUVUvu)^3, \\ & ((uvUV^2uVuvuv)^2UV^2UVUvUv^2uVUVUv)^2, \\ & ((vUV^2uVuvuvU)^2V^2UVUvUv^2uVUVUvu)^3, \\ & ((uvUV^2uVuvuv)^2UV^2UVUvUv^2uVUVUvu(vUV^2uVuv)^2 \\ & uvUV^2UVUvUv^2uVUVUv)^3, \\ & ((vUV^2uVuvU)^2UvUV^2UVUvUv^2uVUVUvu \\ & (vUV^2uVuvuvU)^2VUvUv^2uVUVUvu)^2, \\ & ((UvuvUV^2uVuv)^2uvUV^2UVUvUv^2uVUV)^2, \\ & ((uvUV^2uVuv)^2uvUV^2UVUvUv^2uVUVUv)^6 \,\rangle. \end{aligned}$$

Proof words for all the above presentations can be found at `github.com/GLiversidge/PEACEv2`.

4.4 Comparable algorithms

The PEACE package on which PEACEv2 is based is written in the programming language C and uses make files to compile, so it is not particularly accessible. Our aim was to improve the usability of PEACE by creating the function PEACEv2 implemented in Magma. Implementing the function in a computational language commonly used for computational group theory allows users to benefit from the procedure without having to open a separate package (and copying all the information required).

PEACE also has lots of options for experimenting with coset enumeration types and different equivalent presentations. We've removed these options and instead added functionality for reducing the results obtained. This allows for users to benefit from the procedure even with limited knowledge of the coset enumeration process and style.

By building a complete table of words as required we reduce run-time when requesting multiple proof words, allowing for faster exploration of the proof words available.

Both PEACE and PEACEv2 have the capability of producing proof words, which, to the best of our knowledge, is unique to these algorithms. Proof words allow users to manually check the results if desired, an option which I think will particularly interest students or those new to computational group theory.

Chapter 5

LowIndAlg

5.1 LowIndAlg Algorithm

LowIndAlg takes a finitely-presented group G, an integer M and a list B of words in the generators of G, and returns a list of generating sets for a complete set of representatives of conjugacy classes of subgroups of G that have index M and do not contain any conjugate of any word in B.

The main mechanism for avoiding conjugates of words in B is by applying the words in B to each coset (just as relators are applied in traditional forms of coset enumeration), but instead of using these applications to fill the coset table, we use them to find coincidences of cosets that will result in a word in B, or a conjugate of a word in B, appearing in the subgroup. We wish to avoid such coincidences, and call them *excluded coincidences*. We call coincidences which have not (yet) been excluded *allowed coincidences*.

5.1.1 Definitions

Let C be a (partial) coset table for a subgroup of G, and let $C[c, g]$ be the entry corresponding to the coset c and generator $g \in G$. Note that entries in C occur in pairs: if we assign $C[c_1, g] = c_2$ then we always assign $C[c_2, g^{-1}] = c_1$. This may be assumed throughout the algorithm.

Let L be the list of coset labels, with $L[c_1]$ giving a word w in the generators of G such that w is in the coset c_1. Let E be the list of allowed coincidences. If $c_1 < c_2$ are cosets and $c_2 \in E[c_1]$, then the coincidence $c_1 = c_2$ is allowed. Otherwise, the coincidence is not allowed.

Let X be the list of excluded words. This is initially filled using B. Before adding a word w to X, we *simplify* w by removing any appearances of the relators of G and taking conjugates by generators of G to get the *least representative* of w. By taking least representatives and not taking duplicates, we minimize the size of X. Note that $g^{-1}wg$ is equivalent to w since if $w \in H$ for some subgroup H of G, then for any $g \in G$, $g^{-1}wg \in g^{-1}Hg$ which is in the conjugacy class of H. Let $X[g]$ be the list of all cyclic rearrangements and inverses of excluded words that have first element g.

Similarly, let R be the list of relators of G, and let $R[g]$ be the list of all cyclic rearrangements and inverses of relators of G, that have first element g.

5.1.2 Processing the coincidence of two cosets

Let $c_1 < c_2$ be the two cosets to be processed. If $c_2 \notin E[c_1]$ then we kill this branch of the search tree, in the same way as branches are killed in the standard Low Index Subgroups algorithm (described in Section 2.4) when they lead only to conjugates of subgroups already found. Otherwise, we compare information from row c_2 with information from row c_1 of the coset table.

For each column g, if $C[c_2, g] \neq 0$, then we have two cases depending on the value of $C[c_1, g]$. If $C[c_1, g] = 0$, then we assign $C[c_1, g] = C[c_2, g]$. If $C[c_1, g] \neq 0$ then we have a new coincidence $C[c_1, g] = C[c_2, g]$, and we then add this to a list to process later. Once each column has been compared, we update E by disallowing any coincidence $c_1 = c_3$ such that the coincidence $c_2 = c_3$ is not allowed. Finally, we remove the coset c_2, and move on to the next coincidence in the list.

Example 5.1.1 Suppose we wish to process the coincidence of the cosets 6 and 7. If the corresponding rows of C are $[3, 0, 4, 0]$ and $[0, 5, 9, 0]$, respectively, then row 6 becomes $[3, 5, 4, 0]$ and the pair 4 and 9 is added to the list of coincidences to process later. If 6 appears only in $E[2]$ and $E[4]$ and 7 appears only in $E[4]$ and $E[5]$, then 6 is removed from $E[2]$, but remains in $E[4]$. If $E[6] = \{8, 9, 10\}$ and $E[7] = \{9, 10, 11\}$ then $E[6]$ becomes the set $\{9, 10\}$.

5.1.3 Applying the relator r to the coset c

Let r_i be the ith element of r. Starting with $c_1 = c$, we scan r by letting $c_{i+1} = C[c_i, r_i]$, provided that this value is non-zero. If no zero entries are found and the scan completes at the coset $c' \neq c$, then we have a coincidence $c' = c$ which we process immediately.

If the scan is incomplete, with $C[c_i, r_i] = 0$, then starting from $c'_{|r|} = c$ we scan r backwards by letting $c'_{j-1} = C[c'_j, r_j^{-1}]$, provided this value is non-zero. If $C[c'_i, r_i^{-1}] = c_{i-1} \neq 0$, then we have a coincidence $c_i = c'_{i-1}$ which we process immediately. If $C[c'_i, r_i^{-1}] = 0$, then we can satisfy the relator by assigning $C[c_i, r_i] = c'_i$. If $C[c'_{i+1}, r_{i+1}^{-1}] = 0$ then we can satisfy the relator by adding a single coset n and assigning $C[c_i, r_i] = n$ and $C[n, r_{i+1}] = c'_{i+1}$. If $C[c'_{i+2}, r_{i+2}^{-1}] = 0$ then there are two possibilities. If $c_i = c'_{i+2}$ and $r_i = r_{i+2}^{-1}$ then we can satisfy the relator by adding a single coset n and assigning $C[c_i, r_i] = n$ and $C[n, r_{i+1}] = n$. Otherwise, if $c_i \neq c'_{i+2}$ or $r_i \neq r_{i+1}^{-1}$ then we need two cosets $n-1$ and n, and we assign $C[c_i, r_i] = n-1$, and $C[n-1, r_{i+1}] = n$, and $C[n, r_{i+2}] = c'_{i+2}$. If $C[c'_j, r_j^{-1}] = 0$ for $j < i+2$ then we could add cosets to complete the scan, but in this implementation we restrict the value of j to $i+2$.

Example 5.1.2 Suppose we wish to apply the relator $r = xyxy$ to the coset 2. Let $C[2, x] = 3$, $C[3, y] = 4$, and $C[4, x] = 5$. If $C[5, y] = 6$ then we have the coincidence of cosets 2 and 6 to process. If $C[5, y] = 0$ then we let $c_2 = 2$ and scan r^{-1} from coset 2. If $C[2, y^{-1}] = 7$ then we have the coincidence of the cosets 5 and 7 to process. If $C[2, y^{-1}] = 0$ then we set $C[5, y] = 2$ to satisfy the relator r.

Example 5.1.3 Suppose we wish to apply the relator $r = xyx^{-1}y^{-1}x$ to coset 3, and $C[3, x] = 4$ and $C[4, y] = 0$. Then we scan r^{-1} from coset 3. If $C[3, x^{-1}] = 4$ then we can complete the relator by adding a single coset n, when the current number of cosets is $n - 1 < M$. We define $C[4, y] = n$ and then set $C[n, x^{-1}] = n$ to complete the scan.

5.1.4 Continuing enumeration

We fill as many entries in the coset table as possible, starting by processing all new entries. For an entry corresponding to the coset c and generator g, we first scan each word in $X[g]$ from c as far as possible. If the scan completes at the coset c, then we kill this branch of the search tree. If the scan completes at the coset $c_2 \neq c$, then we update E to disallow the coincidence $c = c_2$. Next, we apply all relators in $R[g]$ to coset c, allowing only the addition of single cosets as long as $|C| \leq M$.

Example 5.1.4 Suppose that $C[2, y] = 3$ is a new entry and $X[y] = \{yx\}$. If $C[3, x] = 2$ then we kill this branch of the tree. If $C[3, x] = 4$ then we remove 4 from $E[2]$ and then apply all relators in $R[y]$.

Once all new entries have been processed, we check to see if the coset table is complete. If the coset table is complete then we have found a subgroup. If $|C| = M$ then we add this to the list of subgroups. This branch of the search tree is now complete.

If the coset table is incomplete then we have two cases. If $|C| < M$ then we add cosets. First we apply each relator of G to each coset, until at least one coset has been added. If no cosets have been added then we fill the first hole in the table. If this hole is in row c and column g then we add a coset to the coset table and assign $C[c, g] = |C|$. If $|C| \geq M$ then we have filled as much of the table as possible, so we add new generators, creating new branches of the search tree.

5.1.5 Adding new generators/branches

First we add new generators by forcing the coincidence of cosets. For each coset j in each list $E[i]$, we create a new branch and continue enumeration with the coincidence $i = j$, then remove j from $E[i]$. This branch is equivalent to adding the subgroup generator given by $w = L[i]L[j]^{-1}$. If we are on the first level of the search tree, that is, we are adding the first generator of a subgroup, then we add w to X, which excludes conjugate subgroups. If we are adding other generators, then excluding conjugate subgroups is more complicated, so we remove them at the end. If adding the generator w to X, we must first update E by scanning w from each coset c_1. If the scan completes at the coset c_2 then we disallow the coincidence $c_1 = c_2$.

Example 5.1.5 Let $E[1]$ be empty and let $E[2] = \{5, 8\}$ where $L[2] = x$, $L[5] = xy$ and $L[8] = xyx$. Our first branch created will add the generator $xy^{-1}x^{-1}$ by forcing the coincidence of cosets 2 and 5. If this is the first generator added then once this branch is complete we add $xy^{-1}x^{-1}$ to X. $E[2]$ will become the set $\{8\}$ and other sets in E may

also be updated by the application of $xy^{-1}x^{-1}$ to other cosets. We then add a branch with the generator $y^{-1}x^{-1}$ by forcing the coincidence of cosets 2 and 8.

Next, we add generators by filling holes in the coset table. If our first hole is in row c_1 and column g, then this position could be filled by any coset c_2 which has 0 in the column g^{-1}. For each possible c_2 we create a new branch and continue enumeration with $C[c_1, g] = c_2$. In this second method of adding branches, we don't allow any coincidences. If a coincidence is required, then the branch is killed. This is equivalent to adding the generator given by $L[c_1]gL[c_2]^{-1}$.

Example 5.1.6 Let $C[2, x] = 0$, $C[6, x^{-1}] = 0$, $L[2] = x$ and $L[6] = yxy$. We create a branch with the added generator $x^2y^{-1}x^{-1}y^{-1}$ by setting $C[2, x] = 6$. We then continue applying relators. If any relator causes any coincidences then the branch is killed.

These two methods ensure that every possibility is explored. The first method covers the coincidence between cosets, the second covers each possible way a partial coset table may be completed without coincidence. Once all possible coset tables have been explored, then all conjugacy classes have been explored. Furthermore, note that there is a finite number of ways to fill a coset table of a subgroup with a given index, so the algorithm is guaranteed to complete.

Once all branches have been completed or killed, we filter the list of subgroups found. For each set of conjugate subgroups in the list, we remove all but one subgroup of the set and then the list is returned.

5.2 Application

Recall that manifolds may be constructed from Coxeter groups and that there has been particular interest in small volume 3-manifolds, constructed from minimum-index torsion-free subgroups of $[p, q, r]$ Coxeter groups with $1/p + 1/q \geq 1/2$, $1/q + 1/r \geq 1/2$ and $p, q, r \geq 3$. There are 15 such Coxeter groups and until recently the minimum-index of a torsion-free subgroup was known for all except the $[4, 3, 5]$ and $[6, 3, 6]$ Coxeter groups.

Our initial attempt to use the Low Index Subgroup algorithm as implemented in MAGMA for the $[4, 3, 5]$ case took a very long time, with the algorithm often failing, and eventually completing after over 12 weeks. In the meantime, we implemented `LowIndAlg` in MAGMA. This algorithm was able to find exactly 14 conjugacy classes of torsion-free subgroups of index 240 in the [4,3,5] Coxeter group within 7 hours, and exactly 12 conjugacy classes of torsion-free subgroups of index 24 in the [6,3,6] Coxeter group in less than a second.

In the case of searching for torsion-free subgroups of Coxeter groups given by the presentation

$$[p, q, r] = \langle\, a, b, c, d \,|\, (ab)^p, (bc)^q, (cd)^r, a^2, b^2, c^2, d^2, (ac)^2, (ad)^2, (bd)^2 \,\rangle,$$

every torsion element is conjugate to an element of one of the maximally finite subgroups generated by a subset S of $\{a, b, c, d\}$, and so the list `B` contains words representing elements of each such subgroup.

The conjugacy classes of torsion-free subgroups of index n can be found by running the `LowIndAlg.m` file and then running `LowIndAlg(G, n, B)`, with the appropriate group G, and the appropriate list B.

Once all conjugacy classes of torsion-free subgroups are found, we check for isomorphisms between representatives of each class. If two subgroups have different abelianisations or have different numbers of subgroups of a particular index, then they cannot be isomorphic. If no differences are found then the MAGMA function `SearchForIsomorphism` may provide an isomorphism between the pair (if one exists). By checking abelianisations and the number of subgroups of index up to 8 we were able to draw up a list of representatives not isomorphic to any other(s), and find isomorphisms between pairs of the remaining ones, except one pair in the [5,3,6] Coxeter group. This is the pair of conjugacy classes listed as numbers 19 and 20 in Table 5.11. We were able to determine that this pair gives two distinct isomorphism classes, by considering the number of homomorphisms onto PSL(2,11). There are 552 such homomorphisms for representative 19 and 556 for representative 20.

5.3 Results

For completeness we list each of the infinite Coxeter groups listed in Milnor's table in [33], as well as the one other listed in [15], which although not hyperbolic is infinite, and so has torsion-free subgroups from which manifolds may be constructed. For each group we give the minimum index of torsion-free subgroups in column 2, the number of conjugacy classes of minimum-index torsion-free subgroups in column 3, and the number of isomorphism classes of minimum-index torsion-free subgroups in column 4.

$[p, q, r]$	Index	Congugacy classes	Isomorphism classes
$[3, 3, 6]$	24	1	1
$[3, 4, 4]$	48	13	8
$[3, 5, 3]$	120	7	7
$[3, 6, 3]$	12	2	1
$[4, 3, 4]$	48	18	9
$[4, 3, 5]$	240	14	12
$[4, 3, 6]$	48	11	10
$[4, 4, 4]$	16	12	6
$[5, 3, 5]$	120	12	8
$[5, 3, 6]$	120	77	77
$[6, 3, 6]$	24	12	8

Table 5.1: Conjugacy and isomorphism classes of minimum-index torsion-free subgroups of selected Coxeter groups

In Tables 5.2 to 5.12 we list generating sets for minimum-index torsion-free subgroups of the 11 Coxeter groups in Table 5.1. We give one generating set for each conjugacy class of subgroups, as found by the `LowIndAlg`. No attempt has been made to shorten the words used to express the generators in each set. In each case, the abelianisation of the torsion-free subgroup is the first homology group of the corresponding manifold. The abelianisation of each subgroup can be found using the `AQInvariants` command in MAGMA.

The [3,3,6] Coxeter group has exactly one conjugacy class and hence only one isomorphism class of minimum-index torsion-free subgroups, as given in Table 5.2.

Among the 13 conjugacy classes of minimum-index torsion-free subgroups of the [3,4,4] Coxeter group, there are three pairs and one triple containing isomorphic subgroups. The pairs occur in Table 5.3 as numbers 4 and 7, numbers 5 and 9, and numbers 12 and 13. The triple occurs in Table 5.3 as numbers 1, 2 and 11.

Among the seven conjugacy classes of minimum-index torsion-free subgroups of the [3,5,3] Coxeter group, two contain isomorphic subgroups. This pair occurs in Table 5.4 as numbers 6 and 7.

The two conjugacy classes of minimum-index torsion-free subgroups of the [3,6,3] Coxeter group contain isomorphic subgroups. They are both given in Table 5.5.

Among the 18 conjugacy classes of minimum-index torsion-free subgroups of the [4,3,4] Coxeter group, there are two pairs, two triples and one set of four containing isomorphic subgroups. The pairs occur in Table 5.6 as numbers 2 and 10, and numbers 9 and 14. The triples occur in Table 5.6 as numbers 3, 13 and 16, and numbers 4, 8 and 18. The set of four occurs in Table 5.6 as numbers 5, 6, 15 and 17.

Among the 14 conjugacy classes of minimum-index torsion-free subgroups of the [4,3,5] Coxeter group, there are two pairs containing isomorphic subgroups. These pairs occur in Table 5.7 as numbers 4 and 10, and numbers 5 and 13.

Among the 11 conjugacy classes of minimum-index torsion-free subgroups of the [4,3,6] Coxeter group, there is one pair containing isomorphic subgroups. This pair occurs in Table 5.8 as numbers 3 and 4.

Among the 12 conjugacy classes of minimum-index torsion-free subgroups of the [4,4,4] Coxeter group, there are three pairs and one set of four containing isomorphic subgroups. The pairs occur in Table 5.9 as numbers 1 and 8, numbers 4 and 6, and numbers 11 and 12. The set of four occurs in Table 5.9 as numbers 3, 5, 7 and 9.

Among the 12 conjugacy classes of minimum-index torsion-free subgroups of the [5,3,5] Coxeter group, there are four pairs containing isomorphic subgroups. These pairs are given in Table 5.10 as numbers 2 and 4, numbers 6 and 9, numbers 8 and 10, and numbers 11 and 12.

All of the 77 conjugacy classes of minimum-index torsion-free subgroups of the [5,3,6] Coxeter group in Table 5.11 are isomorphism classes of such subgroups.

Among the 12 conjugacy classes of minimum-index torsion-free subgroups of the [6,3,6] Coxeter group, there are four pairs containing isomorphic subgroups. These pairs occur in Table 5.12 as numbers 2 and 9, numbers 3 and 7, numbers 4 and 10, and numbers 6 and 11.

No.	Generating Set	Abelianisation
1	$\{dcdcdba, dcbcb\}$	$\mathbb{Z}$

Table 5.2: Generating sets for minimum-index torsion-free subgroups of [3,3,6]

No.	Generating Set	Abelianisation
1	$\{dcb, cdcbc, abdcdcbca\}$	$\mathbb{Z}_2 \oplus \mathbb{Z} \oplus \mathbb{Z}$
2	$\{dcb, cdcbc, abcdcabca\}$	$\mathbb{Z}_2 \oplus \mathbb{Z} \oplus \mathbb{Z}$
3	$\{dcb, abcdbca\}$	$\mathbb{Z}_4 \oplus \mathbb{Z}$
4	$\{dcb, abdcdbca\}$	$\mathbb{Z} \oplus \mathbb{Z}$
5	$\{dcb, abacdbca\}$	$\mathbb{Z}_8 \oplus \mathbb{Z}$
6	$\{abcbadcdba, cdcb, dcbc\}$	$\mathbb{Z} \oplus \mathbb{Z}$
7	$\{abdcbca, cdcb\}$	$\mathbb{Z} \oplus \mathbb{Z}$
8	$\{cdcb, abdcdbca\}$	$\mathbb{Z} \oplus \mathbb{Z}$
9	$\{abdcbca, cbdcbdcbc, dcdcb\}$	$\mathbb{Z}_8 \oplus \mathbb{Z}$
10	$\{bdcbc, cdcba\}$	$\mathbb{Z}_2 \oplus \mathbb{Z}$
11	$\{bdcbc, acbdcba, dcabdcdba\}$	$\mathbb{Z}_2 \oplus \mathbb{Z} \oplus \mathbb{Z}$
12	$\{bdcbc, dcabadcdba, acbadcba\}$	$\mathbb{Z}_2 \oplus \mathbb{Z}_2 \oplus \mathbb{Z}$
13	$\{dcdbcb, acbadcba, bdcdcbc\}$	$\mathbb{Z}_2 \oplus \mathbb{Z}_2 \oplus \mathbb{Z}$

Table 5.3: Generating sets for minimum-index torsion-free subgroups of [3,4,4]

No.	Generating Set	Abelianisation
1	$\{cdcbadcdcbcbcb, cdbaba\}$	$\mathbb{Z}_{29}$
2	$\{cdcbacdbcdba, cdbaba\}$	$\mathbb{Z}_{35}$
3	$\{cdcbacdbcdcbcbcb, cdbaba, bcbcbacbcdbcdbcdbc\}$	$\mathbb{Z}_{29}$
4	$\{cdcbadcdbcdbcdba, cdbaba, bcbcbadbcdbcba\}$	$\mathbb{Z}_9$
5	$\{dcbcbaba, adcbcdbc\}$	$\mathbb{Z}_{11} \oplus \mathbb{Z}_{11}$
6	$\{dcbcbaba, cbadcdbcba, bcbdcdbcdbca\}$	$\mathbb{Z}_2 \oplus \mathbb{Z}_{18}$
7	$\{dcbcbaba, cbdcbdcdbcba, cbadcdbacdba\}$	$\mathbb{Z}_2 \oplus \mathbb{Z}_{18}$

Table 5.4: Generating sets for minimum-index torsion-free subgroups of [3,5,3]

No.	Generating Set	Abelianisation
1	$\{abdcdc, cdb\}$	$\mathbb{Z}_2 \oplus \mathbb{Z}$
2	$\{cbacdc, bca\}$	$\mathbb{Z}_2 \oplus \mathbb{Z}$

Table 5.5: Generating sets for minimum-index torsion-free subgroups of [3,6,3]

No.	Generating Set	Abelianisation
1	$\{dcba, bcbdcabacd\}$	$\mathbb{Z}_3 \oplus \mathbb{Z}$
2	$\{bcbabcd, cdcba, bcdacbacdc\}$	$\mathbb{Z} \oplus \mathbb{Z}$
3	$\{bcbacbcd, cdcba, bcdacbacdc\}$	$\mathbb{Z}_2 \oplus \mathbb{Z} \oplus \mathbb{Z}$
4	$\{bcbadcdbcd, cdcba, bcdacbacdc\}$	$\mathbb{Z}_4 \oplus \mathbb{Z}$
5	$\{bcbadcdcbcd, cdcba, bcdacbacdc\}$	$\mathbb{Z}_2 \oplus \mathbb{Z}_2 \oplus \mathbb{Z}$
6	$\{bdcdacbacdc, bcbacbcd, cdcba\}$	$\mathbb{Z}_2 \oplus \mathbb{Z}_2 \oplus \mathbb{Z}$
7	$\{bacdcbacdc, bcbadbca, dcdcba\}$	$\mathbb{Z}_4 \oplus \mathbb{Z}_4$
8	$\{bacdcbacdc, bcbadcbca, dcdcba\}$	$\mathbb{Z}_4 \oplus \mathbb{Z}$
9	$\{bacdcbacdc, dcdcba, bcbacdcbca\}$	$\mathbb{Z}_2 \oplus \mathbb{Z}_2 \oplus \mathbb{Z}$
10	$\{bcbabcd, bcdcba, babcdc\}$	$\mathbb{Z} \oplus \mathbb{Z}$
11	$\{bcbadbcd, bcdcba, babcdc\}$	$\mathbb{Z}_2 \oplus \mathbb{Z}$
12	$\{bcdcba, babcdc, bcbacbcd\}$	$\mathbb{Z} \oplus \mathbb{Z} \oplus \mathbb{Z}$
13	$\{bcbadcbcd, bcdcba, babcdc\}$	$\mathbb{Z}_2 \oplus \mathbb{Z} \oplus \mathbb{Z}$
14	$\{bcbadcdbcd, bcdcba, babcdc\}$	$\mathbb{Z}_2 \oplus \mathbb{Z}_2 \oplus \mathbb{Z}$
15	$\{bcbadcbcd, badbcdc, bcdcba\}$	$\mathbb{Z}_2 \oplus \mathbb{Z}_2 \oplus \mathbb{Z}$
16	$\{bcbadcbcd, bcdcba, bacdcbcdc\}$	$\mathbb{Z}_2 \oplus \mathbb{Z} \oplus \mathbb{Z}$
17	$\{bcbadcdbcd, bcdcba, bacdcbcdc\}$	$\mathbb{Z}_2 \oplus \mathbb{Z}_2 \oplus \mathbb{Z}$
18	$\{bcbadbca, badbcdc, bdcdcba\}$	$\mathbb{Z}_4 \oplus \mathbb{Z}$

Table 5.6: Generating sets for minimum-index torsion-free subgroups of [4,3,4]

No.	Generating Set	Abelianisation
1	$\{bcbadcdcdbacdc,\ cdcbadcdcbcdcdbcdcab,\ dcba\}$	$\mathbb{Z}_{28} \oplus \mathbb{Z}$
2	$\{bcbacdcbcdcdcbcb,\ dcba,\ cdcabacdcdbcdcdabacd\}$	$\mathbb{Z}_4 \oplus \mathbb{Z}_4 \oplus \mathbb{Z}$
3	$\{dcbcdcba,\ cdcbcdab,\ dcabadcdcdbacdcbab\}$	$\mathbb{Z}_2 \oplus \mathbb{Z}_2 \oplus \mathbb{Z}$
4	$\{bcbadcdcdbacdc,\ dcdbcdcba,\ adcdcbcdab\}$	$\mathbb{Z}_{13} \oplus \mathbb{Z}$
5	$\{bcdcdbcdcba,\ dcdcbcdaba,\ bcbadcdbcdcdab\}$	$\mathbb{Z}_4 \oplus \mathbb{Z}_4 \oplus \mathbb{Z}$
6	$\{cdcabacdcdbcdcdc, badcba, bcbacdcdbacd,$ $bcdcbacbcdcdba\}$	$\mathbb{Z}_{11} \oplus \mathbb{Z}_{11}$
7	$\{bcbadcdcdbacdc,\ dcabacdcdbcdcdbcdc,\ badcba\}$	$\mathbb{Z}_2 \oplus \mathbb{Z}$
8	$\{badcba,\ dcabacdcbcdcdbcd,\ bcbadcdcdbacdcdc\}$	$\mathbb{Z}_{87}$
9	$\{bcbacbacdcdc,\ dcbadcdbcdcdbcdcba,\ badcdcba\}$	$\mathbb{Z}_4 \oplus \mathbb{Z}_{28}$
10	$\{dcbadcdbacdc,\ bacdcbca,\ bcbacdcbacd\}$	$\mathbb{Z}_{13} \oplus \mathbb{Z}$
11	$\{bacdcbca, abadcdbcdcdbcd, dcbadcdbcdcdbcdcba,$ $bcbadcdcdbacdcdc\}$	$\mathbb{Z}_2 \oplus \mathbb{Z}$
12	$\{abadcdcdbcdcdbcd, dcbadcdbacdc, bacdcbca,$ $bcbacdcbcdcdbacd\}$	$\mathbb{Z}_2 \oplus \mathbb{Z}_2 \oplus \mathbb{Z}_2 \oplus$ $\mathbb{Z}_2$
13	$\{bcbacdcbcdcdbacdc, dcbadcdbacdc, bacdcbca,$ $abadcbcdcdbcd\}$	$\mathbb{Z}_4 \oplus \mathbb{Z}_4 \oplus \mathbb{Z}$
14	$\{abadbcdcdbcdcb, bcbacdcdbacd, dcbadcdbcdcdba,$ $badcdcdbca\}$	$\mathbb{Z}_3 \oplus \mathbb{Z}_{15}$

Table 5.7: Generating sets for minimum-index torsion-free subgroups of [4,3,5]

No.	Generating Set	Abelianisation
1	$\{adcdcba, dcbcb\}$	$\mathbb{Z}_8 \oplus \mathbb{Z}$
2	$\{acdcdcba, dcbcb\}$	$\mathbb{Z} \oplus \mathbb{Z}$
3	$\{abcdcdacdbca, dcbcb, dcabcdcdbca\}$	$\mathbb{Z}_2 \oplus \mathbb{Z}_2 \oplus \mathbb{Z}$
4	$\{cdcdcb, dcdcbcb, abacbacdc, cdcdcabdcdcdbacdc, abdcdcbca\}$	$\mathbb{Z}_2 \oplus \mathbb{Z}_2 \oplus \mathbb{Z}$
5	$\{dcdcbcb, cdcdcba, bacbacdc, cdcdcabdcdcdbacdc\}$	$\mathbb{Z}_2 \oplus \mathbb{Z}_2 \oplus \mathbb{Z}$
6	$\{babcdbca, dcdcbcb, acdcdcba, dcabdcdcdbca, cdcdcbcdcdcb\}$	$\mathbb{Z}_2 \oplus \mathbb{Z} \oplus \mathbb{Z}$
7	$\{badcdcbca, dcdcbcb, adcdcdcba, dcabdcdcdbca, cdcdcbcdcdcb\}$	$\mathbb{Z}_2 \oplus \mathbb{Z}_4 \oplus \mathbb{Z}$
8	$\{(dcb)^2, cdcdcb, abadcbca\}$	$\mathbb{Z}_2 \oplus \mathbb{Z} \oplus \mathbb{Z}$
9	$\{cdcbdcdcdb, babcdbca, (dcb)^2, acdcdcba, dcabdcdcdbca\}$	$\mathbb{Z}_2 \oplus \mathbb{Z} \oplus \mathbb{Z}$
10	$\{dcba, dcdcdcbcb\}$	$\mathbb{Z}_4 \oplus \mathbb{Z}$
11	$\{dcabcdcdcbcb, bcdbcaba, badc\}$	$\mathbb{Z}_2 \oplus \mathbb{Z}_4 \oplus \mathbb{Z}$

Table 5.8: Generating sets for minimum-index torsion-free subgroups of [4,3,6]

No.	Generating Set	Abelianisation
1	$\{abcd, cdb\}$	$\mathbb{Z}_8 \oplus \mathbb{Z}$
2	$\{acbdcdc, abacd, cdb\}$	$\mathbb{Z}_4 \oplus \mathbb{Z}$
3	$\{abacd, acbcadc, cdb\}$	$\mathbb{Z}_2 \oplus \mathbb{Z} \oplus \mathbb{Z}$
4	$\{abcdc, cdb\}$	$\mathbb{Z} \oplus \mathbb{Z}$
5	$\{acbacd, abcadc, cdb\}$	$\mathbb{Z}_2 \oplus \mathbb{Z} \oplus \mathbb{Z}$
6	$\{bca, cbcd\}$	$\mathbb{Z} \oplus \mathbb{Z}$
7	$\{bca, cbdcd, abdacdc\}$	$\mathbb{Z}_2 \oplus \mathbb{Z} \oplus \mathbb{Z}$
8	$\{cbacdc, bca\}$	$\mathbb{Z}_8 \oplus \mathbb{Z}$
9	$\{abdcd, bca, cbdacdc\}$	$\mathbb{Z}_2 \oplus \mathbb{Z} \oplus \mathbb{Z}$
10	$\{badc, cbcd\}$	$\mathbb{Z} \oplus \mathbb{Z}$
11	$\{abcdc, badc, cbacd\}$	$\mathbb{Z}_2 \oplus \mathbb{Z}_2 \oplus \mathbb{Z}$
12	$\{cbdcd, badc, abdacdc\}$	$\mathbb{Z}_2 \oplus \mathbb{Z}_2 \oplus \mathbb{Z}$

Table 5.9: Generating sets for minimum-index torsion-free subgroups of [4,4,4]

No.	Generating Set	Abelianisation
1	$\{dcba, badcbdcb, dcabacdcdbacdcdc\}$	$\mathbb{Z}_{35}$
2	$\{bacdcdbcdc, bcbacdcbacdcdc, dcba,$ $dcabadcdcbcdcb\}$	$\mathbb{Z}_{48}$
3	$\{bcbadbcdcdbacd, dcba, badbcdcb\}$	$\mathbb{Z}_{29}$
4	$\{bcbadbacdc, dcba, badbcdcdcbcb\}$	$\mathbb{Z}_{48}$
5	$\{bcbadbacdc, dcabadcdcdbcdcdbacdc, dcdcba,$ $badcdbca\}$	$\mathbb{Z}_{15} \oplus \mathbb{Z}_{15}$
6	$\{bcbadcdcdbacdc, dcdcba, badcdbca\}$	$\mathbb{Z}_5 \oplus \mathbb{Z}_5 \oplus \mathbb{Z}_5$
7	$\{abadcdcbcdcdcbcb, bdcdcbca, bacdbacd,$ $bcbadcdbcdcdbacd\}$	$\mathbb{Z}_5 \oplus \mathbb{Z}_{15}$
8	$\{bcbacbcdcdbacd, bdcdcbca, abadcbacdcdc,$ $badcbcdc\}$	$\mathbb{Z}_3 \oplus \mathbb{Z}_3$
9	$\{bdcdcbca, bcbadcdbcdcdbacd, badcbcdc\}$	$\mathbb{Z}_5 \oplus \mathbb{Z}_5 \oplus \mathbb{Z}_5$
10	$\{abadcdcdbcdc, bacdcdbcdcdc, bdcdcbca,$ $bcbadcdcdbacdcdc\}$	$\mathbb{Z}_3 \oplus \mathbb{Z}_3$
11	$\{bdcdcbacdc, bacdbacd, bcbadcdcba, acbacdbcdc\}$	$\mathbb{Z}_5 \oplus \mathbb{Z}_5 \oplus \mathbb{Z}_5$
12	$\{bcbadbacdc, bcdcbdcdcdbacd, badcdbca, abacdbcd\}$	$\mathbb{Z}_5 \oplus \mathbb{Z}_5 \oplus \mathbb{Z}_5$

Table 5.10: Generating sets for minimum-index torsion-free subgroups of [5,3,5]

No.	Generating Set	Abelianisation
1	$\{dcbdcdbcdcd, adcbcdcba, ababdcdcbcdcaba, dcbcb,$ $adcdcdcbcdcdbacdc, abacbacbcdcdcaba\}$	$\mathbb{Z}_2 \oplus \mathbb{Z}_2 \oplus \mathbb{Z}_4 \oplus$ $\mathbb{Z}$
2	$\{adcdcbcdcba, ababdcdcdbcdcaba, dcbdcdbcdcd,$ $dcbcb, abacbadcbcbababa, abacbcdcdbcdaba,$ $(acdcdba)^2\}$	$\mathbb{Z}_2 \oplus \mathbb{Z}_2 \oplus \mathbb{Z}_2 \oplus$ $\mathbb{Z}_2 \oplus \mathbb{Z}$
3	$\{ad(cdcb)^2a, abcdcadcbaba, dcbdcdbcdcd, dcbcb,$ $cdcabdcdcdcdbcdcdcbacdcdc, ababdcdbcdaba,$ $cdcabcacbdcdbcdcacbacdc\}$	$\mathbb{Z}_2 \oplus \mathbb{Z}_2 \oplus \mathbb{Z} \oplus \mathbb{Z}$
4	$\{adcdcbcdcba, abcdcadcbaba, dcbcb, dcbdcdbcdcd,$ $cdcabcacbdcdbcdaba, cdcabdcdcdcdbcdcdcbacdcdc\}$	$\mathbb{Z}_2 \oplus \mathbb{Z}_2 \oplus \mathbb{Z} \oplus \mathbb{Z}$
5	$\{adcdcbcdcba, dcbdcdbcdcd, cdcabcdcadcbaba,$ $dcbcb, cdcabdcdcdcdbcdcdcbacdcdc\}$	$\mathbb{Z}_2 \oplus \mathbb{Z}_2 \oplus \mathbb{Z} \oplus \mathbb{Z}$
6	$\{adcdcdbcdcba, abcdcadcbaba, dcbdcdbcdcd, dcbcb,$ $ababdcdbcdaba, dca(bcdcdc)^2bacdc\}$	$\mathbb{Z}_2 \oplus \mathbb{Z}_2 \oplus \mathbb{Z} \oplus \mathbb{Z}$
7	$\{abcdcadcbaba, dcbdcdbcdcd, acdcbcdcdba, dcbcb,$ $dcdcabd(cdcdcb)^2acdcdc\}$	$\mathbb{Z}_2 \oplus \mathbb{Z}_2 \oplus \mathbb{Z} \oplus \mathbb{Z}$
8	$\{ababadcdbcdbababa, dcbdcdbcdcd(acdcdba)^2,$ $ac(bcdcdc)^2bacdcd, acbcdcdcdbaba, dcbcb\}$	$\mathbb{Z}_2 \oplus \mathbb{Z}_2 \oplus \mathbb{Z} \oplus \mathbb{Z}$
9	$\{acbd(cdcdcb)^2acdcd, ababadcdbcdbababa, dcbcb,$ $ababab cbdcbdcdbcdcbdcbababa, dcbdcdbcdcd,$ $acbcdcdcdbaba, (acdcdba)^2\}$	$\mathbb{Z}_2 \oplus \mathbb{Z}_2 \oplus \mathbb{Z} \oplus \mathbb{Z}$
10	$\{acbd(cdcdcb)^2acdcd, dcbdcdbcdcd, acbcdcdcdbaba,$ $ababab cbdcdcdbcdbababa, dcbcb, (acdcdba)^2\}$	$\mathbb{Z}_2 \oplus \mathbb{Z}_2 \oplus \mathbb{Z} \oplus \mathbb{Z}$

11	$\{a(cbcdcd)^3cbacdcd, dcabadcdbcdcdcbca, dcbcb,$ $dcbdcdbcdcd, acdcdcbcdcdba, abcdcbcaba,$ $acbacbcdcbacdcbca\}$	$\mathbb{Z}_2 \oplus \mathbb{Z}_2 \oplus \mathbb{Z}_8 \oplus$ $\mathbb{Z}$
12	$\{acbdcdcdcbcdcdbacdc, dcbdcdbcdcd, dcbcb,$ $acbcdcdcdbaba, abcdcdcbacdcd\}$	$\mathbb{Z}_2 \oplus \mathbb{Z}_4 \oplus \mathbb{Z}$
13	$\{adcdcba,\ dcbcb,\ cdcabcdcdcabcdcd\}$	$\mathbb{Z}_{12} \oplus \mathbb{Z}$
14	$\{dcdcbabdcdcbcdcdcbacdc, adcdcba, dcbcb,$ $dcdcbacdcdcbcabcdcd\}$	$\mathbb{Z}_{12} \oplus \mathbb{Z}$
15	$\{dcabdcdcdbcdacbcdcd, dcababcdbcdcdbcacbcdcd,$ $acdcdcba, dcdcbca(cbdcd)^5cbdcb(dcdbc)^5acbcdcd,$ $dcdcdbcdcdbca, dcdcbcacbab(dcdbc)^4acbcdcd,$ $dcbcb\}$	$\mathbb{Z}$
16	$\{dcdcbcacba(cdcdb)^4cacbcdcd, acdcdcba,$ $dcabdcdcdbcdacbcdcd, dcababdcdbcdcdbcacbcdcd,$ $dcdcbca(cbdcd)^5cbdcb(dcdbc)^5acbcdcd, dcbcb,$ $dcdcdbcdcdbca\}$	$\mathbb{Z}$
17	$\{dcdcbcacbadcdbcdcacbcdcd, acdcdcba, dcbcb,$ $dcdcbdcbdcd(bcdcd)^2, abdcdbcdcdbca\}$	$\mathbb{Z}_2 \oplus \mathbb{Z}_8 \oplus \mathbb{Z}$
18	$\{acdcdcba, dcabdcbdcd(bcdcd)^2, dcbcb,$ $(dcdcbca)^2cbcdcd\}$	$\mathbb{Z}$
19	$\{dcdcbcdabdcdcdbcdcadcbcdcd, (adcdbca)^2, dcbcb,$ $adcdcdcba\}$	$\mathbb{Z}_2 \oplus \mathbb{Z}_8 \oplus \mathbb{Z}$
20	$\{(adcdbca)^2, dcdcbcdacdcbdcdacbcdcbadcbcdcd,$ $adcdcdcba, dcbcb, dcdcbcdacbdcdcdbcdadcbcdcd\}$	$\mathbb{Z}_2 \oplus \mathbb{Z}_8 \oplus \mathbb{Z}$

21	$\{adcdcbcdcba, acdcdbcdcdba, dcb(dcdbc)^3dcd,$ $dcbcb, ababdcdbcaba, aba(dcdcb)^2aba\}$	$\mathbb{Z}_2 \oplus \mathbb{Z}_2 \oplus \mathbb{Z} \oplus \mathbb{Z}$
22	$\{adcdcbcdcba, acdcdbcdcdba, abadcbdcdcdbcaba,$ $abacdcbdcdcdbaba, dcb(dcdbc)^3dcd, dcbcb,$ $dcdcbacbcaba\}$	$\mathbb{Z}_2 \oplus \mathbb{Z}_2 \oplus \mathbb{Z} \oplus \mathbb{Z}$
23	$\{adcdcbcdcba, ababcdbcaba, dcb(dcdbc)^3dcd, dcbcb,$ $abcdcdcababa, cdcdca(bcdcdc)^2bacdc\}$	$\mathbb{Z}_2 \oplus \mathbb{Z}_6 \oplus \mathbb{Z}$
24	$\{adcdcbcdcba, abababcbdcacdbacdcdc,$ $dcbcb, cdcabdcdcdcaba, dcb(dcdbc)^3dcd,$ $cdcdca(bcdcdc)^2bacdc\}$	$\mathbb{Z}_2 \oplus \mathbb{Z}_2 \oplus \mathbb{Z}$
25	$\{adcdcbcdcba, cdcabadcdbcdcd, dcbcb,$ $aba(cdbcd)^3cd, (dcdcb)^4dcbdcd(bcdcd)^3,$ $cdcdca(bcdcdc)^2bacdc\}$	$\mathbb{Z}_2 \oplus \mathbb{Z}_2 \oplus \mathbb{Z}$
26	$\{adcdcdbcdcba, dcb(dcdbc)^3dcd, acdcbcdcdba,$ $dcbcb, ababdcdbcaba, aba(dcdcb)^2aba\}$	$\mathbb{Z}_2 \oplus \mathbb{Z}_2 \oplus \mathbb{Z} \oplus \mathbb{Z}$
27	$\{adcdcdbcdcba, abcdcdcdababa, dcb(dcdbc)^3dcd,$ $(abadcbdcaba)^2, dcbcb, cdcabd(cdcdcb)^2acd\}$	$\mathbb{Z}_2 \oplus \mathbb{Z}_2 \oplus \mathbb{Z}$
28	$\{adcdcdbcdcba, dcb(dcdbc)^3dcd, dcbcb,$ $dcabcdcdcdbababa, cdcabdcdcdcbcdcdcbacd\}$	$\mathbb{Z}_2 \oplus \mathbb{Z}_6 \oplus \mathbb{Z}$
29	$\{adcdcdbcdcba, abadcdbacdc, acbabdcdbcdcd,$ $(dcdcb)^4dcb(dcdbc)^3dcd, cdcabd(cdcdcb)^2acd,$ $dcbcb\}$	$\mathbb{Z}_2 \oplus \mathbb{Z}_2 \oplus \mathbb{Z}$
30	$\{adcdcdbcdcba, acb(dcdcbc)^2dcdcbacd,$ $dcbcb, dcdcbdcbdcd(bcdcd)^2, cdcabdcdaba,$ $acbacbdcbadcdcdcbca\}$	$\mathbb{Z}_2 \oplus \mathbb{Z}_4 \oplus \mathbb{Z}$

31	$\{dcdcbdcbdcd(bcdcd)^2, cdcdcabd(cdcdcb)^2acdcd,$ $dcbcb, abdcdcbacdc, abacbcdcaba, abcdcdcababa\}$	$\mathbb{Z}_4 \oplus \mathbb{Z}_4 \oplus \mathbb{Z}$
32	$\{dcdcbdcbdcd(bcdcd)^2, dcdcabdcdaba, dcbcb,$ $cdcdcabd(cdcdcb)^2acdcd, abdcdcbacdc\}$	$\mathbb{Z}_4 \oplus \mathbb{Z}_4 \oplus \mathbb{Z}$
33	$\{ad(cdcdb)^2acd, dcdcbdcbdcd(bcdcd)^2, dcbcb,$ $abdcdcdbacdc, ab(cdcbcd)^2cdcbd(cdcdcb)^3ca,$ $abadcdbcdcba, acbacdcd(bcdcdc)^2bca\}$	$\mathbb{Z}_4 \oplus \mathbb{Z}$
34	$\{abdcdcdcbacdc, dcb(dcdbc)^3dcd, dcbcb,$ $acbcdcdcdbaba, dcdca(bcdcdc)^2bca\}$	$\mathbb{Z}_2 \oplus \mathbb{Z}_4 \oplus \mathbb{Z}$
35	$\{abdcdcdcbacdc, ababacdcd, dcbcb,$ $dcdca(bcdcdc)^2bca, (dcdcb)^4dcb(dcdbc)^3dcd\}$	$\mathbb{Z}_{12} \oplus \mathbb{Z}$
36	$\{abdcdcdcbacdc, acbadc(bcdcd)^2, dcbcb,$ $abacbacdcd, dcdca(bcdcdc)^2bca,$ $(dcdcb)^4dcb(dcdbc)^3dcd\}$	$\mathbb{Z}_{12} \oplus \mathbb{Z}$
37	$\{cdcabadcbdcdbcdcd, abadcdbacdcd, dcbcb,$ $abcdcdcbacdcdc, dca(bdcdc)^2dbcdcdbca,$ $(dcdcb)^2dcbdcdbcdcd\}$	$\mathbb{Z}_2 \oplus \mathbb{Z} \oplus \mathbb{Z}$
38	$\{dcdcabdcdcdcbcdcba, cdcabdcdcdcbcdcdba,$ $dcbcb, dcabacbdcdbcdcd, (dcdcb)^2dcbdcdbcdcd,$ $abadcdcbca\}$	$\mathbb{Z}_2 \oplus \mathbb{Z} \oplus \mathbb{Z}$
39	$\{ad(cdcdb)^2acd, cdcdcb, ababcdcdcdacbcdcaba,$ $abacbacdbcdcdcaba, dcdcbcb, abacbdcdcdbcdaba,$ $acdbcdcba\}$	$\mathbb{Z}_2 \oplus \mathbb{Z}_2 \oplus \mathbb{Z} \oplus \mathbb{Z}$
40	$\{adcdcbcdcba, cdcdcb, abacdcbadcdcdbaba,$ $dcdcbcb, abacbadcbcdaba, (acdcdba)^2\}$	$\mathbb{Z}_2 \oplus \mathbb{Z}_2 \oplus \mathbb{Z}_4 \oplus \mathbb{Z}$

41	$\{bdcdcdcbca, dcdcbcb, bacdcdcbcdcbdcdcdcbab,$ $babacdcdbcdcab, ba(dcdcbc)^2ab, d(cdcdcb)^2acdcd\}$	$\mathbb{Z}_2 \oplus \mathbb{Z}_6 \oplus \mathbb{Z}$
42	$\{a(bdcdc)^2bcdcdcbdcdcdc(bcdcd)^3ba, dcdcbcb,$ $ba(dcdcbc)^2dcdba, abad(cdcdb)^3a, (bcdcdc)^2,$ $adcdcdbca\}$	$\mathbb{Z}_2 \oplus \mathbb{Z}_4 \oplus \mathbb{Z}$
43	$\{dcdcbcb, acbd(cdcdcb)^2cdcdbca, (bcdcdc)^3,$ $acbcdcdcadbcdcdc, adcdcdcba\}$	$\mathbb{Z}_2 \oplus \mathbb{Z}_{12} \oplus \mathbb{Z}$
44	$\{dcdcbcb, acbd(cdcdcb)^2cdcdbca, (bcdcdc)^3,$ $acbdcdcdcadbcdcdc, adcdcdcba\}$	$\mathbb{Z}_2 \oplus \mathbb{Z}_{16} \oplus \mathbb{Z}$
45	$\{badcbdcdcdbab, dcdcbcb, adcdcdbca, (bcdcdc)^3,$ $abd(cdcdcb)^2cdcdba, ba(cdcbd)^2cab\}$	$\mathbb{Z}_2 \oplus \mathbb{Z}_{48} \oplus \mathbb{Z}$
46	$\{dcabdcdcbcdcdbacdcd, dcdcbcb, (cdcdcb)^6,$ $badcdbcdcdcbcdcdc, abadcdcdcbcdcdcb,$ $abdcdcdcbca\}$	$\mathbb{Z}_{12} \oplus \mathbb{Z}$
47	$\{(dcb)^2, adcdcbcdcba, cdcdcb, abadcbcdcdba,$ $cdca(bdcdc)^3dcdbcdcdcbacdcdc,$ $acbadbcdcdcbacdcdc\}$	$\mathbb{Z}_2 \oplus \mathbb{Z}_2 \oplus \mathbb{Z} \oplus \mathbb{Z}$
48	$\{(dcb)^2, bdcdcdcbca, dcabdcdcadcdcbcdab,$ $cbdcdbcdcdbacd, bacbdcadcdbcdcab\}$	$\mathbb{Z}_2 \oplus \mathbb{Z} \oplus \mathbb{Z}$
49	$\{(dcb)^2, bdcdcdbcdc, abadbcdcdbca,$ $bacdcbcdcdcbca, abcdcdcbacdcdc,$ $dca(bdcdc)^2dbcdcdbca\}$	$\mathbb{Z}_6 \oplus \mathbb{Z} \oplus \mathbb{Z}$
50	$\{dca(bdcdc)^2dcbcdcdcbca, (dcb)^2, bdcdcdbcdc,$ $badbca, abcdcdcbacdcdc, dcabacdbacdc\}$	$\mathbb{Z}_2 \oplus \mathbb{Z}_2 \oplus \mathbb{Z} \oplus \mathbb{Z}$

51	$\{acbcdcdcbacdcd, (dcb)^2, bdcdcdbcdc,$ $badcdcdcbcdcdcba, abacdbacdcdc,$ $cdca(bdcdc)^2dbcdcdba\}$	$\mathbb{Z}_2 \oplus \mathbb{Z} \oplus \mathbb{Z}$
52	$\{bd(cdcdb)^4acdcd, cdcdcabacdcdcbcdcdbacdcd,$ $dcdcdcbcb, cdcba\}$	$\mathbb{Z}_3 \oplus \mathbb{Z} \oplus \mathbb{Z}$
53	$\{dcdca(bdcdc)^2bcdcdcbdcdcdc(bcdcd)^3bacdcd,$ $badcdc(bcdcd)^2bacdcd, dcdcdcbcb, cdcba\}$	$\mathbb{Z}_{15} \oplus \mathbb{Z}$
54	$\{dcdcabdcdcbcdcdcbdcdcdc(bcdcd)^2bacdcd,$ $ba(dcdbc)^2dcdbacdcd, dcdcdcbcb, bdcdcba,$ $acbadcbcdcdbacdcd\}$	$\mathbb{Z}_2 \oplus \mathbb{Z}_{10} \oplus \mathbb{Z}$
55	$\{bdcdcdba, cdcabdcdcbcdcdcbdcdcdc(bcdcd)^2bacdc,$ $dcdcdcbcb, bacdcbcd, acbabcdcdbacdc\}$	$\mathbb{Z}_2 \oplus \mathbb{Z}_{10} \oplus \mathbb{Z}$
56	$\{bdcdcdba, cdcabdcdcbcdcdcbdcdcdc(bcdcd)^2bacdc,$ $acbadcbcdcdcbacdc, cdcabadbcdcdbacdc, dcdcdcbcb,$ $bacdcbcd\}$	$\mathbb{Z}_2 \oplus \mathbb{Z}_{10} \oplus \mathbb{Z}$
57	$\{bdcdcdcba, dcbacbdcdcdcbcdabcd, adcdcbab,$ $acdcdcbcdcdcbacdc, dcdcdcbcb, cdcabacdcbabcd,$ $dcbadcdcbdcdcdcbcdcabcd\}$	$\mathbb{Z}_4 \oplus \mathbb{Z}$
58	$\{adcdcbcdcba, bcdcdcbcd, cdcdcabdcdcbab,$ $cdcabdcdaba, dcdcdcbcb, cdcdca(bcdcdc)^2bacdc\}$	$\mathbb{Z}_2 \oplus \mathbb{Z}_6 \oplus \mathbb{Z}$
59	$\{adcdcdbcdcba, abcdcdcdababa, bcdcdcbcd,$ $acbdcdcbab, dcdcdcbcb, cdcabd(cdcdcb)^2acd\}$	$\mathbb{Z}_2 \oplus \mathbb{Z}_6 \oplus \mathbb{Z}$
60	$\{bcdcdcbcd, dcabacdbcdcdcbca, bacdbca,$ $abcdcdcbacdcdc, dcdcdcbcb, dca(bdcdc)^2dbcdcdbca\}$	$\mathbb{Z} \oplus \mathbb{Z}$

61	$\{bacbcdcdb, abad(cdcdcb)^2, dcdcdcbcb,$ $dcdca(bdcdc)^2dcbcdcdb, abdcdcdcbca\}$	$\mathbb{Z}_4 \oplus \mathbb{Z}$
62	$\{dcdca(bdcdc)^2dbcdcba, bacbacdcd, abdcdcbacdc,$ $(bdcdc)^2dcbcdcdcb, dcdcdcbcb\}$	$\mathbb{Z}_2 \oplus \mathbb{Z}_8 \oplus \mathbb{Z}$
63	$\{badbca, abdcdcdbacdc, a(cbcdcd)^2cbacdcd,$ $(bdcdc)^2dcbcdcdcb, dcdcdcbcb\}$	$\mathbb{Z}_4 \oplus \mathbb{Z} \oplus \mathbb{Z}$
64	$\{bacdcbacd, abcdcdbacdcd, abadbacdc,$ $acbdcdcbcdcdcbcdcdba, (bdcdc)^2dcbcdcdcb,$ $dcdcdcbcb\}$	$\mathbb{Z}_4 \oplus \mathbb{Z} \oplus \mathbb{Z}$
65	$\{bcdcdcbdcdcdcbcdcdb, a(cbdcd)^2cbacdcdc,$ $bacdcbacd, abcdcdcbacdcd, dcdcdcbcb, abadcbacdc\}$	$\mathbb{Z}_{12} \oplus \mathbb{Z}$
66	$\{bacdcbacdc, cdca(bcdcd)^2b, abcdcdcbacdcdc,$ $dcabcdcdcbdcdcdcbcdcdbacd, abacdcdbacd,$ $dcdcdcbcb\}$	$\mathbb{Z}$
67	$\{bcdbcdacdcbcab, cdcbacdcdc, dcba,$ $dcdcdcdbcdcdcb, badcbdcadcdbcdcdab\}$	$\mathbb{Z}_2 \oplus \mathbb{Z} \oplus \mathbb{Z}$
68	$\{dca(bdcdc)^2bcdcdcb, badcdcdbacd, dcba,$ $bcbcdcdcbacdcdc\}$	$\mathbb{Z}_{24} \oplus \mathbb{Z}$
69	$\{dca(bdcdc)^2bcdcdcb, bacdcdbcdc, dcba,$ $bcbdcdcdcbacdcdc, bcba(cdcb)^2\}$	$\mathbb{Z}_2 \oplus \mathbb{Z}$
70	$\{bcbacdcbcdcdc, dca(bdcdc)^2dbacdcdc, bacdcdbacd,$ $dcba, bcdcbdcdcbcdc\}$	$\mathbb{Z}_2 \oplus \mathbb{Z}_2 \oplus \mathbb{Z}$
71	$\{dca(bdcdc)^2dbacdcdc, babcdcd, cdcba,$ $(bcdc)^2dcbcd, bcbadcdcdbacdcdc\}$	$\mathbb{Z}_2 \oplus \mathbb{Z}_2 \oplus \mathbb{Z}$

72	$\{dcdcba, (bdcdc)^3 dcbcdcdbacd, cbacdcdc, babcdc, acbacbcdcdbacd\}$	$\mathbb{Z}_2 \oplus \mathbb{Z}_2 \oplus \mathbb{Z}$
73	$\{bacdbcdcdb, dcabadcdcbcdcdcb, dcdcba, bdcdcbdcdcbdcdcdcbcdcdbacd, cbacdcdc, acbacbcdcdbacd\}$	$\mathbb{Z}_2 \oplus \mathbb{Z}_2 \oplus \mathbb{Z}$
74	$\{acbacdcbcdcdcb, dcdcba, (bdcdc)^3 dbcdcdcbacd, cbacdcdc, babcdc, dcabadbcdcdcbacd\}$	$\mathbb{Z}_2 \oplus \mathbb{Z}_6 \oplus \mathbb{Z} \oplus \mathbb{Z}$
75	$\{dcdcba, dcabadcbcdcdcb, (bdcdc)^3 dbcdcdcbacd, cbacdcdc, babcdc, acbacdbcdcdcbacd\}$	$\mathbb{Z}_2 \oplus \mathbb{Z}_2 \oplus \mathbb{Z} \oplus \mathbb{Z}$
76	$\{bcdcbdcdcdcbacd, bdcdcbdcdcdcbacdcdc, dcdcba, babacdc, bcbadcdcdcbca\}$	$\mathbb{Z}_2 \oplus \mathbb{Z}_6 \oplus \mathbb{Z}$
77	$\{bcdcdbacd, cdcdcdbcdcdcbcb, bacdcbdcdcdbab, ba(dcdcbc)^2 ab, bdcdcbca, bcbadbab\}$	$\mathbb{Z}_2 \oplus \mathbb{Z}_2 \oplus \mathbb{Z} \oplus \mathbb{Z}$

Table 5.11: Generating sets for minimum-index torsion-free subgroups of [5,3,6]

No.	Generating Set	Abelianisation
1	$\{cdcabdcdcb, abacd, dcbcb\}$	$\mathbb{Z}_4 \oplus \mathbb{Z}$
2	$\{cdcabacdcb, abacd, dcbcb\}$	$\mathbb{Z}_2 \oplus \mathbb{Z} \oplus \mathbb{Z}$
3	$\{abdacdc, acbacdcd, dcbcb\}$	$\mathbb{Z}_2 \oplus \mathbb{Z}_4 \oplus \mathbb{Z}$
4	$\{abdcdcb, dcbcb\}$	$\mathbb{Z}_4 \oplus \mathbb{Z}$
5	$\{cdcabacdcb, cdcdb, adcbcb\}$	$\mathbb{Z}_2 \oplus \mathbb{Z}_4 \oplus \mathbb{Z}$
6	$\{acbcdacdc, cdcdb, abdcb\}$	$\mathbb{Z}_2 \oplus \mathbb{Z}_2 \oplus \mathbb{Z}$
7	$\{acbcdacdc, cdcdb, adcdcbcb\}$	$\mathbb{Z}_2 \oplus \mathbb{Z}_4 \oplus \mathbb{Z}$
8	$\{cdcdb, dcbdcb, abcdacdc, acbadcd\}$	$\mathbb{Z}_2 \oplus \mathbb{Z}_2 \oplus \mathbb{Z}_2 \oplus \mathbb{Z}$
9	$\{bca, cdcdcabdcdc, dcbdcb\}$	$\mathbb{Z}_2 \oplus \mathbb{Z} \oplus \mathbb{Z}$
10	$\{bcbcdcdcd, bca, dcbacdcd\}$	$\mathbb{Z}_4 \oplus \mathbb{Z}$
11	$\{badc, dcbdcb, dcabacdcb\}$	$\mathbb{Z}_2 \oplus \mathbb{Z}_2 \oplus \mathbb{Z}$
12	$\{bacdc, dcbdcb, dcabcdcdc\}$	$\mathbb{Z} \oplus \mathbb{Z} \oplus \mathbb{Z}$

Table 5.12: Generating sets for minimum-index torsion-free subgroups of [6,3,6]

The above results have been published in [11]. An extended version, including details about determining isomorphisms between conjugacy classes of minimum-index torsion-free subgroups for each of the 11 Coxeter groups from Table 5.1 is available at `github.com/GLiversidge/LowIndAlg`.

5.4 Comparable algorithms

The `LowIndAlg` function is to the best of the author's knowledge the only such function available in Magma. In our application of `LowIndAlg` we compared our results to those found via alternative methods using the Low Index Subgroups algorithm available in Magma. The most notable time difference occurred in a case where the Low Index Subgroups algorithm completed after over 12 weeks, whereas `LowIndAlg` completed the task within 7 hours, as explained in Section 5.2.

We were unaware of the option to exclude words in the implementation of the low index algorithm in GAP. We expected this GAP option to be very similar to `LowIndAlg`, but our testing revealed that `LowIndAlg` often runs far more quickly.

In five of the eleven cases, both `LowIndAlg` and the GAP function `LowIndexSubgroupsFpGroup` took less than a second, but in the other six cases, namely [3,4,4], [3,5,3], [4,3,5], [4,3,6], [5,3,5] and [5,3,6], GAP was clearly slower than `LowIndAlg`:

- For [3,4,4], `LowIndAlg` took 1.84 seconds while GAP took 2.93 seconds;
- For [3,5,3], `LowIndAlg` took 45.01 seconds while GAP took 4966.14 seconds (over 80 minutes);
- For [4,3,6], `LowIndAlg` took 2.35 seconds while GAP took 3.61 seconds;

- For case [5,3,5], `LowIndAlg` took 282.65 seconds (under 5 minutes) while GAP took over three days;
- For cases [4,3,5] and [5,3,6], `LowIndAlg` took 24742.83 and 21517.99 seconds (under 7 and 6 hours respectively), while GAP had taken over eleven days before we abandoned those two GAP jobs.

Bibliography

[1] C. Adams. The noncompact hyperbolic 3-manifold of minimum volume. *Proc. Amer. Math. Soc.*, 100:601–606, 1987.

[2] M.A. Armstrong. Calculating the fundamental group of an orbit space. *Proc. Amer. Math. Soc.*, 84:267–271, 1982.

[3] K. Asciak, M.D.E. Conder, S. Pavlikova, and J. Siran. Orientable and non-orientable regular maps with given exponent group. *J. Algebra*, 620:519–533, 2023.

[4] W.W. Boone. The word problem. *Ann. of Math.*, 70:207–265, 1959.

[5] W. Bosma, J. Cannon, and C. Playoust. The Magma algebra system. I. The user language. *J. Symbolic Comput.*, 24(3-4):235–265, 1997.

[6] M.R. Bridson, M.D.E. Conder, and A.W. Reid. Determining Fuchsian groups by their finite quotients. *Israel J. Math*, 214:1–41, 2016.

[7] B. Brink and R.B. Howlett. A finiteness property and an automatic structure for Coxeter groups. *Math. Ann.*, 296:179–190, 1993.

[8] C. Cao and G.R. Meyerhoff. The orientable cusped hyperbolic 3-manifolds of minimum volume. *Invent. Math.*, 146:451–478, 2001.

[9] M. Conder. The genus of compact Riemann surfaces with maximal automorphism group. *J. Algebra*, 108:204–247, 1987.

[10] M. Conder. *Group actions on graphs, maps and surfaces with maximum symmetry*, volume 1 of *Groups St Andrews 2001 in Oxford*, chapter 11, pages 63–91. Cambridge University Press, Cambridge, 2003.

[11] M. Conder and G. Liversidge. Small volume 3-manifolds constructible from string Coxeter groups of rank 4. *J. Algebra*, 635:775–789, 2023.

[12] M. Conder, G. Liversidge, and M. Vsemirnov. Generating pairs for $\mathrm{SL}(n, \mathbb{Z})$. 2023. In preparation.

[13] M.D.E. Conder, E. Robertson, and P. Williams. Presentations for 3-dimensional special linear groups over integer rings. *Proc. Amer. Math. Soc.*, 115:19–26, 1992.

[14] David Eppstein. Image: k_7 on torus. en.wikipedia.org/wiki/File:7x-torus.svg. Accessed:5/11/2023.

[15] B. Everitt. Coxeter groups and hyperbolic manifolds. *Math. Ann.*, 330:127–150, 2004.

[16] H. Felsch. Programmierung der Restklassenabzählung einer Gruppe nach Untergruppen. *Numer. Math*, 3:250–256, 1961.

[17] D. Firth. *An algorithm to find normal subgroups of a finitely presented group up the a given index.* University of Warwick, PhD thesis, 2005.

[18] W. Fulton. *Algebraic topology.* Grad. Texts in Math., No. 153. Springer-Verlag, New York, 1995.

[19] D. Gabai, R. Meyerhoff, and P. Milley. Minimum volume cusped hyperbolic three-manifolds. *J. Amer. Math. Soc.*, 22:1157–1215, 2009.

[20] H. Gieseking. *Analytische Untersuchungen über topologische Gruppen.* University of Münster, PhD thesis, 1912.

[21] G. Havas and C. Ramsay. ACE: Advanced coset enumerator. version 3. 1999. Available at staff.itee.uq.edu.au/havas.

[22] G. Havas and C. Ramsay. *Andrews-Curtis and Todd-Coxeter proof words*, volume 1 of *Groups St Andrews 2001*, page 232–237. Cambridge University Press, Cambridge, 2003.

[23] G. Havas and C. Ramsay. *On proofs in finitely presented groups*, volume 2 of *Groups St Andrews 2005*, pages 475–485. Cambridge University Press, Cambridge, 2007.

[24] D.L. Johnson. *Topics in the theory of Group Presentations.* Cambridge University Press, 1980.

[25] J. Leech. Coset enumeration on digital computers. *Proc. Cambridge Philos. Soc.*, 59:257–267, 1963.

[26] J. Leech. Generators for certain normal subgroups of (2,3,7). *Math. Proc. Cambridge Philos. Soc.*, 61:321–332, 1965.

[27] P. Lorimer. Four dodecahedral spaces. *Pacific J. Math.*, 156:329–335, 1992.

[28] P. Lorimer. Models for a finite universe. *Internat. J. Theoret. Phys.*, 41:1201–1274, 2002.

[29] A.M. Macbeath. On a curve of genus 7. *Proc. London Math Soc.*, 15:527–542, 1965.

[30] W.S. Massey. *A basic course in algebraic topology.* Grad. Texts in Math., No. 127. Springer-Verlag, New York, 1991.

[31] S.V. Matveev and A.T. Fomenko. Isoenergetic surfaces of hamiltonian systems, the enumeration of three-dimensional manifolds in order of growth of their complexity, and the calculation of the volumes of closed hyperbolic manifolds. *Russian Math. Surveys*, 43:3–24, 1988.

[32] J. Milnor. *Introduction to algebraic K-theory.* Annals of Mathematics Studies, No. 72. Princeton University Press, Princeton, NJ, 1971.

[33] J. Milnor. How to compute volume in hyperbolic space. *John Milnor Collected Papers*, Vol. 1, Geometry, Publish or Perish, 1994.

[34] G.D. Mostow. Quasi-conformal mappings in n-space and the rigidity of hyperbolic space forms. *Publications Mathématiques de l'IHÉS*, 34:53–104, 1968.

[35] P.S. Novikov. On the algorithmic unsolvability of the word problem in group theory. *Proc. Steklov Inst. Math.*, 44, 1955.

[36] G. Prasad. Strong rigidity of $\mathbb{Q}$-rank 1 lattices. *Invent. Math.*, 21:255–286, 1973.

[37] C. Ramsay. PACE: Parallel advanced coset enumerator. 2000. Available at staff.itee.uq.edu.au/havas.

[38] C. Ramsay. PEACE 1.000: Proof extraction after coset enumeration. 2000. Available at staff.itee.uq.edu.au/havas.

[39] J.G. Ratcliffe. *Foundations of Hyperbolic Manifolds*, volume 149 of *Graduate Texts in Math.* Springer, Berlin, 1994.

[40] C.C. Sims. *Computation with Finitely Presented Groups.* Cambridge University Press, 1994.

[41] M.C. Tamburini, J.S. Wilson, and N. Gavioli. On the (2,3)-generation of some classical groups. *J. Algebra*, 168:353–370, 1994.

[42] J.A. Todd and H.S.M. Coxeter. A practical method for enumerating cosets of a finite abstract group. *Proc. Edinb. Math. Soc.*, 5:26–34, 1936.

[43] S.M. Trott. A pair of generators for the unimodular group. *Canad. Math. Bull.*, 5:245–252, 1962.

[44] unknown. Image: Polar map on the sphere. alchetron.com/Regular-map-(graph-theory). Accessed:2/11/2023.

[45] Jarke van Wijk. Image: Klein map. Reproduced with permission.

[46] M. Vsemirnov. On (2,3)-generation of matrix groups over the ring of integers. *St. Petersburg Math. J.*, 19:883–910, 2008.

[47] M. Vsemirnov. On (2,3)-generation of matrix groups over the ring of integers, part 2. *St. Petersburg Math. J.*, 32:865–884, 2021.

[48] J. Weeks. *Hyperbolic structures on 3-manifolds.* Princeton University, PhD thesis, 1985.