Computing normalisers of highly intransitive groups

Mun See Chang

University of St Andrews

This thesis is submitted in partial fulfilment for the degree of
Doctor of Philosophy (PhD)
at the University of St Andrews

December 2020

Abstract

We investigate the normaliser problem, that is, given $G, H \leq S_n$, compute $N_G(H)$. The fastest known theoretical algorithm for this problem is simply exponential, but more efficient algorithms are known for some restriction of classes for G and H. In this thesis, we will focus on highly intransitive groups, which are groups with many orbits. We give new algorithms to compute $N_{S_n}(H)$ for highly intransitive groups $H \leq S_n$ and for some subclasses that perform substantially faster than previous implementations in the computer algebra system GAP.

Acknowledgements

General acknowledgements

My greatest gratitude goes to my two supervisors, Christopher Jefferson and Colva M. Roney-Dougal, who have tirelessly put up with my many shortcomings and patiently guide me in my development. Needless to say, the thesis would not exist without their expert guidance. I would also like to thank my examiners Alexander Hulpke and Peter Cameron, for their comments and discussions on the thesis. Additionally, I would like to thank Sergio Siccha for the many discussions on computing normalisers.

Furthermore, I am grateful to the CIRCA members, the Constraint Programming group, the GAP community and the St Andrews-Halle Permutation Algorithms group for the advice and discussions. I would also like to thank the administrative teams in the School of Computer Science and the School of Mathematics and Statistics for their help and organisation. I appreciate all those who have kept me sane these few years, with office talks, long lunches and the much-needed coffee and tea breaks. To those who had to listen to my rants, a massive thank you.

The last round of thanks is reserved for my partner's and my families, whom I love dearly, and who have supported and encouraged me throughout the years. This thesis is dedicated to Dun for empathising and reasoning with me on the countless times I wanted to quit.

Funding

This work was supported by the University of St Andrews (School of Computer Science and St Leonard's College Scholarship).

Research Data/Digital Outputs access statement

Research data underpinning this thesis are available at `https://doi.org/10.17630/710dfd8d-356b-4080-b2ad-c6791b7c21fe` [Cha21].

Underpinning Research Data or Digital Outputs

Candidate's declaration

I, Mun See Chang, understand that by declaring that I have original research data or digital outputs, I should make every effort in meeting the University's and research funders' requirements on the deposit and sharing of research data or research digital outputs.

Date 12/4/2021 Signature of candidate

Permission for publication of underpinning research data or digital outputs

We understand that for any original research data or digital outputs which are deposited, we are giving permission for them to be made available for use in accordance with the requirements of the University and research funders, for the time being in force.

We also understand that the title and the description will be published, and that the underpinning research data or digital outputs will be electronically accessible for use in accordance with the license specified at the point of deposit, unless exempt by award of an embargo as requested below.

The following is an agreed request by candidate and supervisor regarding the publication of underpinning research data or digital outputs:

No embargo on underpinning research data or digital outputs.

Date 12/4/2021 Signature of candidate

Date 13/4/2021 Signature of supervisor

Contents

Introduction

Groups are algebraic objects that arise from the symmetries of combinatorial structures. *Permutation groups*, which are groups consisting of permutations (bijections of a set), are the oldest and most well-studied types of groups. The area of *computational group theory* studies algorithms for groups. In this thesis, we will focus on permutation group algorithms.

Given permutation groups $G, H \leq S_n$, the problem of computing the normaliser $N_G(H) \coloneqq \{g \in G \mid g^{-1}Hg = H\}$ is called the *normaliser problem*. Normalisers are important for analysing group structures and have been used intensively in group constructions. Hence it is important to have algorithms with reasonable runtimes. However, there are currently no known polynomial time algorithms to solve the normaliser problem in general. The fastest known bound for this problem in general is simply exponential [Wie19]. Better bounds are known for various cases: for example, quasipolynomial if H is primitive [RDS20], and polynomial if G has restricted composition factors [LM02].

As a consequence of Babai's quasipolynomial solution to string isomorphism [Bab16], the intersection of permutation groups can be computed in quasipolynomial time. So, with a quasipolynomial cost, it suffices to compute $N_{S_n}(H)$, then $N_G(H) = N_{S_n}(H) \cap G$. More specifically, we shall focus on the following special case of the normaliser problem.

Problem 0.0.1 (Norm-Sym)**.** Given $H = \langle X \rangle \leq S_n$, compute $N_{S_n}(H)$.

We shall assume that a group $H \leq S_n$ is given by a generating set X. Since permutations in S_n are stored as a list of images of $\{1, 2, \dots, n\}$, we therefore have an input size of $|X|n$. In general we will measure complexity in terms of input size. It is well known that in time $O(|X|n^2 + n^5)$, we may replace X with a generating set of size at most n (see for example [Ser03]). Therefore, we may assume that the generating set X has size at most n and we shall measure complexity in terms of n.

In [Luk93], Luks shows that the graph isomorphism problem is polynomial time reducible to Norm-Sym, which is a special case of the normaliser problem. However, not much more is known about where Norm-Sym fits into the complexity hierarchy. In particular, we do not know its relation with the string isomorphism problem and the intersection problem. To better understand its worst case complexity, it is helpful to study the case where the problem seems to be the hardest. In this thesis, we will consider *highly intransitive* groups. More specifically, we compute $N_{S_n}(H)$ of a given group $H \leq S_n$ with many orbits.

For $H \leq S_n$, we say $H = H_1 \times H_2 \times \ldots \times H_r$ is a *disjoint direct product decomposition* of H if it is a direct product decomposition of H and the factors H_i have pairwise disjoint supports. If H has a non-trivial disjoint direct product decomposition $H = H_1 \times H_2 \times \ldots \times H_r$ where $r > 1$, then computing the normaliser $N_{S_n}(H)$ is reduced to computing the normaliser $N_{\mathrm{Sym}(Supp(H_i))}(H_i)$ for each $1 \leq i \leq r$ and solving the conjugacy problem for each distinct pair of the factors H_i and H_j. Disjoint direct product decompositions also have many applications beyond computing normalisers, in permutation group algorithms and beyond. The first result of this thesis is to show that the finest disjoint direct product decomposition of a group $H = \langle X \rangle \leq S_n$ can be computed in polynomial time. As we shall see in the chapter, the algorithm only manipulates certain strong generating sets, so is very efficient in practice, once the required strong generating set is obtained.

The fastest implemented algorithm for computing the normaliser $N_{S_n}(H)$ in general is a *backtrack search* algorithm. In practice, this algorithm is efficient in many classes of H. One objective of this thesis is to identify classes of H such that computing $N_{S_n}(H)$ is slow in the backtrack algorithm. Two permutation groups H_1 and H_2 are said to be *permutation isomorphic* if they are "the same up to the labelling of points". The normaliser $N_{S_n}(H)$ permutes the H-orbits on which the projections of H are permutation isomorphic. Therefore we consider the following class of groups.

Definition 0.0.2. Let $A \leq S_m$ be a transitive group. Let $\mathfrak{InP}(A)$ be a class consisting of all groups $H \leq S_n$ with k orbits $\Omega_1, \Omega_2, \ldots, \Omega_k$ and each projection $H|_{\Omega_i}$ is permutation isomorphic to A.

There are many practical algorithms to solve special cases of the normaliser problem $N_G(H)$. For example, Holt gave methods for the case H is regular [Hol91], Hulpke considered the case where H is elementary abelian and produced new methods using invariant sets [Hul08], and Miyamoto gave new methods using association schemes [Miy06]. For the NORM-SYM problem, certain structures of H can be used to prune the search tree. In general, these structures should be efficient to compute and preserved under conjugation.

A group $H \leq S_n$ with orbits $\Omega_1, \Omega_2, \ldots, \Omega_k$ can be regarded as a subdirect product of $H|_{\Omega_1} \times H|_{\Omega_2} \times \ldots \times H|_{\Omega_k}$. Goursat's lemma dictates that a subdirect product is uniquely identified by k normal subgroups of the $H|_{\Omega_i}$ and k homomorphisms [Sch94, BSZ15]. Using these structures and the aforementioned disjoint direct product decomposition, we formulate new pruning techniques to compute the normalisers $N_{S_n}(H)$ of intransitive groups H. In an effort to focus on the hardest case, we identify the two classes of H such that the current techniques do not yield much pruning.

In the first case, we have $H \in \mathfrak{InP}(C_p)$, where $C_p \leq S_p$ is the cyclic group of prime degree p. We believe the class $\mathfrak{InP}(C_p)$ of H is likely to be one of the hardest cases of the NORM-SYM problem, as the implementation in the computer algebra system GAP [GAP20] is very slow in solving the NORM-SYM problem for groups in $\mathfrak{InP}(C_p)$. We will show that the Norm-Sym problem for $H \in \mathfrak{InP}(C_p)$ is polynomial equivalent to computing the monomial automorphism group of a code over $\mathbb{F}_p$. Using methods for computing automorphisms of linear codes, we give a new faster algorithm for solving the Norm-Sym problem for $H \in \mathfrak{InP}(C_p)$, and demonstrate that our new algorithm performs far better than the one currently implemented in GAP. The fastest implemented algorithm to compute $N_{S_n}(H)$ has a run time of $2^{O(n \log n)}$. Using methods based on coding theory, we shall bound the complexity of the Norm-Sym problem for $H \in \mathfrak{InP}(C_p)$ by $\min\left(2^{O(\frac{n}{p}\log\frac{n}{p})}, 2^{O(\frac{n}{2p}\log n)}\right)$, and give an algorithm that performs efficiently in practice.

The second case that was thought to be hard is where H is in the class $\mathfrak{InP}(T)$, where $T \leq S_m$ is a transitive non-abelian simple group. As a consequence of the Classification of Finite Simple Groups (see [Gor82, Gor83, ALSS11]), many properties of a non-abelian simple group are known. Using the fact that $|Out(T)|$ is polynomially bounded, Luks and Miyazaki showed that the Norm-Sym problem for non-abelian simple groups H can be computed in polynomial time [LM02]. We shall show that the Norm-Sym problem for $H \in \mathfrak{InP}(T)$ can also be computed in polynomial time.

With a little extra work, our algorithms are generalised to compute the normalisers $N_{S_n}(H)$ for other classes of $H \leq S_n$. We shall show how we can adapt our algorithm to the case where $H \in \mathfrak{InP}(D_{2p})$, the class of groups with intransitive projections permutation isomorphic to dihedral groups of order $2p$, where p is an odd prime, and also to the case where $H \in \mathfrak{InP}(S_m)$, that is the class of groups with intransitive projections permutation isomorphic to the symmetric group of degree $5 \leq m \neq 6$.

The thesis is structured in the following way. We start with some background materials on permutation groups in Chapter 1. Section 1.4 is of significant importance, as it gives certain structures of intransitive groups which we will use throughout the thesis. In Chapter 2, we will present a polynomial time library and describe some problems in the Luks hierarchy. We will also describe backtrack search in groups, and present some literature review on computing normalisers of permutation groups, in terms of theoretical complexity as well as some practical approaches. In Chapter 3, we will show that the finest disjoint direct product decomposition of a given permutation group can be computed in polynomial time. We return our focus to computing normalisers in Chapter 4, where we give several pruning methods for computing normalisers of highly intransitive groups using some properties preserved under the conjugations. In Chapter 5, we will consider the groups $H \leq S_n$ in class $\mathfrak{InP}(C_p)$. As mentioned earlier, we will show that we can compute the normaliser $N_{S_n}(H)$ for $H \in \mathfrak{InP}(C_p)$ by computing certain group of automorphisms of a linear code. Using this alternate viewpoint, we give a faster algorithm to compute the normalisers $N_{S_n}(H)$ of groups $H \leq S_n$ in this class. We will also see how we use similar methods to improve the normaliser computation for groups in $\mathfrak{InP}(D_{2p})$ for p an odd prime. In Chapter 6, we consider the groups in class $\mathfrak{InP}(T)$, where T is non-abelian simple. We will show that we can compute the normalisers $N_{S_n}(H)$ for the groups $H \leq S_n$ in this class in polynomial time, and give an algorithm that also performs competitively in practice. Lastly, we see how these methods generalise to computing normalisers of groups in class $\mathfrak{InP}(S_m)$ for $5 \leq m \neq 6$.

Chapter 1

Permutation Groups

In this chapter we will give some elementary concepts and results we will be using in the later chapters. Most of the results in this chapter can be found in Chapters 1–4 of [DM96].

Definition 1.0.1. Let Ω be a set. A *permutation* is a bijection from Ω to itself.
The *symmetric group* on Ω is the group consisting of all permutations of Ω, denoted by $\mathrm{Sym}(\Omega)$. If $\Omega = \{1, 2, \ldots, n\}$, we write $\mathrm{Sym}(\Omega)$ as S_n.
A *permutation group* is a subgroup of a symmetric group.

Throughout the thesis, we write permutations as products of disjoint cycles. All groups considered in the thesis are finite and all permutation groups are acting on finite sets.

1.1 Group actions

Definition 1.1.1. Let G be a group and let Ω be a non-empty set. Let $\xi : G \times \Omega \to \Omega$ be a mapping, where we denote the image $\xi(x, \alpha)$ by α^x. Then the mapping defines an *action* of G on Ω if

1. $\alpha^1 = \alpha$ for all $\alpha \in \Omega$, where 1 is the identity element of G; and
2. $\alpha^{(xy)} = (\alpha^x)^y$ for all $\alpha \in \Omega$ and $x, y \in G$.

If such a ξ exists then we say that G *acts on* Ω.

Group actions are closely related to permutation representations.

Definition 1.1.2. A homomorphism $\rho : G \to \mathrm{Sym}(\Omega)$ is called a *permutation representation* of G on Ω.

An action of G on a set Ω gives a permutation representation of G. Conversely every permutation representation corresponds to some action.

Proposition 1.1.3. *1. Let G be a group acting on a set Ω. For each $g \in G$, let $\bar{g} : \Omega \to \Omega$ be defined by $\alpha \mapsto \alpha^g$. Let $\rho : G \to \mathrm{Sym}(\Omega)$ be defined by $\rho(g) = \bar{g}$. Then ρ is a permutation representation of G.*

2. Let $\rho : G \to \mathrm{Sym}(\Omega)$ be a permutation representation. Define a mapping $\xi : G \times \Omega \to \Omega$ by $(g, \alpha) \mapsto \alpha^{\rho(g)}$. Then ξ defines an action of G on Ω,

Proof. Part 1: For $g \in G$, the map $\bar{g}$ is a bijection since $\bar{g}^{-1}$ maps α^g to $\alpha^{gg^{-1}} = \alpha$. So $\bar{g} \in \mathrm{Sym}(\Omega)$. The map ρ is a homomorphism as

$$\alpha^{\rho(gh)} = \alpha^{\overline{(gh)}} = \alpha^{gh} = (\alpha^{\rho(g)})^{\rho(h)}, \qquad \text{for all } \alpha \in \Omega \text{ and } g, h \in G.$$

Therefore $\rho(gh) = \rho(g)\rho(h)$.
Part 2: Let $\xi : G \times \Omega \to \Omega$ be defined by $(g, \alpha) \mapsto \alpha^{\rho(g)}$. As homomorphisms send the identity to the identity, $\xi(1, \alpha) = \alpha^{\rho(1)} = \alpha$ and

$$\alpha^{gh} = \xi(gh, \alpha) = \alpha^{\rho(gh)} = \alpha^{\rho(g)\rho(h)} = (\xi(g, \alpha))^{\rho(h)} = (\alpha^g)^{\rho(h)} = \xi(h, \alpha^g) = (\alpha^g)^h.$$

□

The *degree* of the action of G on Ω is $|\Omega|$, and the *kernel* of the action is the kernel $\{g \in G \mid \varphi(g) = 1\}$ of the corresponding permutation representation φ. We say an action is *faithful* if its kernel is trivial, or equivalently, if the corresponding permutation representation φ is injective.

We use permutation equivalence to compare group actions and the corresponding permutation representations.

Definition 1.1.4. Two permutation representations $\rho : G \to \mathrm{Sym}(\Omega)$ and $\sigma : G \to \mathrm{Sym}(\Gamma)$ of G are *equivalent* if there exists a bijection $\lambda : \Omega \to \Gamma$ such that

$$\lambda(\alpha^{\rho(g)}) = \lambda(\alpha)^{\sigma(g)} \quad \text{for all } \alpha \in \Omega \text{ and } g \in G, \tag{1.1}$$

and we say λ is a bijection *witnessing* the equivalence. Two actions are said to be *equivalent* if the corresponding permutation representations are equivalent.

Equivalent permutation representations should not be confused with permutation isomorphism, which captures the permutation groups that are "the same up to relabelling of points".

Definition 1.1.5. Let $G \leq \mathrm{Sym}(\Omega)$ and $H \leq \mathrm{Sym}(\Delta)$. Then G and H are said to be *permutation isomorphic* if there exist a bijection $\phi : \Omega \to \Delta$ and a group isomorphism $\psi : G \to H$ such that

$$\phi(\alpha^g) = \phi(\alpha)^{\psi(g)} \quad \text{for all } \alpha \in \Omega \text{ and } g \in G, \tag{1.2}$$

and we say ϕ and ψ is a bijection and an isomorphism *witnessing* the permutation isomorphism.

Example 1.1.6. Let $G \cong C_2 \times C_2$ and with elements $(1,1),(1,a),(a,1),(a,a)$ where $aa = 1$. Let $\rho : G \to S_4$ defined by $(1,a) \mapsto (1,2)$ and $(a,1) \mapsto (3,4)$ be a permutation representation of G.

1. Let $\sigma : G \to S_4$ defined by $(1,a) \mapsto (3,4)$ and $(a,1) \mapsto (1,2)$ be a permutation representation of G. Then ρ and σ are equivalent permutation representations of G as the bijection $\lambda : \{1,2,3,4\} \to \{1,2,3,4\}$ defined by $1 \mapsto 3$, $2 \mapsto 4$, $3 \mapsto 1$ and $4 \mapsto 2$ satisfies Equation (1.1).

2. Now let $\sigma : G \to S_4$ defined by $(1,a) \mapsto (1,2)$ and $(a,1) \mapsto (1,2)(3,4)$ be another permutation representation of G. There are no such bijection λ as it would require $\lambda(3) = \lambda(3^{(1,2)}) = \lambda(3^{\rho((a,1))}) = \lambda(3)^{\sigma((a,1))} = \lambda(3)^{(1,2)(3,4)}$. So ρ and σ are not equivalent permutation representations of G.

However, in both cases, $\rho(G)$ and $\sigma(G)$ are permutation isomorphic (they are indeed the same group).

Observe that two permutation groups on the same set Ω are permutation isomorphic if and only if they are conjugate subgroups of $\mathrm{Sym}(\Omega)$.

Proposition 1.1.7. *Let $H, G \leq \mathrm{Sym}(\Omega)$. Then H and G are permutation isomorphic if and only if H and G are conjugate in $\mathrm{Sym}(\Omega)$.*
Furthermore, if $\phi : \Omega \to \Omega$ is a bijection witnessing the permutation isomorphism from H to G, then $H^\phi = G$.

Proof. $\Rightarrow$: Let $\phi : \Omega \to \Omega$ and $\psi : H \to G$ be a bijection and an isomorphism that together witness the permutation isomorphism between H and G, as in Definition 1.1.5. We shall show that $H^\phi = G$.
Let $h \in H$ and let $\beta \in \Omega$. Then there exists $\alpha \in \Omega$ such that $\phi(\alpha) = \beta$. Then

$$\beta^{(h^\phi)} = \alpha^{\phi\phi^{-1}h\phi} = \alpha^{h\phi} = \phi(\alpha^h) = \phi(\alpha)^{\psi(h)} = \beta^{\psi(h)}.$$

Hence $h^\phi = \psi(h)$ and so $H^\phi \leq \psi(H) = G$.
$\Leftarrow$: Let $\sigma \in \mathrm{Sym}(\Omega)$ such that $H^\sigma = G$. Let $\phi = \sigma$ and let $\psi : H \to G$ be the isomorphism defined by $\psi(h) = h^\sigma$ for all $h \in H$. Then for all $h \in H$ and $\alpha \in \Omega$, we have

$$\phi(\alpha^h) = \alpha^{h\sigma} = (\alpha^\sigma)^{\sigma^{-1}h\sigma} = \phi(\alpha)^{h^\sigma} = \phi(\alpha)^{\psi(h)}.$$

□

For a group G acting on a set Ω, each point $\alpha \in \Omega$ is moved by elements of G to other points of Ω. These images of α form the orbit of α under G.

Definition 1.1.8. Let G be a group acting on Ω and let $\alpha \in \Omega$. The *orbit of* α under G is the set $\alpha^G = \{\alpha^g \mid g \in G\}$ of images of α under G. We call the set of all orbits of $\alpha \in \Omega$ under the action of G the *orbits of* G.

Two orbits of G are either disjoint or equal [DM96, Theorem 1.4A], so the orbits of G form a partition of Ω. We say that a group G is *transitive* if G has one orbit Ω and *intransitive* otherwise.

We may sometimes want to exclude points that are fixed by an element or a subset of G. Hence we introduce support and fixed points.

Definition 1.1.9. Let G be a group acting on Ω, and let $g \in G$. The *support* of g is the set $Supp(g) \coloneqq \{\alpha \in \Omega \mid \alpha^g \neq \alpha\}$ and the *fixed points* of g form the set $Fix(g) \coloneqq \{\alpha \in \Omega \mid \alpha^g = \alpha\}$.
Let S be a subset of G. The *support* of S is the set $Supp(S) \coloneqq \cup_{s \in S} Supp(s)$ and the *fixed points* of S form the set $Fix(S) \coloneqq \cap_{s \in S} Fix(s)$.

The dual concept to an orbit is the point stabiliser.

Definition 1.1.10. For a group G acting on a set Ω, the *stabiliser* of $\alpha \in \Omega$ in G is the set $G_\alpha = \{g \in G \mid \alpha^g = \alpha\}$.

Proposition 1.1.11 ([DM96, Theorem 1.4A])**.** *Let G be a group acting on Ω and let $\alpha, \beta \in \Omega$ and $g \in G$. The following hold.*

1. *If $\alpha^g = \beta$, then $G_\beta = G_\alpha^g$.*
2. *(Orbit-stabilizer property)* $|\alpha^G| = |G : G_\alpha|$.

The action of G on Ω is *regular* if G is transitive and $G_\alpha = 1$ for all $\alpha \in \Omega$. By the orbit-stabiliser property, if G is finite, then the action is regular if and only if $|G| = |\Omega|$.

In terms of stabilisers of a set, we differentiate between the pointwise and the setwise stabilisers.

Definition 1.1.12. Let G be a group acting on Ω and let $\Delta \subseteq \Omega$. The *pointwise stabiliser* of Δ in G is the subgroup $G_{(\Delta)} = \{g \in G \mid \alpha^g = \alpha \text{ for all } \alpha \in \Delta\}$. The *setwise stabiliser* of Δ in G is the subgroup $G_{\{\Delta\}} = \{g \in G \mid \alpha^g \in \Delta \text{ for all } \alpha \in \Delta\}$.

A group G acting transitively on Ω may preserve an (unordered) partition of Ω, which will give blocks of imprimitivity.

Definition 1.1.13. Let G be a group acting transitively on Ω. Then $\Delta \subseteq \Omega$ is a *block* for G if for every $g \in G$, either $\Delta^g = \Delta$ or $\Delta \cap \Delta^g = \emptyset$.
If Δ is a block then the set $\{\Delta^g \mid g \in G\}$ is a partition of Ω, called a *system of blocks* for G.

The set of singleton subsets of Ω forms a system of blocks of imprimitivity for G. Similarly, Ω is a block for G. We call these blocks *trivial blocks*. A transitive group is said to be *imprimitive* if it has non-trivial blocks, and is said to be *primitive* otherwise.

Finally observe that the orbits of a normal subgroup give a system of blocks.

Proposition 1.1.14 ([DM96, Theorem 1.6A]). *Let G be a group acting transitively on Ω and let $N \trianglelefteq G$ be a normal subgroup of G. Then the orbits of N form a system of blocks for G.*

1.2 Constructing groups from groups

We begin with the product of subgroups.

Definition 1.2.1. Let G be a group and let H and K be subgroups of G. Then the *product of subgroups* H and K is the set $HK := \{hk \mid h \in H \text{ and } k \in K\}$.

The set HK need not be a subgroup of G, but it is if either H or K are normal in G. If $G = HK$, we say that K is a *supplement* of H in G. If $G = HK$ and $H \cap K = 1$, we say that K is a *complement* of H in G. If $G = HK$, $H \cap K = 1$ and both K and H are normal in G, then G is the (internal) direct product of H and K.

Definition 1.2.2. Let G and H be groups. Then the *direct product* $G \times H$ of G and H is the group with elements $\{(g, h) \mid g \in G,\, h \in H\}$ together with the component-wise binary operation.

The direct product defined above is an external one. Note that for each external direct product there is a corresponding internal direct product.

Remark 1.2.3. Let G and H be groups and let $K := G \times H$. Then $\overline{G} = G \times 1$ and $\overline{H} = 1 \times H$ are normal subgroups of K with trivial intersection and any element k of K can be written uniquely as a product $k = \overline{g}\overline{h}$, where $\overline{g} \in \overline{G}$ and $\overline{h} \in \overline{H}$. So K is an *internal direct product* of $\overline{G}$ and $\overline{H}$.

Associated to direct products are the projection maps.

Notation 1.2.4. Let $G_1, G_2, \ldots, G_k$ be groups and let $G = G_1 \times G_2 \times \ldots \times G_k$. We denote by π_i the projection of G onto G_i. So for $g = (g_1, g_2, \ldots, g_k) \in G$, we have $\pi_i(g) = g_i$.
For $I \subseteq \{1, 2, \ldots, k\}$, we denote by Π_I the projection onto I, so $\Pi_I(g) = (g_{i_1}, g_{i_2}, \ldots, g_{i_r})$, where $i_1, i_2, \ldots, i_r$ are the elements of I in the natural ordering.

The projection maps are homomorphisms. The surjectivity of the projections π_i gives the definition of a subdirect product.

Definition 1.2.5. Let $G_1, G_2, \ldots, G_k$ be groups and let $G = G_1 \times G_2 \times \ldots \times G_k$. A subgroup H of G is a *subdirect product* of G if the projections $\pi_i : H \to G_i$ are surjective for all $1 \leq i \leq k$.

Therefore we may regard an intransitive group as a subdirect product of the direct products of the projections on its orbits.

Proposition 1.2.6. *Let $H \leq S_n$ and let $\Omega_1, \Omega_2, \ldots, \Omega_k$ be the orbits of H. Then H is a subdirect product of $H|_{\Omega_1} \times H|_{\Omega_2} \times \ldots \times H|_{\Omega_k}$, where we identify the direct product as a subgroup of S_n.*

We will further discuss the structure of intransitive groups in Section 1.4.

A generalisation of direct product is the semidirect product.

Definition 1.2.7. Let H and K be groups and let H act on K. The (external) *semidirect product* of H by K is the group $K \rtimes H := \{(k, h) \mid k \in K \text{ and } h \in H\}$ where the products are defined by

$$(k_1, h_1)(k_2, h_2) = (k_1 k_2^{h_1^{-1}}, h_1 h_2)$$

for all $k_1, k_2 \in K$ and $h_1, h_2 \in H$.

Example 1.2.8. Let $H \leq G$ and $K \triangleleft G$ such that $G = HK$ and $K \cap H = 1$. Then any element g of G can be written as kh, for some $k \in K$ and $h \in H$. Since $K = K^g = K^{kh}$, we have $K^h = K$. So H acts on K by conjugation. Let $g_1 = k_1 h_1$ and $g_2 = k_2 h_2$ be elements of G, then $g_1 g_2 = k_1 h_1 k_2 h_2 = k_1 h_1 k_2 h_1^{-1} h_1 h_2 = k_1 k_2^{h_1^{-1}} h_1 h_2$. Therefore $G \cong K \rtimes H$.

Finally, a wreath product is a special case of a semidirect product.

Definition 1.2.9. Let K, H be groups and let H act on a non-empty finite set Γ. Let $L \cong K^{|\Gamma|}$ be the direct product $K_1 \times K_2 \times \ldots \times K_{|\Gamma|}$ of $|\Gamma|$ copies of K. The *wreath product* $K \wr_\Gamma H$ of K by H is the semidirect product $L \rtimes H$, where the action of H on $K^{|\Gamma|}$ is defined by $k_i^h = k_{i^{h^{-1}}}$ for all $h \in H$ and $k_i \in K_i$.

The subgroup L of $K \wr_\Gamma H$ is called the *base group* of the wreath product. We denote the case where H is given as a permutation group in $\mathrm{Sym}(\Gamma)$ simply as $K \wr H$ instead of $K \wr_\Gamma H$, which we call the *standard wreath product.*

The wreath product is strongly connected to imprimitive groups. For groups K and H acting on Δ and Γ respectively, the wreath product $K \wr H$ acts imprimitively on $\Delta \times \Gamma$. Conversely, any imprimitive group can be embedded into a wreath product.

Proposition 1.2.10 ([Cam99, Theorem 1.8])**.** *Let $G \leq \mathrm{Sym}(\Omega)$ be imprimitive with blocks $\Delta_1, \Delta_2, \ldots, \Delta_k$. Let $K \leq S_k$ be the group induced by the action of G on the Δ_i. Then G can be embedded into the wreath product $G_{\{\Delta_1\}}|_{\Delta_1} \wr K$.*

Note that $K \wr_\Gamma H$ also acts (primitively) on the cartesian product of $|\Gamma|$ copies of Δ. This action is called the *product action* of the wreath product.

1.3 Normal subgroups and normalisers

In this section we give some elementary results relating to normal subgroups and normalisers.

Definition 1.3.1. Let G be a group. A *minimal normal subgroup* of G is a non-trivial normal subgroup N of G such that there are no non-trivial normal subgroups K of G such that $K \subsetneq N$.

Lemma 1.3.2 ([DM96, Theorem 4.3A])**.** *Let G be a finite group. Let K be a minimal normal subgroup of G and let $L \trianglelefteq G$ with $K \neq L$. Then either $K \leq L$ or $\langle K, L \rangle = K \times L$. Hence if L is also a minimal normal subgroup of G, then $\langle K, L \rangle = K \times L$.*

The socle of a group plays an integral role in understanding primitive groups.

Definition 1.3.3. Let G be a group. The *socle* of G is the group $soc(G)$ generated by all minimal normal subgroups of G.

Proposition 1.3.4 ([DM96, Theorem 4.3A])**.** *Let G be a non-trivial finite group. Then*

1. *There exist minimal normal subgroups $N_1, N_2, \ldots, N_k$ of G such that $soc(G) = N_1 \times N_2 \times \ldots \times N_k$.*

2. *If N is a minimal normal subgroup of G, then N is a direct product of simple normal subgroups of N which are conjugate in G.*

We now turn our attention to normalisers of groups. For $G, H \leq S_n$, we denote by $N_G(H)$ the normaliser of H in G. We prove the following elementary but useful lemma.

Lemma 1.3.5 ([Ser03, Lemma 6.1.7])**.** *Let $H \leq \mathrm{Sym}(\Omega)$. Then $N_{\mathrm{Sym}(\Omega)}(H)$ permutes the H-orbits.*

Proof. Let $g \in N_{\mathrm{Sym}(\Omega)}(H)$. Let $\alpha, \beta \in \Omega$ be points in the same H-orbit. Then there exists $h \in H$ mapping α to β. So $h' := h^g$ is an element of H and $(\alpha^g)^{h'} = \alpha^{gg^{-1}hg} = \beta^g$. Therefore α^g and β^g are in the same H-orbit. Similarly, g maps points in different orbits to points in different orbits. $\square$

The normaliser of H is closely related to the automorphism group of H.

Definition 1.3.6. For a group G, the *automorphism group* $\mathrm{Aut}(G)$ of G is the set of all isomorphisms $\alpha : G \to G$ under composition of maps.

An automorphism $\alpha \in \mathrm{Aut}(G)$ of G is an *inner automorphism* if it is induced by the conjugation by an element $h \in G$. That is, there exists $h \in G$ such that $g^\alpha = g^h$ for all $g \in G$. Otherwise α is said to be an *outer automorphism* of G. The *inner automorphism group* $Inn(G)$ of G is the group consisting of all inner automorphisms of G, and is normal in $\mathrm{Aut}(G)$. The *outer automorphism group* of G is the quotient group $Out(G) = \mathrm{Aut}(G)/Inn(G)$.

We prove the following well-known theorem. Note that since $g^\nu \in G$ for all $g \in G$ and $\nu \in N_{S_n}(G)$, the element ν induces an automorphism of G.

Theorem 1.3.7. *Let $G \leq S_n$. Let $\phi : N_{S_n}(G) \to \mathrm{Aut}(G)$ be such that $\phi(\nu) = \overline{\nu}$, where $\overline{\nu}$ denotes the element of $\mathrm{Aut}(G)$ induced by conjugation by ν. Then ϕ is a homomorphism with $\mathrm{Ker}(\phi) = C_{S_n}(G)$. Hence $N_{S_n}(G)/C_{S_n}(G)$ is isomorphic to a subgroup of $\mathrm{Aut}(G)$.*

Proof. The map ϕ is a homomorphism as $g^{\phi(\nu_1\nu_2)} = g^{(\nu_1\nu_2)} = (g^{\phi(\nu_1)})^{\nu_2} = g^{\phi(\nu_1)\phi(\nu_2)}$ for all $\nu_1, \nu_2 \in N_{S_n}(G)$ and $g \in G$. To show $\mathrm{Ker}(\phi) = C_{S_n}(G)$, observe that $\nu \in N_{S_n}(G)$ is in $\mathrm{Ker}(\phi)$ if and only if $\phi(\nu) = 1$. That is, for all $g \in G$, we have $g^\nu = g^{\phi(\nu)} = g$, which is the condition for ν to be an element of $C_{S_n}(G)$. □

Definition 1.3.8. Let G be a group. A *characteristic subgroup* H of G is a subgroup $H \leq G$ where all automorphisms $\alpha \in \mathrm{Aut}(G)$ of G map H back to itself. That is, $H^\alpha = H$ for all $\alpha \in \mathrm{Aut}(G)$.

If $H \leq G$ is a characteristic subgroup, each element ν of $N_{S_n}(G)$ induces an automorphism $\overline{\nu}$ of G, and so $H^{\overline{\nu}} = H^\nu = H$. Therefore the normaliser $N_{S_n}(G)$ is contained in $N_{S_n}(H)$. Note that as automorphisms of G permute the minimal normal subgroups of G, the socle $soc(G)$ is a characteristic subgroup of G.

Lastly, we show how certain automorphisms of G can be used to generate $N_{S_n}(G)$.

Lemma 1.3.9. *Let $G \leq S_n$. Let R be a transversal of $Inn(G)$ in $\mathrm{Aut}(G)$. For each $\omega \in R$, if there exists $x \in S_n$ such that $h^\omega = h^x$ for all $h \in G$, let x_ω be one such element, and let X be the set of all such elements (at most one x_ω for each ω). Then $N_{S_n}(G) = \langle C_{S_n}(G), G, X\rangle$.*

Proof. $\geq$: Certainly $C_{S_n}(G)$ and G are contained in $N_{S_n}(G)$. Let $x_\omega \in X$ and $h \in G$. Then $h^{x_\omega} = h^\omega \in G$. So $x_\omega \in N_{S_n}(G)$.
$\leq$: Let $\phi : N_{S_n}(G) \to \mathrm{Aut}(G)$ map each $\nu \in N_{S_n}(G)$ to the element $\overline{\nu}$ of $\mathrm{Aut}(G)$ induced by conjugation by ν. By Theorem 1.3.7, ϕ is a homomorphism with kernel $C_{S_n}(G)$.
Let $\nu \in N_{S_n}(G)$. Since $\phi(\nu) \in \mathrm{Aut}(G)$, there exists $\iota \in Inn(G)$ and $\omega \in R$ such that $\phi(\nu) = \iota\omega$. We denote by $\phi^{-1}(R)$ the set $\{\phi^{-1}(r) \mid r \in R\}$, where $\phi^{-1}(r)$ is an element η of $N_{S_n}(G)$ such that $\phi(\eta) = r$. Since $\mathrm{Ker}(\phi) = C_{S_n}(G)$, we have

$$\nu \in \langle \phi^{-1}(Inn(G)), \phi^{-1}(R), C_{S_n}(G)\rangle.$$

By the definition of ϕ, we have $\phi(G) = Inn(G)$, so $\phi^{-1}(Inn(G)) = \langle G, C_{S_n}(G)\rangle$. Similarly, $\phi(X) = R$, so $\phi^{-1}(R) = \langle X, C_{S_n}(G)\rangle$. Hence, $\nu \in \langle C_{S_n}(G), G, X\rangle$. □

1.4 Structure of intransitive groups

This thesis will focus on computing the normalisers of intransitive groups. Recall from Proposition 1.2.6 that an intransitive group can be regarded as a subdirect product of its transitive constituents. Theorem 1.4.1 is commonly known as Goursat's lemma. It describes the subgroups of a direct product and appears in the literature in various places, including, for example, [Sch94, PS18].

Theorem 1.4.1 ([Gou89]). *Let G_1, G_2 be groups. Let H be a subdirect product of $G_1 \times G_2$. Let $\pi_1 : H \to G_1$ and $\pi_2 : H \to G_2$ be the projection maps of H onto G_1 and G_2 respectively. Let $N_1 := \pi_1(\mathrm{Ker}(\pi_2))$ and $N_2 := \pi_2(\mathrm{Ker}(\pi_1))$. Then the following hold.*

1. *$N_1 \trianglelefteq G_1$ and $N_2 \trianglelefteq G_2$.*
2. *G_1/N_1 is isomorphic to G_2/N_2, with isomorphism θ given by $N_1h_1 \mapsto N_2h_2$ where $(h_1, h_2) \in H$.*

Let R_1 and R_2 be transversals of N_1 in G_1 and N_2 in G_2 respectively. Let $\hat{\theta} : R_1 \to R_2$ be a map induced by θ, where $\hat{\theta}(r_1) = r_2$ if $\theta(N_1r_1) = N_2r_2$. Then letting $\mathcal{G} = \{(r, \hat{\theta}(r)) \mid r \in R_1\}$, we have $H = \langle \mathcal{G}, N_1 \times 1, 1 \times N_2 \rangle$.

For subdirect products of the direct product of more than two groups, we use an asymmetrical version of Theorem 1.4.1.

Proposition 1.4.2 ([BSZ15, Theorem 2.3]). *Let G_1, G_2 be groups. Let H be a subdirect product of $G_1 \times G_2$. Let $\pi_1 : H \to G_1$ and $\pi_2 : H \to G_2$ be the projection maps of H onto G_1 and G_2 respectively. Then the following hold.*

1. *Let $N_2 := \pi_2(\mathrm{Ker}(\pi_1))$. Then $N_2 \trianglelefteq G_2$.*
2. *Let $\theta : G_1 \to G_2/N_2$ be defined by $h_1 \mapsto N_2h_2$ for all $(h_1, h_2) \in H$. Then θ is a surjective homomorphism.*
3. *Let R_2 be a transversal of N_2 in G_2. Then by letting $\mathcal{G} = \{(g_1, r_2) \mid g_1 \in G_1, r_2 \in R_2$ such that $\theta(g_1) = N_2r_2\}$, we have $H = \langle \mathcal{G}, 1 \times N_2 \rangle$.*

We shall denote the set $\{1, 2, \ldots, i\}$ by $\bar{i}$. Let $G_1, G_2, \ldots, G_k$ be groups. Now, consider a subdirect product H of $G_1 \times G_2 \times \ldots \times G_k$. For a subset $I = \{i_1, i_2, \ldots, i_r\} \subseteq \bar{k}$, we use Π_I to denote the projection map of H onto $G_{i_1} \times G_{i_2} \times \ldots \times G_{i_r}$, as in Notation 1.2.4.

By iteratively considering $\Pi_{\overline{i+1}}(H)$ as a subdirect product of $\Pi_{\bar{i}}(H) \times G_{i+i}$, we get the following result. Since we will be using this result repeatedly, we present the proof here. Theorem 1.4.3 will be referenced in many places throughout the thesis.

Theorem 1.4.3 ([BSZ15, Theorem 3.2]). *Let $G_1, G_2, \ldots, G_k$ be groups. Let H be a subdirect product of $G_1 \times G_2 \times \ldots \times G_k$. Then for $1 \leq i \leq k-1$, the following hold.*

1. *Let $N_{i+1} := \pi_{i+1}(\mathrm{Ker}(\Pi_{\bar{i}})) = \{\pi_{i+1}(h) \in G_{i+1} \mid h \in H$ and $\Pi_{\bar{i}}(h) = 1\}$. Then $N_{i+1} \trianglelefteq G_{i+1}$.*
2. *Let $\theta_i : \Pi_{\bar{i}}(H) \to G_{i+1}/N_{i+1}$ be defined by $\Pi_{\bar{i}}(h) \mapsto N_{i+1}\pi_{i+1}(h)$. Then θ_i is a surjective homomorphism.*

3. Let R_{i+1} be a transversal of N_{i+1} in G_{i+1} and let

$$\begin{array}{rcl}\varphi_i : \Pi_{\bar{i}}(H) & \to & \Pi_{\bar{i}}(H) \times R_{i+1} \\ \Pi_{\bar{i}}(h) & \mapsto & (\Pi_{\bar{i}}(h), r) \quad \text{if } \theta_i(\Pi_{\bar{i}}(h)) = N_{i+1}r, \text{ with } r \in R_{i+1}.\end{array}$$

Then $\Pi_{\overline{i+1}}(H) = \langle \varphi_i(\Pi_{\bar{i}}(H)), \underbrace{1 \times \ldots \times 1}_{i \text{ times}} \times N_{i+1} \rangle$.

Proof. Part 1: Let $n \in N_{i+1}$. Then there exists $h_1 \in H$ such that $\Pi_{\bar{i}}(h_1) = 1$ and $n = \pi_{i+1}(h_1)$. Let $g \in G_{i+1}$. Since π_{i+1} is surjective, there exists $h_2 \in H$ such that $\pi_{i+1}(h_2) = g$. Then $\Pi_{\bar{i}}(h_1^{h_2}) = 1$, so $n^g = \pi_{i+1}(h_1^{h_2}) \in N_{i+1}$.

Part 2: First we show that θ_i is well-defined. Let h_1 and h_2 be elements of H such that $\Pi_{\bar{i}}(h_1) = \Pi_{\bar{i}}(h_2)$. Then $\Pi_{\bar{i}}(h_1h_2^{-1}) = 1$, and so $\pi_{i+1}(h_1h_2^{-1}) \in N_{i+1}$. Since π_{i+1} is a homomorphism, it follows that $\theta_i(\Pi_{\bar{i}}(h_1)) = N_{i+1}\pi_{i+1}(h_1) = N_{i+1}\pi_{i+1}(h_2) = \theta_i(\Pi_{\bar{i}}(h_2))$.
To show that θ_i is a homomorphism, let $h_1, h_2 \in H$. Then

$$\begin{aligned}\theta_i(\Pi_{\bar{i}}(h_1)\Pi_{\bar{i}}(h_2)) = \theta_i(\Pi_{\bar{i}}(h_1h_2)) &= N_{i+1}\pi_{i+1}(h_1h_2) \\ &= (N_{i+1}\pi_{i+1}(h_1))(N_{i+1}\pi_{i+1}(h_2)) \\ &= \theta_i(\Pi_{\bar{i}}(h_1))\theta_i(\Pi_{\bar{i}}(h_2)).\end{aligned}$$

To show that θ_i is surjective, observe that since H is a subdirect product, for each coset $N_{i+1}g$, there exists $h \in H$ such that $\pi_{i+1}(h) \in N_{i+1}g$. Therefore the image $\theta_i(\Pi_{\bar{i}}(h))$ is $N_{i+1}\pi_{i+1}(h) = N_{i+1}g$.

Part 3: $\geq$: Let $n \in N_{i+1}$, then there exists $h \in H$ such that $\Pi_{\bar{i}}(h) = 1$ and $\pi_{i+1}(h) = n$. So $(1, \ldots, 1, n) \in \Pi_{\overline{i+1}}(H)$ and hence $\underbrace{1 \times \ldots \times 1}_{i \text{ times}} \times N_{i+1} \leq \Pi_{\overline{i+1}}(H)$.
Let $g \in \varphi_i(\Pi_{\bar{i}}(H))$. Then there exists $h_1 \in H$ and $r \in R_{i+1}$ such that $g = (\Pi_{\bar{i}}(h_1), r)$ and $\theta_i(\Pi_{\bar{i}}(h_1)) = N_{i+1}r$. Let $h_2 \in H$ be such that $\Pi_{\bar{i}}(h_1) = \Pi_{\bar{i}}(h_2)$. Then

$$g\Pi_{\overline{i+1}}(h_2)^{-1} = (1, r\pi_{i+1}(h_2)^{-1}) \leq \underbrace{1 \times \ldots \times 1}_{i \text{ times}} \times N_{i+1} \leq \Pi_{\overline{i+1}}(H).$$

So $g \in \Pi_{\overline{i+1}}(H)$.
$\leq$: Let $h \in H$. Let $K := \langle \varphi_i(\Pi_{\bar{i}}(H)), \underbrace{1 \times \ldots \times 1}_{i \text{ times}} \times N_{i+1} \rangle$. Let $r \in R_{i+1}$ be such that $N_{i+1}r = N_{i+1}\pi_{i+1}(h)$. Then $(\Pi_{\bar{i}}(h), r) \in \varphi_i(\Pi_{\bar{i}}(H)) \subseteq K$ and

$$\Pi_{\overline{i+1}}(h)(\Pi_{\bar{i}}(h), r)^{-1} = (1, \pi_{i+1}(h)r^{-1}) \in \underbrace{1 \times \ldots \times 1}_{i \text{ times}} \times N_{i+1} \leq K.$$

So $\Pi_{\overline{i+1}}(h) \in K$. □

We require that we always have $1 \in R_{i+1}$. We use Proposition 1.2.6 and Theorem 1.4.3 to describe the structure of intransitive groups.

Corollary 1.4.4. *Let $H \leq \mathrm{Sym}(\Omega)$ and let $\Omega_1, \Omega_2, \dots, \Omega_k$ be the orbits of H. For $1 \leq i \leq k$, let $\Delta_i = \cup_{j \leq i} \Omega_j$. Then for $1 \leq i \leq k-1$, the following hold.*

1. *Let $N_{i+1} \coloneqq (H_{(\Delta_i)})|_{\Omega_{i+1}}$. Then $N_{i+1} \trianglelefteq H|_{\Omega_{i+1}}$.*
2. *Let $\theta_i : H|_{\Delta_i} \to (H|_{\Omega_{i+1}})/N_{i+1}$ be defined by $h|_{\Delta_i} \mapsto N_{i+1}(h|_{\Omega_{i+1}})$ for all $h \in H$. Then θ_i is a surjective homomorphism.*
3. *Let R_{i+1} be a transversal of N_{i+1} in $H|_{\Omega_{i+1}}$. By considering $\mathrm{Sym}(\Delta_i)$ and $\mathrm{Sym}(\Omega_{i+1})$ as subgroups of $\mathrm{Sym}(\Delta_{i+1})$, let*
$$\begin{aligned} \varphi_i : H|_{\Delta_i} &\to \mathrm{Sym}(\Delta_{i+1}) \\ h|_{\Delta_i} &\mapsto h|_{\Delta_i} r \quad \text{if } r \in R_{i+1} \text{ and } \theta_i(h|_{\Delta_i}) = N_{i+1} r_{i+1}. \end{aligned}$$
Let $\overline{N_{i+1}}$ denote the subgroup of $\mathrm{Sym}(\Delta_{i+1})$ with support Ω_{i+1} such that $\overline{N_{i+1}}|_{\Omega_{i+1}} = N_{i+1}$. Then $H|_{\Delta_{i+1}} = \langle \varphi_i(H|_{\Delta_i}), \overline{N_{i+1}} \rangle$.

Since the preceeding theorem is rather technical, we give an example.

Example 1.4.5. Let $x_1 \coloneqq (1,2,3)(7,9,8)(10,12,11)$, $x_2 \coloneqq (4,5,6)(7,8,9)(10,11,12)$, $x_3 \coloneqq (5,6)(8,9)(11,12)$ and $x_4 \coloneqq (7,8,9)(10,11,12)$. Let $H \coloneqq \langle x_1, x_2, x_3, x_4 \rangle \leq S_{12}$. Then $\Omega_1 = \{1,2,3\}$, $\Omega_2 = \{4,5,6\}$, $\Omega_3 = \{7,8,9\}$ and $\Omega_4 = \{10,11,12\}$ are the orbits of H. For $1 \leq i \leq 4$, let $G_i = H|_{\Omega_i}$. Then H is a subdirect product of $G_1 \times G_2 \times G_3 \times G_4$, where we identify the direct product as a subgroup of S_{12}.
For $1 \leq i \leq 4$, let $\Delta_i \coloneqq \cup_{j \leq i} \Omega_j$. Then $H_{(\Delta_2)} = \langle (7,8,9)(10,11,12) \rangle$. Let N_3 be as in Corollary 1.4.4, we see that $N_3 = \langle (7,8,9) \rangle$ is normal in G_3.
Let $\theta_2 : H|_{\Delta_2} \to G_3/N_3$ be as in Corollary 1.4.4. Then $\theta_2(x_1|_{\Delta_2}) = \theta_2((1,2,3)) = \theta_2(x_2|_{\Delta_2}) = \theta_2((4,5,6)) = N_3$ and $\theta_2(x_3|_{\Delta_2}) = \theta_2((5,6)) = N_3(8,9)$. So θ_2 is surjective.
Let $R_3 \coloneqq \{(), (8,9)\}$ be a transversal of N_3 in G_3. Let φ_2 be as in Corollary 1.4.4. Then $\varphi_2(H|_{\Delta_2}) = \langle (1,2,3), (4,5,6), (5,6)(8,9) \rangle$, and one could check that indeed we have $\langle \varphi_2(H|_{\Delta_2}), 1_{\Delta_2} \times N_3 \rangle = H|_{\Delta_3}$.

Lastly, note an elementary corollary of Corollary 1.4.4.

Corollary 1.4.6. *Let $H \leq \mathrm{Sym}(\Omega)$ and let $\Omega_1, \Omega_2, \dots, \Omega_k$ be the orbits of H. Let $1 \leq i \leq k-1$.*

1. *If $(H_{(\Delta_i)})|_{\Omega_{i+1}} = H|_{\Omega_{i+1}}$, then $H|_{\Delta_{i+1}} = H|_{\Delta_i} \times H|_{\Omega_{i+1}}$, where we identify the direct product as a subgroup of $\mathrm{Sym}(\Delta_{i+1})$.*
2. *If $(H_{(\Delta_i)})|_{\Omega_{i+1}} = 1$, then $H|_{\Delta_{i+1}} = \{h' \theta_i(h') \mid h' \in h|_{\Delta_i}\}$.*

Proof. Part 1: Using the notation of Corollary 1.4.4, we have $N_{i+1} = H|_{\Omega_{i+1}}$, and so $\mathrm{Im}(\theta_{i+1}) = 1$. Then by taking $R_{i+1} = \{1\}$, the image $\varphi_i(H|_{\Delta_i})$ has support Δ_i. So $\varphi_{i+1}(H|_{\Delta_i}) = H|_{\Delta_i}$. Therefore $H|_{\Delta_{i+1}} = \langle H|_{\Delta_i}, H|_{\Omega_{i+1}} \rangle$. Since $H|_{\Delta_i}$ and $H|_{\Omega_{i+1}}$ have disjoint supports, we have $H|_{\Delta_{i+1}} = H|_{\Delta_i} \times H|_{\Omega_{i+1}}$.
Part 2: This follows from Corollary 1.4.4 since $N_{i+1} = 1$. □

1.5 Base and strong generating sets

The base and strong generating set of a permutation group G are important for computing with G. In fact, many permutation group algorithms start with computing a base and a corresponding strong generating set. In this section, we introduce some basic definitions. We will see how bases and strong generating sets can be used to compute with permutation groups later in Chapter 2.

Definition 1.5.1. Let $G \leq \mathrm{Sym}(\Omega)$. A *base* B of G is a tuple $(\beta_1, \beta_2, \ldots, \beta_m) \in \Omega^m$ such that the pointwise stabiliser $G_{(\beta_1,\beta_2,\ldots,\beta_m)} = 1$. A base is said to be *non-redundant* or *irredundant* if each $G_{(\beta_1,\beta_2,\ldots,\beta_{i+1})}$ is a proper subgroup of $G_{(\beta_1,\beta_2,\ldots,\beta_i)}$.
Let $g \in G$. The *base image* of g relative to a base B is the tuple $B^g := (\beta_1^g, \beta_2^g, \ldots, \beta_m^g)$.

If two elements $g, h \in G$ have the same base image then gh^{-1} fixes B and so is trivial. Therefore the base image B^g of g uniquely determines $g \in G$. This means that elements of G can be represented by $|B|$-tuples over Ω.

A base of G defines a subgroup chain of G.

Definition 1.5.2. A *stabiliser chain* defined by a base $B = (\beta_1, \beta_2, \ldots, \beta_m)$ of G is a subgroup chain

$$1 = G^{[m+1]} \leq G^{[m]} \leq \ldots \leq G^{[2]} \leq G^{[1]} = G,$$

where each $G^{[i]} = G_{(\beta_1,\beta_2,\ldots,\beta_{i-1})}$ is the pointwise stabiliser of the first $i-1$ base points.

A strong generating set is a special type of generating set where the elements that lie in the stabiliser $G^{[i]}$ form a generating set for $G^{[i]}$. Therefore we may extract the groups in the stabiliser chain from a strong generating set.

Definition 1.5.3. A generating set S of $G \leq \mathrm{Sym}(\Omega)$ relative to a base B is called a *strong generating set* if $\langle S \cap G^{[i]} \rangle = G^{[i]}$ for $1 \leq i \leq m+1$.

We call the orbits $\beta_i^{G^{[i]}}$ the *fundamental orbits* of G. For each point in the orbit, it is often useful to store an element in $G^{[i]}$ which maps β_i to that point. These elements form a transversal R_i for $G^{[i+1]}$ in $G^{[i]}$. Each transversal R_i is stored in a structure called a Schreier tree.

Definition 1.5.4. A *Schreier tree* of $G \leq \mathrm{Sym}(\Omega)$ rooted at $\alpha \in \Omega$ is a directed tree with root α, where its vertices are labelled by elements of the orbit α^G. Each edge is directed from a point $\gamma \in \Omega$ at depth m to some point $\beta \in \Omega$ at depth $m-1$, and is labelled with a group element $g \in G$ such that $\gamma^g = \beta$.

Let $\alpha, \beta \in \Omega$ be distinct points in the same orbit of G. Let T be a Schreier tree of G rooted at α and let g be the product of the labels of the path from β to α in T. Then g^{-1} is an element of G which maps α to β.

We will see in Theorem 2.1.7 that a base and strong generating set of a group can be computed in polynomial time, which requires polynomial time construction of Schreier trees. In Section 2.1.2, we shall see how the polynomial time computation of base and strong generating sets leads to other polynomial time algorithms. Then in Section 2.3, we shall see how the base and strong generating sets give us means to systematically search in a group.

Chapter 2

Permutation Group Algorithms

In this chapter, we will introduce some permutation group algorithms. In Section 2.1, we will present some elementary polynomial time results. In Section 2.2, we state some problems that have no known polynomial time solution and present the complexity hierarchy of these problems. In practice, we solve these problems using backtrack search, which we shall describe in Section 2.3. Lastly, in Section 2.4, we consider the normaliser problem and discuss the problem in terms of its complexity and its practical computation.

2.1 Polynomial time algorithms

In this section, we present some polynomial time results we will use in later chapters. We will assume that any permutation group $G \leq S_n$ is given by a generating set X of G. This gives an input size of $O(|X|n)$[1], so a polynomial time algorithm should have complexity $O((|X|n)^c)$ for some constant c.

Theorem 2.1.1 ([Ser03, Theorem 10.1.3]). *Let $H = \langle X \rangle \leq S_n$. Then we may replace X with a generating set of size at most n in time $O(|X|n^2 + n^5)$.*

Therefore we may assume that all given generating sets have size at most n and measure complexity in terms of n.

Recall base and strong generating set from Definitions 1.5.1 and 1.5.3. Many permutation group algorithms require the computation of a base and a strong generating set relative to it. We start by giving some polynomial time results which do not require a base and strong generating set.

Proposition 2.1.2 ([Ser03, Theorem 2.1.1], [Atk75]). *Given $G = \langle X \rangle \leq S_n$, then in polynomial time, we can*

1. *compute the G-orbits;*

2. *decide if G is primitive and if not, output a non-trivial block system.*

[1]We follow the convention in [Ser03] where we do not count the $O(\log n)$ representation of integers $i \leq n$.

2.1.1 The sifting procedure

Recall the definition of a Schreier tree from Definition 1.5.4. Let $G \leq \mathrm{Sym}(\Omega)$ and let $B := (\beta_1, \beta_2, \ldots, \beta_m) \in \Omega^m$ be a base of G. Recall from Definition 1.5.2 the subgroups $G^{[i]}$ for all $1 \leq i \leq m+1$. For $1 \leq i \leq m$, let R_i be a transversal of $G^{[i+1]}$ in $G^{[i]}$. Then the Schreier tree of $G^{[i]}$ rooted at β_i can be used to compute R_i. These Schreier trees can be computed in polynomial time.

Theorem 2.1.3 ([BCFS91, CF94]). *Let $G = \langle X \rangle \leq \mathrm{Sym}(\Omega)$ and let $\alpha \in \Omega$. Then*

1. *there exists a deterministic algorithm which computes a Schreier tree of depth at most $2 \log |G|$ for the transversal of G_α in G in $O(n \log^2 |G| + |X|n)$ time, and*

2. *there exists a randomised algorithm which, with probability at least $1 - |\alpha^G|^{-0.29c}$, computes a Schreier tree of depth at most $2 \log |\alpha^G| + 4$ for the transversal of G_α in G using $c(8 \log |\alpha^G| + 16)$ random elements, where $c > 1$.*

The Schreier tree is useful in testing if a given permutation $h \in S_n$ is in a group $G \leq S_n$. First observe that any element $g \in G$ can be uniquely written as a product $g = r_m r_{m-1} \ldots r_1$, where each $r_i \in R_i$. These r_i can be determined by a procedure called *sifting*.

Definition 2.1.4. Let h be a given permutation in S_n. The *sifting of h by G* is as follows. We initialise $g_1 := h$. For $1 \leq i \leq m$, we recursively find $r_i \in R_i$ such that $\beta_i^{r_i} = \beta_i^{g_i}$, and setting $g_{i+1} := g_i r_i^{-1}$. The procedure terminates when either

1. $1 \leq s \leq m$ and there are no $r_s \in R_s$ such that $\beta_s^{r_s} = \beta_s^{g_s}$, or

2. $s = m+1$ and we have computed g_s.

In both cases, g_s is a *siftee* of h by G.

We may conclude that $h \in G$ if we get a siftee $g_{m+1} = 1$. Observe that for $1 \leq i \leq s-1$, the permutation g_{i+1} fixes β_i. Then as β_i is fixed by all R_j for $j \geq i$, a siftee $g_s = g r_1^{-1} r_2^{-1} \ldots r_{s-1}^{-1}$ also fixes β_i.

We give an example of the sifting procedure. In the example, we will write bases with square brackets to differentiate between bases and permutations.

Example 2.1.5. Let $X := \{(1,2,3,4,5), (2,5)(3,4)\}$ and $G := \langle X \rangle$. Then X is a strong generating set of G relative to base $B := [1,2]$. Let $R_1 := \{1, (1,5,4,3,2), (1,4,2,5,3), (1,2)(3,5), (1,3,5,2,4)\}$ and $R_2 := \{1, (2,5)(3,4)\}$ be the transversals for $G_{(1)}$ in G and for $G_{(1,2)}$ in $G_{(1)}$ respectively. Consider sifting $h = (1,2,4,5)$ by G. Initialise $g_1 := h$. Then $r_1 := (1,2)(3,5)$ is an element of R_1 mapping 1 to $1^{g_1} = 2$. So $g_2 = g_1 r_1^{-1} = (2,4,3,5)$. Now there is no $r_2 \in R_2$ mapping 2 to $2^{g_2} = 4$. Therefore we get a siftee $(2,4,3,5)$.

Lastly we show that we can determine if a given tuple is a base image in polynomial time.

Lemma 2.1.6. *Let $G = \langle X \rangle \leq \mathrm{Sym}(\Omega)$ and let $B = (\beta_1, \beta_2, \ldots, \beta_m)$ be a base of G. Given a tuple $A = (\alpha_1, \alpha_2, \ldots, \alpha_m) \in \Omega^m$, in polynomial time, we can determine if A is a base image of B in G and if so, output $g \in G$ such that $B^g = A$.*

Proof. For $1 \leq i \leq m$, let R_i be a transversal of $G^{[i+1]}$ in $G^{[i]}$, which can be computed in polynomial time by Theorem 2.1.3. Find $r_1 \in R_1$ such that $\beta_1^{r_1} = \alpha_1$ and set $g_1 = r_1$. Then for $2 \leq i \leq m$, recursively find $r_i \in R_i$ such that $\beta_i^{r_i g_{i-1}} = \alpha_i$ and let $g_i = r_i g_{i-1}$, or return fail if such an r_i does not exists. Then $g_m = r_m r_{m-1} \ldots r_1$ is an element of G. Since each r_{i+1} fixes each of $\beta_1, \beta_2, \ldots, \beta_i$, we have

$$\beta_i^{g_m} = \beta_i^{r_m r_{m-1} \ldots r_i g_{i-1}} = \beta_i^{r_i g_{i-1}} = \alpha_i,$$

for all $1 \leq i \leq m$. Therefore $B^{g_m} = A$. If we fail to compute g_m then there is no $g \in G$ such that $A^g = G$. Since each $|R_i| \leq |\Omega|$, the algorithm runs in polynomial time. $\square$

2.1.2 Schreier-Sims algorithm and its consequences

The Schreier-Sims algorithm computes a irredundant base and corresponding strong generating set of a given group $G \leq S_n$. We will not describe the Schreier-Sims algorithm here, but note that it has polynomial time complexity.

Theorem 2.1.7 ([Ser03, Theorem 4.2.4])**.** *Let $G = \langle X \rangle \leq S_n$. Then a base and a strong generating set for G can be computed in $O(n^2 \log^3 |G| + |X| n^2 \log |G|)$ time using $O(n^2 \log |G| + |X| n)$ memory. Alternatively, a strong generating set for G can be computed in time $O(n^3 \log^3 |G| + |X| n^3 \log |G|)$ time using $O(n \log^2 |G| + |X| n)$ memory.*

A *Monte Carlo algorithm* is a randomised algorithm that will produce a wrong answer with a small probability, while a *Las Vegas algorithm* is a randomised algorithm that will always output the correct results but may fail. In practice, implementations of the Schreier-Sims algorithm use the randomised version.

Theorem 2.1.8 ([Ser03, Theorem 4.5.5])**.** *Let $G = \langle X \rangle \leq S_n$. Then there exists a Monte Carlo algorithm for computing a base and a strong generating set for G in time $O(n \log n \log^4 |G| + |X| n \log |G|)$ and space $O(n \log |G| + |X| n)$, with a probability of error less that $1/n^d$, for a constant d.*

As a consequence of a polynomial time computation of bases and strong generating sets, we get the following polynomial time results.

Theorem 2.1.9. *Let $G = \langle X \rangle$ be a subgroup of $\mathrm{Sym}(\Omega)$ where $|\Omega| = n$. Let $B \in \Omega^m$ be a base of G. Then the following can be done in polynomial time.*

1. *Compute the order $|G|$.*
2. *Given $h \in S_n$, compute a siftee of h by G, and decide if $h \in G$.*
3. *Given $\Delta \subseteq \Omega$, compute the pointwise stabiliser $G_{(\Delta)}$.*

Let $\phi : G \to \mathrm{Sym}(\Delta)$ be a map given by the images of X.

4. *Decide if ϕ defines a homomorphism.*

5. *Decide if ϕ defines an isomorphism.*

6. *Decide if ϕ defines an automorphism.*

Let $\varphi : G \to \mathrm{Sym}(\Delta)$ be a homomorphism given by the images of X.

7. *Compute* $\mathrm{Ker}(\varphi)$.

8. *Compute $\varphi(g)$ for a given $g \in G$.*

9. *Find $g \in G$ such that $\varphi(g) = h$ for a given $h \in \mathrm{Im}(\varphi)$.*

For Part 1, let B be a non-redundant base of G. Then by repeated application of the orbit-stabiliser theorem, $|G| = \prod_{i=1}^{m} |G_{(\beta_1,\ldots,\beta_{i-1})} : G_{(\beta_1,\ldots,\beta_i)}|$. Part 2 is in polynomial time by using the sifting procedure described in Section 2.1.1. Parts 3, 4 and 7 to 9 are discussed in [Ser03, Sections 5.1.1 & 5.1.2]. For Part 5, ϕ gives an isomorphism if ϕ defines a homomorphism and $|\mathrm{Im}(\varphi)| = |\langle \phi(x) \mid x \in X \rangle| = |G|$. For Part 6, we check that ϕ defines an isomorphism, and that $\phi(x) \in G$ for all $x \in X$.

2.1.3 Equivalent orbits and centralisers

It is well known that the centraliser $C_{S_n}(G)$ of a group $G \leq S_n$ can be computed in polynomial time [CFL89]. In this section, we will introduce orbit equivalence, which plays a crucial role in computing centralisers of intransitive groups. We will see how equivalent orbits can be used for normaliser calculations in later chapters. For the rest of the section, we will take the natural inclusion of $\mathrm{Sym}(\Delta)$ into $\mathrm{Sym}(\Omega)$, for all subsets Δ of Ω.

Definition 2.1.10. Two orbits Ω_i, Ω_j of $H \leq S_n$ are *equivalent* if there exists a bijection $\psi : \Omega_i \to \Omega_j$ such that for all $h \in H$ and $\delta \in \Omega_i$, we have $\psi(\delta^h) = \psi(\delta)^h$. We denote this by $\Omega_i \equiv_o \Omega_j$, and say that ψ *witneses* the equivalence.

Let Ω_i and Ω_j be H-orbits. By taking ρ and σ in Definition 1.1.4 to be the restrictions of H to Ω_i and Ω_j respectively, we see that $\Omega_i \equiv_o \Omega_j$ if and only if the actions of H on Ω_i and Ω_j are equivalent.

Notation 2.1.11. Let $\varphi : \Omega_i \to \Omega_j$ be a bijection for some $1 \leq i, j \leq k$. We denote by $\overline{\varphi}$ the involution g in S_n with support $\Omega_i \cup \Omega_j$ such that $\alpha^{\overline{\varphi}} = \varphi(\alpha)$.

We will often use the following result to detect equivalent orbits.

Lemma 2.1.12. *Let Ω_i and Ω_j be orbits of $H \leq \mathrm{Sym}(\Omega)$. Then $\Omega_i \equiv_o \Omega_j$ if and only if there exists an involution $g \in \mathrm{Sym}(\Omega)$ with support $\Omega_i \cup \Omega_j$ such that*

$$h|_{\Omega_j} = (h|_{\Omega_i})^g \quad \text{for all } h \in H, \tag{2.1}$$

where we identify $\mathrm{Sym}(\Omega_i)$ *and* $\mathrm{Sym}(\Omega_j)$ *with subgroups of* S_n.

Proof. $\Rightarrow$: Let $\psi : \Omega_i \to \Omega_j$ be a bijection witnessing $\Omega_i \equiv_o \Omega_j$. Then by letting $g = \overline{\psi}$, Equation (2.1) holds.
$\Leftarrow$: Let $\psi : \Omega_i \to \Omega_j$ be defined by $\psi(\delta) = \delta^g$ for all $\delta \in \Omega_i$. Then for all $h \in H$ and $\delta \in \Omega_i$ we have

$$\psi(\delta)^h = \delta^{gh} = \delta^{g(h|_{\Omega_j})} = \delta^{g(g^{-1}(h|_{\Omega_i})g)} = \delta^{(h|_{\Omega_i})g} = \delta^{hg} = \psi(\delta^h).$$

□

Observe that $\equiv_o$ is an equivalence relation. Proposition 2.1.13 describes the relationship between equivalent orbits and the centraliser of intransitive groups.

Proposition 2.1.13 ([Ser03, Lemma 6.1.8]). *Let $H \leq S_n$ be an intransitive group with orbits $\Omega_1, \Omega_2, \ldots, \Omega_k$. Let $\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_t$ be the $\equiv_o$-classes of the H-orbits. For $1 \leq i \leq t$, let $\mathcal{B}_i := \{\Omega_{i1}, \Omega_{i2}, \ldots, \Omega_{i|\mathcal{B}_i|}\}$. For $1 \leq i \leq t$ and $2 \leq b \leq |\mathcal{B}_i|$, let $\psi_{ib} : \Omega_{i1} \to \Omega_{ib}$ be a bijection witnessing the orbit equivalence. Let $B_i := \langle \overline{\psi}_{ib} \mid 2 \leq b \leq |\mathcal{B}_i| \rangle$, where $\overline{\psi}_{ib}$ is the involution in S_n, as in Notation 2.1.11. For $1 \leq i \leq k$, let C_i be the subgroup of S_n with support Ω_i such that $C_i|_{\Omega_i} = C_{\mathrm{Sym}(\Omega_i)}(H|_{\Omega_i})$.*
Then

$$\begin{aligned} C_{S_n}(H) &= \langle C_1 \times C_2 \times \ldots \times C_k, B_1 \times B_2 \times \ldots \times B_t \rangle \\ &\cong \prod_{i=1}^{t} C_{\mathrm{Sym}(\Omega_{i1})}(H|_{\Omega_{i1}}) \wr S_{|\mathcal{B}_i|}. \end{aligned}$$

A permutation group algorithm with input $H = \langle X \rangle$ is in *nearly-linear time* if it runs in time $O(n|X|\log^c |G|)$. To obtain the polynomial time computation of $C_{S_n}(H)$, we first state the nearly-linear time centraliser computation for transitive groups.

Theorem 2.1.14 ([Ser03, Theorem 6.1.6]). *Let $G = \langle X \rangle \leq \mathrm{Sym}(\Omega)$ be transitive. Then given an irredundant base B of G and a strong generating set S relative to B, the centraliser $C_{\mathrm{Sym}(\Omega)}(G)$ can be computed in nearly linear time.*

Next we show that we can decide if two orbits are equivalent in polynomial time. Recall the notation of $Fix(H)$ from Definition 1.1.9.

Lemma 2.1.15 ([Ser03, Lemma 6.1.9]). *Let Ω_i and Ω_j be H-orbits of equal size and let $\alpha \in \Omega_i$. Then $\Omega_i \equiv_o \Omega_j$ if and only if $Fix(H_\alpha) \cap \Omega_j \neq \emptyset$.*
Hence for two distinct orbits Ω_i and Ω_j of H, in polynomial time, we can decide if $\Omega_i \equiv_o \Omega_j$ and if so, exhibit a bijection φ that witnesses it.

Proof. The first part of the result is shown in [Ser03, Lemma 6.1.9]. The last assertion follows since point stabilisers and their fixed points can be computed in polynomial time. □

Therefore we get a polynomial time result for computing centralisers in the symmetric groups.

Theorem 2.1.16. *Let $G = \langle X \rangle \leq \mathrm{Sym}(\Omega)$. Then $C_{\mathrm{Sym}(\Omega)}(G)$ can be computed in polynomial time.*

Proof. By Theorem 2.1.7, a base and a corresponding strong generating set of G can be computed in polynomial time. Then the result follows from Proposition 2.1.13, Theorem 2.1.14 and Lemma 2.1.15. □

2.1.4 Other polynomial time algorithms

We end this section by giving some other polynomial time results we will be using later in the thesis.

Proposition 2.1.17 ([BKL83, KL90])**.** *Let $G = \langle X \rangle \leq S_n$. Then we can compute the socle soc(G) of G in polynomial time.*

Proposition 2.1.18 ([Kan85])**.** *Let $G = \langle X \rangle \leq S_n$ and let p be prime. Then a Sylow p-subgroup of G can be computed in polynomial time.*

Next, we show that we can decide if an isomorphism between two subgroups of S_m is induced by conjugation in polynomial time. Since the proof in [LM11] aims only to prove the theoretical time complexity and we are also interested in a practical algorithm, we give an alternative proof for the result here.

Lemma 2.1.19 ([LM11, Lemma 3.5])**.** *Let $G = \langle X \rangle$ and $H = \langle Y \rangle$ be subgroups of S_m, given by their generators. Let $\varphi : G \to H$ be an isomorphism, given by the images of X. Then in polynomial time, we can decide if there exists $c \in S_m$ such that $\varphi(g) = g^c$ for all $g \in G$ and if such a c exists, output one such c.*
Hence, given an automorphism φ of $G = \langle X \rangle$ by the images of X, in polynomial time, we can decide if there exists $c \in S_m$ such that $\varphi(g) = g^c$ for all $g \in G$ and if such a c exists, output one such c.

Proof. By Part 5 of Theorem 2.1.9, we can check if φ is indeed an isomorphism. Let $d \in S_{2m}$ be an involution such that $\{1, 2, \ldots, m\}^d = \{m+1, m+2, \ldots 2m\}$. Consider the group $G_2 := \{g\varphi(g)^d \mid g \in G\} \leq S_{2m}$.
We first show that there exists a $c \in S_m$ such that $\varphi(g) = g^c$ for all $g \in G$ if and only if there exists a $b \in C_{S_{2m}}(G_2)$ such that $\{1, 2, \ldots, m\}^b = \{m+1, m+2, \ldots, 2m\}$.
$\Rightarrow$: Consider c as an element of S_m with support contained in $\{1, 2, \ldots, m\}$. Let $b = cdc^{-1}$. Then $\{1, 2, \ldots, m\}^b = \{m+1, m+2, \ldots, 2m\}$. We shall show that $b \in C_{S_{2m}}(G_2)$.

Let $g\varphi(g)^d \in G_2$. Then $g \in G$ and

$$\begin{aligned}
(g\varphi(g)^d)^{cdc^{-1}} &= (gg^{cd})^{cdc^{-1}} && \text{since } \varphi(g) = g^c \\
&= (g^c g^{cd})^{dc^{-1}} && \text{since } c \text{ pointwise stabilises } Supp(g^{cd}) \\
&= (g^{cd}g^c)^{c^{-1}} && \text{since } d \text{ is an involution} \\
&= g^{cd}g.
\end{aligned}$$

Finally, since $Supp(g^{cd})$ and $Supp(g)$ are disjoint, g^{cd} and g commute, so

$$(g\varphi(g)^d)^{cdc^{-1}} = g^{cd}g = gg^{cd} = g\varphi(g)^d.$$

Hence $b = cdc^{-1} \in C_{S_{2m}}(G_2)$.
$\Leftarrow$: Let $g\varphi(g)^d \in G_2$. Then $g\varphi(g)^d = (g\varphi(g)^d)^b = g^b\varphi(g)^{db}$. Since $Supp(g)$ and $Supp(\varphi(g)^{db})$ are subsets of $\{1, 2, \ldots, m\}$ and $Supp(\varphi(g)^d)$ and $Supp(g^b)$ are subsets of $\{m+1, m+2, \ldots, 2m\}$, we have $g^b = \varphi(g)^d$. Then $g^{bd^{-1}} = \varphi(g)$. So φ is induced by conjugation by $c := bd^{-1}$.

To test whether such a c exists, we construct G_2 and look for a centralising element b that maps $\{1, 2, \ldots, m\}$ to $\{m+1, m+2, \ldots, 2m\}$. Let $\mathcal{O}_1$ be the set consisting of all orbits of G_2 contained in $\{1, 2, \ldots, m\}$. Similarly, let $\mathcal{O}_2$ be the set consisting of all orbits of G_2 contained in $\{m+1, m+2, \ldots, 2m\}$. As a consequence of Proposition 2.1.13, a centralising element that maps $\{1, 2, \ldots, m\}$ to $\{m+1, m+2, \ldots, 2m\}$ exists if and only if there exists a bijection $\gamma : \mathcal{O}_1 \to \mathcal{O}_2$ such that $\omega \equiv_o \gamma(\omega)$ for all $\omega \in \mathcal{O}_1$.
We attempt to construct such a γ by considering each $\omega_1 \in \mathcal{O}_1$ in turn, setting $\gamma(\omega_1) = \omega_2$ if we find an orbit $\omega_2 \in \mathcal{O}_2$ equivalent to ω_1, and the procedure fails if no such ω_2 exists. Since the orbit equivalence is an equivalence relation, we would never have to backtrack. By Lemma 2.1.15, deciding if two orbits are equivalent can be done in polynomial time. As there are polynomially many pairs of orbits, deciding the existence of such a γ can be done in polynomial time. Furthermore, if such a γ exists, by Lemma 2.1.15, in polynomial time, we can compute a centralising element $b \in C_{S_{2m}}(G_2)$ such that $\{1, 2, \ldots, m\}^b = \{m+1, m+2, \ldots, 2m\}$. $\square$

2.2 Problems not known to be in polynomial time

In this section, we will introduce some permutation group and combinatorial problems with no known polynomial time solution, and describe the relationship between these problems. In particular, we will describe parts of the complexity hierarchy from [Luk93].

We start with the graph automorphism problem and Babai's quasipolynomial time solution for it. A *graph* Γ is a pair (V, E) where V is a set and E is a set of 2-subsets of V. We call V and E the *vertices* and the *edges* of graph Γ respectively. The *automorphism group* $\mathrm{Aut}(\Gamma)$ of Γ consists of all permutations α of V such that $\{v^\alpha, w^\alpha\} \in E$ for all $\{v, w\} \in E$.

Problem 2.2.1 (Graph-Aut)**.** Given a graph $\Gamma = (V, E)$, compute its automorphism group $\mathrm{Aut}(\Gamma)$.

Theorem 2.2.2 ([Bab15])**.** Graph-Aut *(Problem 2.2.1) can be computed in time* $2^{O(\log^c |V|)}$*, for some constant c.*

Next we introduce three problems in permutation groups and see how they are related to Graph-Aut. For the rest of the section, we will assume each permutation group is given by a generating set, and we measure complexity of permutation group algorithms in terms of degrees.

Problem 2.2.3 (Set-Stab)**.** Given $G \leq \mathrm{Sym}(\Omega)$ and $\Delta \subseteq \Omega$, compute the setwise stabiliser $G_{\{\Delta\}}$.

Problem 2.2.4 (Cent)**.** Given $H, G \leq \mathrm{Sym}(\Omega)$, compute the centraliser $C_G(H)$.

Problem 2.2.5 (Inter)**.** Given $H, G \leq \mathrm{Sym}(\Omega)$, compute the intersection $G \cap H$.

The graph automorphism problem is polynomial time reducible to each of the (polynomial time equivalent) problems above.

Theorem 2.2.6 ([Luk93])**.**

1. Graph-Aut *(Problem 2.2.1) is polynomial time reducible to* Set-Stab *(Problem 2.2.3).*
2. Set-Stab*,* Cent *and* Inter *(Problems 2.2.3, 2.2.4, 2.2.5) are polynomial time equivalent to each other.*

Babai, in [Bab15], gives a quasipolynomial solution to the string isomorphism problem (Problem 2.2.7), which gives quasipolynomial time solution to Set-Stab.

Problem 2.2.7 (String-Isom)**.** Let $G \leq \mathrm{Sym}(\Omega)$. Let Σ be a finite alphabet. Given functions $x : \Omega \to \Sigma$ and $y : \Omega \to \Sigma$, find $\{g \in G \mid x^g = y\}$, where the action of G on functions is defined by $z^g(\alpha) = z(\alpha^{g^{-1}})$ for all functions $z : \Omega \to \Sigma$ and $\alpha \in \Omega$.

Note that since $\{g \in G \mid x^g = y\}$ forms a coset of the stabiliser of the string x, the output of Problem 2.2.7 can be expressed as a generators of such a stabiliser together with a coset representative, hence Problem 2.2.7 has a polynomial size output.

Theorem 2.2.8 ([Bab15])**.** *There exists a constant c such that the string isomorphism problem (Problem 2.2.7) can be solved on* $2^{O(\log(|\Omega|)^c)}$*.*

Corollary 2.2.9. Set-Stab*,* Cent *and* Inter *(Problems 2.2.3, 2.2.4, 2.2.5) can be solved in quasipolynomial time.*

Proof. We show that Set-Stab is a special case of the string isomorphism problem, then the result follows from Part 2 of Theorem 2.2.6 and Theorem 2.2.8.
Let $\Delta \subset \Omega$ and let $\Sigma := \{0, 1\}$. Let $x : \Omega \to \Sigma$ be defined by $x(\alpha) = 1$ if $\alpha \in \Delta$ and $x(\alpha) = 0$ otherwise. Let $g \in G$. Then $x^g = x$ if and only if $x(\alpha^{g^{-1}}) = x(\alpha)$ for all $\alpha \in \Omega$. This is equivalent to $\Delta^{g^{-1}} = \Delta$. Hence the solution of the string isomorphism problem for x and $y := x$ is the setwise stabiliser $G_{\{\Delta\}}$. □

A permutation group problem that is believed to be strictly harder than SET-STAB is the normaliser problem.

Problem 2.2.10 (NORM). Given $H, G \leq \mathrm{Sym}(\Omega)$, compute $N_G(H)$.

Theorem 2.2.11 ([Luk93]). SET-STAB *(Problem 2.2.3) is polynomial time reducible to* NORM *(Problem 2.2.10).*

It is unknown if NORM is polynomial time reducible to SET-STAB. In Section 2.4 we will present some complexity results for different classes of input groups for NORM, as well as how we solve the problem in practice. Recall that we measure complexity in terms of the degree.

A special case of NORM is the NORM-SYM problem.

Problem 2.2.12 (NORM-SYM). Given $H \leq \mathrm{Sym}(\Omega)$, compute $N_{\mathrm{Sym}(\Omega)}(H)$.

It is clear that NORM-SYM is polynomially reducible to NORM. However, we do not know its relation to SET-STAB. It is however at least as hard as GRAPH-AUT.

Theorem 2.2.13 ([Luk93]). GRAPH-AUT *(Problem 2.2.1) is polynomial time reducible to* NORM-SYM*(Problem 2.2.12).*

In terms of complexity, we now know that it is simply exponential.

Theorem 2.2.14 ([Wie19]). NORM-SYM*(Problem 2.2.12) can be computed in time* $2^{O(|\Omega|)}$.

Next we give the decisional variants of the problems we have mentioned so far.

Problem 2.2.15 (GRAPH-ISOM). Given graphs $\Gamma_1 = (V_1, E_1)$ and $\Gamma_2 = (V_2, E_2)$, decide if Γ_1 and Γ_2 are isomorphic.

Problem 2.2.16 (SET-TRANS). Given $G \leq \mathrm{Sym}(\Omega)$ and $\Delta_1, \Delta_2 \subseteq \Omega$, decide if there exists $g \in G$ such that $\Delta_1^g = \Delta_2$.

Problem 2.2.17 (CONJ-ELT). Given $G \leq \mathrm{Sym}(\Omega)$ and $x, y \in \mathrm{Sym}(\Omega)$, decide if there exists $g \in G$ such that $x^g = y$.

Problem 2.2.18 (DC-EQ). Given $G, H \leq \mathrm{Sym}(\Omega)$ and $x, y \in \mathrm{Sym}(\Omega)$, decide if $GxH = GyH$.

Problem 2.2.19 (CONJ-GROUP). Given $G, H, K \leq \mathrm{Sym}(\Omega)$, decide if there exists $g \in G$ such that $H^g = K$.

Theorem 2.2.20 ([Luk93]).

1. GRAPH-ISOM *(Problem 2.2.15) is polynomial time equivalent to* GRAPH-AUT *(Problem 2.2.1).*
2. SET-TRANS *(Problem 2.2.16) is polynomial time equivalent to* SET-STAB *(Problem 2.2.3).*

3. *CONJ-ELT (Problem 2.2.17) is polynomial time equivalent to* CENT *(Problem 2.2.4).*

4. *DC-EQ (Problem 2.2.18) is polynomial time equivalent to* INTER *(Problem 2.2.5).*

5. *CONJ-GROUP (Problem 2.2.19) is polynomial time equivalent to* NORM *(Problem 2.2.10).*

We do not know the decision problem equivalent to NORM-SYM. The most obvious choice is CONJ-SYM.

Problem 2.2.21 (CONJ-SYM)**.** Given $H, K \leq \mathrm{Sym}(\Omega)$, is there $g \in \mathrm{Sym}(\Omega)$ such that $H^g = K$?

CONJ-SYM is polynomial time reducible to NORM-SYM but we do not know if the converse is true.

Recall that computing the centraliser in the symmetric group can be done in polynomial time, whereas CENT is at least as hard as the graph isomorphism problem. In the case of computing normalisers, we do not know if NORM is strictly harder than NORM-SYM.

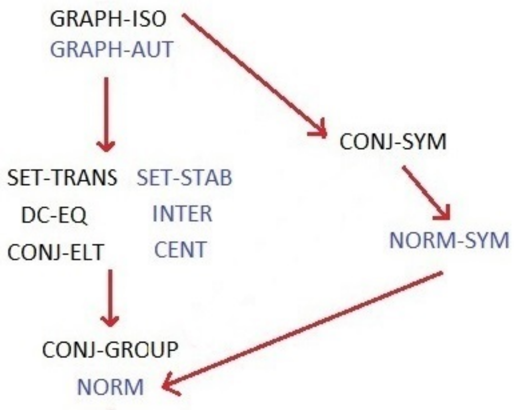


Figure 2.1: Summary of Luks' hierarchy

2.3 Backtrack search in permutation groups

In this section, we will describe backtrack search in permutation groups, which can be used to solve permutation group problems with no known polynomial time algorithms. These include all the problems that are shown to be at least as hard as graph isomorphism (see Section 2.2).

Let $G \leq S_n$ and let $\Omega = \{1, 2, \ldots, n\}$. Suppose that we want to compute elements of G satisfying a given property $\mathcal{P}$. We will assume that we can check if an element $g \in G$ satisfies $\mathcal{P}$ in polynomial time. In general, these elements form a subgroup or a coset of a subgroup of G. For simplicity, we shall assume that these elements form a subgroup $G(\mathcal{P})$ of G. Let $B := (\beta_1,\beta_2, \ldots,\beta_m)$ be a base for G. To systematically search through G, we shall define a search tree of G consisting of partial base images of B.

Definition 2.3.1. Let $B \coloneqq (\beta_1,\beta_2,\ldots,\beta_m)$ be a base for G. Let $g \in G$. The *partial base image* for g of length $r \leq m$ with respect to base B is the tuple $(\beta_1^g, \beta_2^g, \ldots, \beta_r^g)$.

Definition 2.3.2. Let $G \leq S_n$ and let $B \coloneqq (\beta_1,\beta_2,\ldots,\beta_m)$ be a base for G. The *search tree T of G with respect to the base B* is a tree of depth m defined as follows.

1. The root of T is labeled by the empty list $[\,]$.
2. A node at depth $d < m$ is labeled by a partial base image $[\alpha_1,\alpha_2,\ldots,\alpha_d]$ of B.
3. Let t be a node labeled by $[\alpha_1,\alpha_2,\ldots,\alpha_d]$ for $d < m$, then its children have the form $[\alpha_1,\alpha_2,\ldots,\alpha_d,\alpha_{d+1}]$, for all distinct α_{d+1} in $\{\beta_{d+1}^g \mid g \in G \text{ and } \beta_i^g = \alpha_i \text{ for } 1 \leq i \leq d\}$.
4. Furthermore, we set the left-most node at depth d to be labeled by the partial base points $[\beta_1,\beta_2,\ldots,\beta_d]$.

For a search tree T of G with respect to base B, consider the map ϕ mapping each node t of T to the elements represented by the leaves under the subtree rooted at t. We denote the power set of G as $\wp(G)$.

Definition 2.3.3. Define a mapping $\phi : T \to \wp(G)$ by:

$$[\alpha_1,\alpha_2,\ldots,\alpha_d] \mapsto \{g \in G \mid \beta_i^g = \alpha_i \text{ for all } 1 \leq i \leq d\}.$$

Let $t = [\alpha_1, \alpha_2, \ldots, \alpha_d]$ be a node at depth d. Then $\phi(t)$ forms a coset of $G_{(\beta_1,\beta_2,\ldots,\beta_d)}$ containing all elements of G that map $\beta_i^g = \alpha_i$ for all $1 \leq i \leq d$. In particular, if t is the leftmost node at depth d, then $\phi(t)$ is the point stabiliser $G_{(\beta_1,\beta_2,\ldots,\beta_d)}$. Also note that if t is a leaf of T, then $\phi(t)$ contains a unique element $g \in G$.

We can find a representative r of $\phi(t)$ using the transversals of $G_{(\beta_1,\beta_2,\ldots,\beta_i)}$ in $G_{(\beta_1,\beta_2,\ldots,\beta_{i-1})}$ for $1 \leq i \leq d$. Observe that $[\alpha_1,\alpha_2,\ldots,\alpha_d,\alpha_{d+1}]$ is a child of $[\alpha_1,\alpha_2,\ldots,\alpha_d]$ if and only if there exists $gr \in G_{(\beta_1,\beta_2,\ldots,\beta_d)}r$ such that $\beta_{d+1}^{gr} = \alpha_{d+1}$. Therefore the children of $[\alpha_1,\alpha_2,\ldots,\alpha_d]$ are all $[\alpha_1,\alpha_2,\ldots,\alpha_d,\alpha_{d+1}]$ where $(\alpha_{d+1})^{r^{-1}}$ is in the fundamental orbit $\beta_{d+1}^{G_{(\beta_1,\beta_2,\ldots,\beta_d)}}$.

We compute $G(\mathcal{P})$ using Algorithm 1, gathering the elements of G satisfying property $\mathcal{P}$ as S. We initialise S as the identity group and traverse the search tree depth-first. When we reach a leaf node t, we check if the permutation g in $\phi(t)$ satisfies property $\mathcal{P}$. If it does, we update S as $\langle S, g\rangle$, else we backtrack and continue with the search. At the end of the search, we shall have $G(\mathcal{P}) = S$.

Remark 2.3.4. The runtime of the backtrack search is correlated to the number of nodes in the search tree we visited. There are two ways we can reduce this:

Algorithm 1 Backtrack search - main

```
Input: $G \leq S_n$, base $B = (\beta_1, \beta_2, \ldots, \beta_m)$ of $G$, property $\mathcal{P}$
Output: $G(\mathcal{P})$
  Initialise $S = 1$
  RecurseSearch([])
  return $S$

  procedure RecurseSearch($[\alpha_1, \alpha_2, \ldots, \alpha_d]$)
      if $d = m$ then
          LeafCheck($[\alpha_1, \alpha_2, \ldots, \alpha_d]$)
      else
          for each possible image $\alpha_{d+1}$ of $\beta_{d+1}$ do
              RecurseSearch($[\alpha_1, \alpha_2, \ldots, \alpha_{d+1}]$)
          end for
      end if
  end procedure

  procedure LeafCheck($[\alpha_1, \alpha_2, \ldots, \alpha_m]$)
      Get $g \in G$ such that $\beta_i^g = \alpha_i$ for all $1 \leq i \leq m$
      if $g$ satisfies $\mathcal{P}$ then
          $S \leftarrow \langle S, g \rangle$
      end if
  end procedure
```

1. If there exists a proper subgroup U of G containing $G(\mathcal{P})$, then search in U instead.
2. If we can deduce that $\phi(t) \cap G(\mathcal{P}) \subseteq S$ or $\phi(t) \cap G(\mathcal{P}) = \emptyset$, then we can skip traversing the subtree rooted at t.

Reducing the search tree by Part 1 of Remark 2.3.4 is usually problem specific. The methods used for the normaliser problem are discussed in more detail in Chapter 4. We can reduce the search space by deducing that $\phi(t) \cap G(\mathcal{P}) \subseteq S$ using the following lemma.

Lemma 2.3.5. *Let t be a node of depth d in the search tree T. Let $G^{[d+1]} := G_{(\beta_1, \beta_2, \ldots, \beta_d)}$. Suppose that we have computed $R := G^{[d+1]} \cap G(\mathcal{P})$. Let $g \in \phi(t) \cap G(\mathcal{P})$, then $\phi(t) \cap G(\mathcal{P}) \subseteq \langle R, g \rangle$.*

Proof. Let $h \in \phi(t) \cap G(\mathcal{P})$. Since $h \in \phi(t)$ and g is a coset representative of $\phi(t)$, we have $h \in G^{[d+1]}g$ and so $hg^{-1} \in G^{[d+1]}$. Then as $h, g \in G(\mathcal{P})$, we have $hg^{-1} \in G^{[d+1]} \cap G(\mathcal{P}) = R$. Therefore $h \in \langle R, g \rangle$. $\square$

Suppose that we are at a node in the subtree rooted at $[\beta_1, \beta_2, \ldots, \beta_{d-1}, \alpha_d]$, where $\alpha_d \neq \beta_d$. Since we traverse the search tree by depth-first search and the left-most nodes are labelled by partial bases, at this point of search, we have traversed the subtree rooted at $[\beta_1, \beta_2, \ldots, \beta_d]$, and so we have computed $G_{(\beta_1, \beta_2, \ldots, \beta_d)} \cap G(\mathcal{P})$. Hence we may use

Lemma 2.3.5 to skip traversing the rest of the subtree rooted at $[\beta_1, \beta_2, \ldots, \beta_{d-1}, \alpha_d]$. Therefore, we may replace the LeafCheck procedure with LeafCheckWithBacktrack in Algorithm 2.

Algorithm 2 Procedure at a leaf node $t = [\alpha_1, \alpha_2, \ldots, \alpha_m]$

```
procedure LeafCheckWithBacktrack($[\alpha_1, \alpha_2, \ldots, \alpha_m]$)
    Get $g \in G$ such that $\beta_i^g = \alpha_i$ for all $1 \le i \le m$
    if $g$ satisfies $\mathcal{P}$ then
        $S \leftarrow \langle S, g \rangle$
        Find the largest $r$ such that $\alpha_i = \beta_i$ for all $1 \le i \le r$
        if $r$ is defined then
            Backtrack to node $[\alpha_1, \alpha_2, \ldots, \alpha_r]$
        end if
    end if
end procedure
```

Reducing the search space by Part 2 of Remark 2.3.4 is called the *pruning* of the search tree. There are some problem-independent and problem-dependent pruning methods. For example, the orbits of S yield a problem-independent pruning method.

Lemma 2.3.6. *Suppose that $S \coloneqq G_{(\beta_1, \beta_2, \ldots, \beta_{d+1})} \cap G(\mathcal{P})$. Let α_{d+1} and α'_{d+1} be points in the same orbit of S. Then there exists $g \in G_{(\beta_1, \beta_2, \ldots, \beta_d)} \cap G(\mathcal{P})$ such that $\beta_{d+1}^g = \alpha_{d+1}$ if and only if there exists $g' \in G_{(\beta_1, \beta_2, \ldots, \beta_d)} \cap G(\mathcal{P})$ such that $\beta_{d+1}^{g'} = \alpha'_{d+1}$.*

Proof. Let $g \in G_{(\beta_1, \beta_2, \ldots, \beta_d)} \cap G(\mathcal{P})$ such that $\beta_{d+1}^g = \alpha_{d+1}$. Since α_{d+1} and α'_{d+1} are in the same S-orbit, there exists $s \in S$ such that $\alpha_{d+1}^s = \alpha'_{d+1}$. Then gs is an element of $\langle S, g \rangle \subseteq G(\mathcal{P})$ that fixes each of $\beta_1, \beta_2, \ldots, \beta_d$ and maps β_{d+1} to α'_{d+1}. The converse follows by symmetry. $\square$

Suppose that we have just traversed the search tree rooted at $[\beta_1, \beta_2, \ldots, \beta_{d+1}]$ and have backtracked to $[\beta_1, \beta_2, \ldots, \beta_d]$. Then we are now considering branching to a node $[\beta_1, \beta_2, \ldots, \beta_d, \alpha_{d+1}]$ where $\alpha_{d+1} \neq \beta_{d+1}$. Observe that at this time, S is exactly $G_{(\beta_1, \beta_2, \ldots, \beta_{d+1})} \cap G(\mathcal{P})$. Then by Lemma 2.3.6, we only branch on $[\beta_1, \beta_2, \ldots, \beta_d, \alpha_{d+1}]$ if α_{d+1} is the smallest point in its orbit under S. Note that this is a simplified version of some methods in [Ser03, Section 9.1.1].

Pruning methods for the normaliser problem will be discussed in more detail in Chapter 4. For methods for the centraliser, intersection and the setwise stabiliser problem, we refer the reader to [Ser03, Section 9.1.2]. In general, we use a series of boolean functions called *pruning tests.*

Definition 2.3.7. A boolean function $\mathcal{T} : T \to \mathit{Bool}$ is a *pruning test* if $\mathcal{T}(t) = $ False implies that $\phi(t) \cap G(\mathcal{P}) = \emptyset$.

So at node t in the search, if $\mathcal{T}(t) = $ False, we backtrack. If $\mathcal{T}(t)$ returns True, no deductions about $\phi(t) \cap G(\mathcal{P})$ can be made, so we continue the search. Here we would like to stress that $\mathcal{T}(t) = $ True does not imply that $\phi(t) \cap G(\mathcal{P}) \neq \emptyset$.

The pruning tests are what make backtrack search a practical solution to problems with no known polynomial time algorithms. A good pruning test should return FALSE as much as possible. In particular, the shallower t is, the bigger the subtree we can skip over if $\mathscr{T}(t) =$ FALSE, and hence the more time we can save. Since we are computing $\mathscr{T}(t)$ at every node t that we visit, the function should also be quick. In particular, it should be computable in polynomial time.

If $\phi(t) \cap G(\mathcal{P}) \neq \emptyset$, we may still use an approximation of $\phi(t) \cap G(\mathcal{P})$ for search space reduction. Hence we are also interested in another type of function which we will call *refiners*.

Definition 2.3.8. A function $\mathscr{R} : T \to \wp(G)$ is a *refiner* if $\phi(t) \cap G(\mathcal{P}) \subseteq \mathscr{R}(t)$.

Note that the image $\mathscr{R}(t)$ of a refiner $\mathscr{R}$ at the root node $t = [\,]$ gives a subset of G containing $\phi(t) \cap G(\mathcal{P}) = G(\mathcal{P})$. Hence if $\mathscr{R}(t)$ is a group, we may search in a smaller group $\mathscr{R}(t)$ instead of G. Note also that for a refiner function $\mathscr{R}$, the boolean function $f : T \to Bool$ defined by

$$f(t) = \text{FALSE if and only if } \mathscr{R}(t) = \emptyset$$

is a pruning test.

Refiners are useful, as even when $\mathscr{T}(t) =$ TRUE, we may get a smaller approximation of $\phi(t) \cap G(\mathcal{P})$, hence reducing the search space. For more information on how such approximations $\mathscr{R}(t)$ can be represented, see Sections 2.3.1 and 2.3.2.

Lastly, note that we need not fix a base B of G prior to the search. As we always begin the search by iteratively fixing more base points of G, we may choose new base points based on the anticipated deductive power of such a base point.

2.3.1 Connections to constraint programming

A constraint satisfaction problem is a type of problem which has the following as inputs

1. a set of decision *variables* $x_1, x_2, \ldots, x_m$,
2. a set *domain* D_i for each variable x_i, and
3. a set of *constraints* on the variables,

and outputs an *assignment* or all assignments of the variables using the domains that satisfies all of the given constraints, or show that no such assignment exists.

The decision variables represent the choices we can make. The domain D_i of a variable x_i is the set of values the variable is allowed to take. Each decision variable x_i is *assigned* a value from the domain D_i. Each constraint specifies which assignments of the decision variables are allowed. For more examples of constraints, see [Dem].

A constraint satisfaction problem can be solved by backtrack search. The backtrack search for constraint satisfaction problems works similarly to that for groups. We systematically and iteratively make some assignments of variables, backtracking when we deduce that the constraints cannot be satisfied under the current variable assignment. If the solutions of the problem form a group, we may use methods in [PJ08] to obtain a generating set for the solutions.

By using the constraints, we can sometimes deduce that the current assignment of variables will cause some other assignments of variables to be non-viable. If we deduce this, we can then remove some values from the domains of some unassigned variables, and hence reduce the search tree. This process of reducing the domains is called *pruning*. In turn, together with the constraints, pruning may trigger more deductions. This is called constraint *propagation* [Bes06]. Good constraint propagators are central in having efficient solutions, and much effort has been made in giving powerful propagators for a large variety of constraints. Note that at any stage of the search, the domains give an approximation to our solution set under the subtree rooted at the current node.

Now we return to the problem in Section 2.3. That is, given a group $G \leq S_n$ and a property $\mathcal{P}$, we want to compute the subgroup $G(\mathcal{P})$ of G consisting of all elements $g \in G$ satisfying $\mathcal{P}$. Then computing $G(\mathcal{P})$ is a constraint satisfaction problem in the following way. Let $G \leq S_n$. Let $B = (\beta_1, \beta_2, \ldots, \beta_m)$ be a base of G. Since elements of G are fully identified from their base images, a solution in $G(\mathcal{P})$ is equivalent to an assignment of the images of $\beta_1, \beta_2, \ldots, \beta_m$. We set the decision variables be $x_1, x_2, \ldots, x_m$, each with domain $\{1, 2, \ldots, n\}$. The constraints are as follows.

1. For all $1 \leq i \leq m$, let α_i be the value assigned to x_i. Then $(\alpha_1, \alpha_2, \ldots, \alpha_m)$ is a valid base image of G.

2. The element $g \in G$ such that $\beta_i^g = \alpha_i$ for all $1 \leq i \leq m$ (if it exists) has to have property $\mathcal{P}$.

We want to find all assignments of the variables which satisfy the constraints.

For a tuple to be a valid base image of G, the entries of the tuple must be pairwise distinct. Hence the variables $x_1, x_2, \ldots, x_m$ must have pairwise distinct assignments. This is what we call an AllDifferent constraint, and can give strong deductive power. For more information on the propagation of such a constraint, we refer to [GMN08]. However, since we are working with groups, the group structure can be exploited to give us even stronger deduction power. These are the aforementioned pruning tests and refiners (Definitions 2.3.7 and 2.3.8). In this thesis, we will focus on using the group structure to prune the search tree. However, note that integrating better propagators into backtrack search for groups has the potential to speed up calculations.

2.3.2 Non-traditional backtrack search in groups

In [MP14], McKay gives a practical solution to the graph isomorphism problem using ordered partitions. In [Leo91], Leon introduced similar techniques for solving the permutation group problems in Section 2.2. This method by Leon is called *partition backtrack* and it remains as the state-of-the-art method to solve these problems in general. In partition backtrack, the domains are represented as a pair of ordered partitions over Ω, which gives efficient implementation methods. We will not give the implementation details here.

As before, let $G \leq \mathrm{Sym}(\Omega)$ and suppose that we want to find the subgroup $G(\mathcal{P})$ of G containing all elements which satisfy a given property $\mathcal{P}$. The partition backtrack method represents the domains using pairs of ordered partitions (P, Q) of Ω, where the sizes of the i-th cell of P and the i-th cell of Q are the same. The algorithm starts with P and Q being the trivial partitions of Ω with one cell. As in traditional backtracking, at each search node $t \coloneqq [\alpha_1, \alpha_2, \dots, \alpha_d]$, we have fixed some images of the base points $\beta_1, \beta_2, \dots, \beta_d$ to be $\alpha_1, \alpha_2, \dots, \alpha_d$ respectively. Let (P, Q) be the current pair of ordered partitions of Ω. Then for $1 \leq i \leq d$, the pair β_i and α_i are correlated singleton cells of P and Q respectively. Let $\phi(t)$ be as in Definition 2.3.3. The pair (P, Q) of ordered partitions give an approximation $A(P, Q) = \{g \in G \mid P^g = Q\}$ of $\phi(t) \cap G(\mathcal{P})$. If we find that $A(P, Q) = \emptyset$, we backtrack. If $|A(P, Q)| = 1$ and the element $g \in A(P, Q)$ satisfies property $\mathcal{P}$, update S as $\langle S, g \rangle$. If $|A(P, Q)| > 1$, we continue the search.

An important part of partition backtrack is the refiner process. At each node t, we use *partition refiners* to get partitions P' and Q', each finer than P and Q respectively, to obtain a tighter approximation of $\phi(t) \cap G(\mathcal{P})$.

A set V together with a set E of tuples in $V \times V$ is a *directed graph*. The induced action of $H \leq \mathrm{Sym}(\Omega)$ on $\Omega \times \Omega$ gives rise to certain directed graphs called the *orbital graphs*. More specifically, an *orbital graph* of H is a directed graph with vertices Ω and edge set an orbit of the action of H on $\Omega \times \Omega$. Then the normaliser $N_{\mathrm{Sym}(\Omega)}(H)$ permutes the orbital graphs of H. Theißen in his thesis gives refiners for the normaliser problem using orbital graphs [The97]. Then in [JPW19], Jefferson et. al. extend the use of orbital graphs as refiners to other permutation group problems.

An advantage of partition backtrack over the traditional backtrack is its ability to choose its next base point on the left-most branch. Consider the left-most node $t = [\beta_1, \beta_2, \dots, \beta_d]$ of depth d. Then $\phi(t)$ stabilises a partition P where each β_i is in a singleton cells. Refinement of the partition may fix more points, resulting in more singleton cells. So we choose our next base points to be some points not in singleton cells. Eventually this gives the base of $G(\mathcal{P})$, which is also known as the R-base.

Another different paradigm for backtrack search in groups is the graph backtrack [JPWW19]. The main idea is that rather than translating the orbital graphs into a pair of ordered partitions, we keep the graph and use its structure to refine our assignments in search. So the approximation of $\phi(t) \cap G(\mathcal{P})$ is an approximation of the automorphism group of some graph. This gives a closer approximation of $\phi(t) \cap G(\mathcal{P})$ and is shown to be able to greatly reduce the size of the search tree.

2.4 The normaliser problem

Recall Problem 2.2.10, which is also called the normaliser problem. In Section 2.4.1, we will list some complexity results for the normaliser problem for some restricted classes of H and G. Then in Section 2.4.2, we present some existing techniques for computing $N_G(H)$ in practice.

2.4.1 Complexity of the normaliser problem

Let $H, G \leq S_n$. Recall from Theorem 2.2.14 that $N_{S_n}(H)$ can be computed in simply exponential time. Then by Corollary 2.2.9, we can also compute $N_G(H) = G \cap N_{S_n}(H)$ in simply exponential time. We start by presenting some polynomial time normaliser results by restricting the class of G. Recall that all groups are given by their generating sets.

Theorem 2.4.1 ([KL90, LRW94])**.** *Let $H \leq G \leq S_n$. In polynomial time, we can test if G is nilpotent, and if so compute $N_G(H)$.*

Theorem 2.4.2 ([LM11])**.** *Let $d \in \mathbb{Z}^+$. Let Γ_d be the class of finite groups, all of whose non-abelian composition factors embed into S_d. In particular, this includes all solvable groups. Then, given $G, H \leq \mathrm{Sym}(\Omega)$ such that $G \in \Gamma_d$, the normaliser $N_G(H)$ can be computed in polynomial time.*

Theorem 2.4.3 ([Kan90])**.** *Given $K \lhd G \leq S_n$ and a Sylow p-subgroup P of K for a prime p, then $N_G(P)$ can be computed in polynomial time.*

Now, consider instead the Norm-Sym problem of computing $N_{S_n}(H)$ of a given group $H \leq S_n$.

Theorem 2.4.4 ([LM11])**.** *Let $H \leq S_n$ be simple, then $N_{S_n}(H)$ can be computed in polynomial time.*

Theorem 2.4.5 ([RDS20])**.** *Let $H \leq S_n$ be almost simple, then $N_{S_n}(H)$ can be computed in polynomial time.*

Theorem 2.4.6 ([RDS20])**.** *Let $H \leq S_n$ be primitive, then $N_{S_n}(H)$ can be computed in time $2^{O(\log^3 n)}$.*

Theorem 2.4.7 ([Sic20])**.** *Let $H \leq S_n$ be a primitive group with non-regular socle, then $N_{S_n}(H)$ can be computed in polynomial time.*

The proof of Theorem 2.4.6 in [RDS20] uses the following result for transitive groups H. Here we note that it is also true for intransitive groups H.

Lemma 2.4.8. *Let B be a base of $H = \langle X \rangle \leq S_n$. Then $N_{S_n}(H)$ can be computed in $2^{O(|X||B| \log n)}$.*

Proof. Let $X = \{x_1, x_2, \dots, x_d\}$ and let $|B| = m$. We will consider all possible $A \in \{1, 2, \dots, n\}^m$ and all possible $I_i \in \{1, 2, \dots, n\}^m$ for $1 \leq i \leq d$. For each such tuple $t := (A, I_1, I_2, \dots, I_d)$, if A is a base of H and each I_i is a valid base image of A, we attempt to define a map $\phi_t : H \to H$ by: for $x_i \in X$, let the image $\phi_t(x_i)$ be the element of H mapping A to I_i. If $\phi_t(x_i)$ is an automorphism and is induced by conjugation in S_n, using Lemma 2.1.19, we find a witness $g \in S_n$. We gather all such witnesses in a group G.
We claim that $N_{S_n}(H) = \langle C_{S_n}(H), G\rangle$. The backward inclusion is clear. For the forward inclusion, let $\nu \in N_{S_n}(H)$. Let $A := B^\nu$ and $I_i := (B^{x_i})^\nu$ for $1 \leq i \leq d$. Note first A is a base of H since $H_{(A)} = H_{(B^\nu)} = (H_{(B)})^\nu = 1$. Secondly, note that $A^{x_i^\nu} = B^{\nu\nu^{-1}x_i\nu} = B^{x_i\nu} = I_i$ for all $1 \leq i \leq d$, so the I_i are valid base images of A. Hence, by letting $t := (A, I_1, I_2, \dots, I_d)$, the map ϕ_t is defined. Since the base images of each element of H are unique and both $\phi_t(x_i)$ and x_i^ν map A to I_i, we have $\phi_t(x_i) = x_i^\nu$ for $1 \leq i \leq d$. In particular, ϕ_t is an automorphism induced by conjugation in S_n and so there exists $g \in G$ such that $\phi_t(x_i) = x_i^g$ for all $1 \leq i \leq d$. Therefore, for all $1 \leq i \leq d$, we have $x_i^g = x_i^\nu$. Hence $\nu g^{-1} \in C_{S_n}(H)$.
For the complexity result, note first that there are $O(n^{(d+1)|B|}) = O(n^{|X||B|})$ choices of $t = (A, I_1, I_2, \dots, I_d)$. By Lemma 2.1.6, we can check if the I_i are valid base images of A in polynomial time. By Part 6 of Theorem 2.1.9, we can check if ϕ_t induces an automorphism in polynomial time. Then by Lemma 2.1.19, in polynomial time, we can decide if ϕ_t is induced by conjugation in S_n and if so output a witness g. □

So if a group $G \leq S_n$ has base B and generating set X where $|B|$ and X are of size $O(\log n)$, then $N_{S_n}(H)$ can be computed in time $2^{O(\log^3 n)}$.

2.4.2 Computing normalisers in practice

In general, we compute normalisers via backtrack search (see Section 2.3). In this subsection, we will first present some elementary results used for pruning tests and refiners (Definitions 2.3.7 and 2.3.8) for the normaliser problem.

For the rest of the subsection, let $H, G \leq S_n$ and suppose that we want to compute $N_G(H)$. Let $\Omega = \{1, 2, \dots, n\}$. Let $B = (\beta_1, \beta_2, \dots, \beta_m)$ be a base of G and let T be the search tree of G with respect to the base B. Let $\mathcal{P}$ be the property "conjugates H to H". So $G(\mathcal{P}) = \{g \in G \mid H^g = H\} = N_G(H)$.

The following result is frequently used for pruning.

Lemma 2.4.9. *Let $g \in N_G(H)$ such that $(\delta_1, \delta_2, \dots, \delta_k)^g = (\gamma_1, \gamma_2, \dots, \gamma_k)$ for some k. Then $(H_{(\delta_1,\delta_2,\dots,\delta_k)})^g = H_{(\gamma_1,\gamma_2,\dots,\gamma_k)}$.*

Proof. Let $h \in H_{(\delta_1,\delta_2,\dots,\delta_k)}$. Then h^g fixes the γ_i, so $h^g \in H_{(\gamma_1,\gamma_2,\dots,\gamma_k)}$. The reverse inclusion follows from symmetry. □

Suppose that we are at node $t = [\alpha_1, \alpha_2, \dots, \alpha_d]$ in the search three T. Then we wish to backtrack when $H_{(\alpha_1,\alpha_2,\dots,\alpha_d)}$ and $H_{(\beta_1,\beta_2,\dots,\beta_d)}$ are not conjugate in G. Hence we want our pruning tests and refiners to return FALSE and $\emptyset$ respectively if $H_{(\alpha_1,\alpha_2,\ldots,\alpha_d)}$ and $H_{(\beta_1,\beta_2,\ldots,\beta_d)}$ are not conjugate in G. However, deciding if two groups are conjugate in G is difficult, even when G is the symmetric group. So instead, we identify and compute properties of groups that are preserved under conjugation, and use them to construct pruning tests and refiners.

Definition 2.4.10. Denote by $\mathcal{S}(S_n)$ the set of all subgroups of S_n. A *conjugacy invariant function* Ψ is a function with domain $\mathcal{S}(S_n)$ such that if $K, L \leq S_n$ are conjugate in S_n, then $\Psi(K) = \Psi(L)$.

Example 2.4.11. Let Ψ_1 and Ψ_2 be functions with domain $\mathcal{S}(S_n)$ defined by

1. $\Psi_1(K) = |K|$.

2. $\Psi_2(K)$ gives the multiset consisting of the sizes of the orbits of K.

Then since conjugation preserves size and orbit sizes, Ψ_1 and Ψ_2 are conjugacy invariant functions.

We can use a conjugacy invariant function to prune our search tree.

Lemma 2.4.12. *Let Ψ be a conjugacy invariant function. Define a function $\mathcal{T} : T \to$ Bool by*

$$\mathcal{T}([\alpha_1,\alpha_2,\ldots,\alpha_d]) = \begin{cases} \text{TRUE} & \textit{if } \Psi(H_{(\beta_1,\beta_2,\ldots,\beta_d)}) = \Psi(H_{(\alpha_1,\alpha_2,\ldots,\alpha_d)}), \\ \text{FALSE} & \textit{otherwise.} \end{cases}$$

Then $\mathcal{T}$ is a pruning test for computing the normaliser $N_G(H)$.

Proof. Let $t := [\alpha_1,\alpha_2,\ldots,\alpha_d] \in T$ such that $\mathcal{T}(t) =$ FALSE. Then $\Psi(H_{(\beta_1,\beta_2,\ldots,\beta_i)}) \neq \Psi(H_{(\alpha_1,\alpha_2,\ldots,\alpha_i)})$. By the definition of a conjugacy invariant function, $H_{(\beta_1,\beta_2,\ldots,\beta_i)}$ and $H_{(\alpha_1,\alpha_2,\ldots,\alpha_i)}$ are not conjugate in S_n. Then by Lemma 2.4.9, there does not exist $\nu \in N_G(H) \leq N_{S_n}(H)$ such that $(\beta_1,\beta_2,\ldots,\beta_d)^\nu = (\alpha_1,\alpha_2,\ldots,\alpha_d)$. In other words, $N_G(H) \cap \phi(t) = \emptyset$. □

Similarly, we define conjugacy invariant refiners which will give refiners for the normaliser problem.

Definition 2.4.13. Denote by $\mathcal{S}(S_n)$ the set of all subgroups of S_n. A *conjugacy invariant refiner* Φ is a function with domain $\mathcal{S}(S_n) \times \{1, 2, \ldots, n\}$ such that for subgroups $K, L \leq S_n$ for which there exists $g \in S_n$ such that $K^g = L$, we have $\Phi(K, \alpha) = \Phi(L, \alpha^g)$ for all $\alpha \in \Omega$.

Example 2.4.14. Let Φ be a function with domain $\mathcal{S}(S_n) \times \{1, 2, \ldots, n\}$ such that $\Phi(K, \alpha)$ returns the size of the K-orbit containing α. Then since g maps a K-orbit to a K^g-orbit of the same size, Φ is a conjugacy invariant refiner.

Lemma 2.4.15. *Let Φ be a conjugacy invariant refiner. Define a function $\mathscr{R} : T \to \wp(G)$ by*

$$\mathscr{R}([\alpha_1,\alpha_2,\ldots,\alpha_d]) = \{g \in G \mid \Phi(H_{(\beta_1,\ldots,\beta_d)},\gamma) = \Phi(H_{(\alpha_1,\ldots,\alpha_d)},\gamma^g) \text{ for all } 1 \leq \gamma \leq n\}$$

Then $\mathscr{R}$ is a refiner for computing the normaliser $N_G(H)$.

Proof. Let $t = [\alpha_1,\alpha_2,\ldots,\alpha_d] \in T$. To show that $\phi(t) \cap N_G(H) \subseteq \mathscr{R}(t)$, let $g \notin \mathscr{R}(t)$. We show that $g \notin \phi(t) \cap N_{S_n}(H)$. If $g \notin \phi(t)$ then we are done. Suppose that $g \in \phi(t)$. Since $g \notin \mathscr{R}(t)$, there exists $\gamma \in \Omega$ such that $\Phi(H_{(\beta_1,\beta_2,\ldots,\beta_i)},\gamma) \neq \Phi(H_{(\alpha_1,\alpha_2,\ldots,\alpha_i)},\gamma^g)$. Then by the definition of a conjugacy invariant refiner, $(H_{(\beta_1,\beta_2,\ldots,\beta_i)})^g \neq H_{(\alpha_1,\alpha_2,\ldots,\alpha_i)}$. So by Lemma 2.4.9, $g \notin N_{S_n}(H)$. □

Butler in [But83] used the orbit structure of groups as a conjugacy invariant function (see Example 2.4.11), and Theißen in [The97] uses the orbital graphs as conjugacy invariant refiners. We will present other conjugacy invariant functions and refiners in Chapter 4.

In addition to Lemma 2.4.9, the following lemma can be used in conjunction with the conjugacy invariant functions and refiners to prune and/or refine the search tree:

Lemma 2.4.16. *Let Δ_1 and Δ_2 be unions of H-orbits. Let $g \in N_G(H)$ such that $\Delta_1^g = \Delta_2$. Then $(H|_{\Delta_1})^g = H|_{\Delta_2}$.*

Proof. Since the support of $(H|_{\Delta_1})^g$ is Δ_2 and $H^g = H$, we have $(H|_{\Delta_1})^g \leq H|_{\Delta_2}$. The reverse inclusion follows from symmetry. □

Suppose that we are at a node of the search tree labeled by $t = [\alpha_1,\alpha_2,\ldots,\alpha_d]$. Let Δ be the union of H-orbits fixed pointwise by $H_{(\beta_1,\beta_2,\ldots,\beta_d)}$. Then $(\beta_1,\beta_2,\ldots,\beta_d)$ forms a base for $H|_\Delta$, and so $(\alpha_1,\alpha_2,\ldots,\alpha_d)$ forms a base for $H|_{\Delta'}$ for some union Δ' of H-orbits. Then we may use conjugacy invariant functions and refiners to prune and refine the search tree as before.

We end the section with some existing techniques for improving the computation of $N_G(H)$. Holt in [Hol91] introduced two pruning tests. Firstly, we can prune using the orbit sizes of the stabilisers of the partial base points and base images. This is equivalent to the combination of Example 2.4.14 and Lemma 2.4.15. Secondly, he observed that the normalising elements induce an automorphism of H. In particular, if H is regular, choosing the base image of two points fully determines the image of each element of H, as in Lemma 2.4.8. Also if H acts faithfully on a set Δ, then once the image of Δ is determined, we have obtained an automorphism of H, defined on the generators. This helps with the pruning of the search tree.

Holt's result works well for regular groups, but not on elementary abelian groups that are neither regular nor the intransitive direct product on disjoint sets. Hulpke in [Hul08] tackles this situation by also using the fact that each normalising element induces an automorphism of H. Let A be the subgroup of $\mathrm{Aut}(H)$ induced by the conjugation action of $N_G(H)$. Let C be the maximal subgroup of $\mathrm{Aut}(H)$ which preserves equivalence classes induced by the cycle structures of elements of H. Since conjugation preserves cycle structures, we have $A \leq C$. By considering C as a permutation group in $\mathrm{Sym}(H)$, we perform a backtrack search in C to find A and hence compute $N_G(H)$.

Let $H \leq S_n$ be intransitive and suppose that we are computing $N_{S_n}(H)$. In [Hul05, Section 11], Hulpke present methods of obtaining a proper subgroup S of S_n containing $N_{S_n}(H)$ by considering the permutation isomorphism classes of the projections of H onto the H-orbits to construct conjugacy invariant functions. Much of the ideas presented in [Hul05, Section 11] will be restated in Section 4.3. Lastly note that for the case when H is transitive but not primitive, Hulpke observes that similar methods can be used by considering the H-blocks instead of the H-orbits.

Lastly, Miyamoto in [Miy06] uses association schemes to improve normaliser computations. Let $H \leq S_n$ be transitive and let A be the automorphism group of the orbitals of H. Since $N_{S_n}(H) \leq A$, a block Δ of A is also a block of H and of $N_{S_n}(H)$. Let $\overline{H}$ be the action of H on the blocks. Then we have $N_{S_n}(H) \leq (N_{A_{\{\Delta\}}|_\Delta}(H_{\{\Delta\}}|_\Delta) \wr N_{\overline{A}}(\overline{H}))$, so we may search in a smaller group instead.

Chapter 3

Disjoint Direct Product Decomposition

A *direct product decomposition* of a given group H is an expression of H as a direct product of groups. The direct product decomposition is useful for understanding the structure of the group, and to solve problems more efficiently. Hence, it is important to find an efficient algorithm for computing such decompositions. Kayal and Nezhmetdinovm, in [KN09], give a polynomial time algorithm for computing a direct product decomposition of a group H given by its multiplication table, which has input size $|H|^2$. We consider computing direct product decompositions of permutation groups given by generating sets, which are usually much smaller than the order of the group. Wilson, in [Wil12] gives a polynomial time solution to such a problem. However, as far as we know, this algorithm has not yet been implemented.

In this chapter, we will consider a particular type of direct product decomposition for finite permutation groups, which we will call a *disjoint direct product decomposition*, which are direct product decompositions of permutation groups where the factors have disjoint supports. We will give a polynomial time algorithm for finding disjoint direct product decompositions of a permutation group given by a generating set and also demonstrate the practical efficiency of our algorithm. In this chapter, we will regard the direct product of groups with disjoint supports as a subgroup of the symmetric group on the union of the supports of the direct factors.

Definition 3.0.1 (Disjoint direct product decomposition)**.** Let $H \leq S_n$. We say that $H = H_1 \times H_2 \times \ldots \times H_r$ is a *disjoint direct product decomposition* of H if it is a direct product decomposition of H and the groups H_i have pairwise disjoint supports.
If there exists a disjoint direct product decomposition $H = H_1 \times H_2 \times \ldots \times H_r$ of H with $r > 1$, then we say that H is *d.d.p. decomposable*, otherwise we say that H is *d.d.p. indecomposable.*
A disjoint direct product decomposition is *finest* if each factor is d.d.p. indecomposable.

Since the disjoint direct product decomposition is more restrictive than a more general decomposition, it can be computed much faster and has many useful applications. As we will demonstrate in Section 3.4, the disjoint direct product decomposition of a permutation group can be used to greatly speed up various other calculations with permutation groups. Furthermore, calculations that previously could not be completed in a reasonable time frame can be solved very quickly using the disjoint direct product decomposition to subdivide the computation into smaller pieces.

Another important application of disjoint direct product decompositions lies in other areas of computer science, where groups arise from symmetries of combinatorial objects. To reduce the computation time, groups are used to eliminate the symmetries of the objects through a process called *symmetry breaking* [GPP06].

Donaldson et. al. [DM06] use the disjoint direct product decomposition to improve the performance of detecting symmetric states in model checking and Grayland et. al. [GJMRD09, Theorem 10] uses the disjoint direct product decomposition when generating symmetry breaking constraints for symmetric problems. Grayland et. al. gives an algorithm for symmetry breaking which uses the disjoint direct product of two symmetry groups [GJMRD09] but otherwise does not consider general direct product decompositions. In both of these applications, disjoint direct product decompositions lead to significant speed-ups.

For these applications in both computational group theory and otherwise, the time saved depends on the number of factors in the decomposition. Hence we are interested in an algorithm that always computes a *finest* disjoint direct product decomposition. By the Krull–Schmidt theorem, any finite group has a unique finest direct product decomposition [Hun74, Theorem 3.8], up to isomorphism. In Proposition 3.2.2, we show that the finest disjoint direct product decomposition of a given finite permutation group is unique.

The main result of this chapter is to provide an efficient algorithm to compute the finest disjoint direct product decomposition of a given permutation group, and hence prove the following.

Theorem 3.0.2. *Let $H \leq S_n$ be given by a generating set X. Then the finest disjoint direct product decomposition of H can be computed in time polynomial in $|X|n$.*

Our algorithm behind Theorem 3.0.2 manipulates a strong generating set and therefore is fast in practice once a base and strong generating set have been found. Finding a base and strong generating set is an initial part of most permutation group algorithms. Hence, finding a disjoint direct product decomposition will not add significantly to the runtime of these algorithms.

The structure of this chapter is as follows. In Section 3.1, we present some related work in the literature, and the definitions, notation and background knowledge we use later on. In Section 3.2, we present the theoretical framework which we use for the algorithms we present in Section 3.3. Also in Section 3.3, we prove Theorem 3.0.2. Lastly, in Section 3.4, we demonstrate how the algorithm can be used to speed up computation in some permutation group theoretic functions in GAP.

3.1 Background and preliminaries

If G is a direct factor of H, then $G \trianglelefteq H$. So a naive approach to finding its disjoint direct product decomposition is to consider all normal subgroups N of H, check if there exists K such that $N \times K = H$, then recursively try to decompose N and K. While it is possible to optimise this approach, it has worst-case exponential complexity, since it requires considering all normal subgroups of H, and there can be exponentially many of them.

Wilson's polynomial time algorithm in [Wil12] computes a finest (not necessarily disjoint) direct product decomposition of a given permutation group H. As far as we are aware, the algorithm has yet to be implemented. We show that it is substantially easier to compute the finest disjoint direct product decomposition than the finest direct product decomposition.

Donaldson and Miller in [DM09, Section 5.1] present a polynomial algorithm for computing a disjoint direct product decomposition by considering only the generators. They use the observation that, if $H = \langle X \rangle$ and there exists $S \subset X$ such that the support of S and the support of $X \backslash S$ are disjoint, then $H = \langle S \rangle \times \langle X \backslash S \rangle$ is a disjoint direct product decomposition. The method by Donaldson and Miller is a subprocedure of the algorithm we present in Section 3.3. However, note that the method in [DM09] does not guarantee that the decomposition is a finest one as different choices of X may produce different decompositions. Donaldson and Miller reported that using the generators computed from the graph automorphism program they used, this method seems to almost always produce the finest decomposition. We hypothesise that these programs almost always produce separable strong generating sets, which we shall define in Definition 3.2.8.

In [DM09], Donaldson and Miller also present an exponential-time algorithm to compute the finest disjoint direct product decomposition of H. The algorithm involves recursively computing disjoint direct product decompositions with two factors. To construct such a decomposition of H, they consider all partitions of the H-orbits with two cells C_1 and C_2. Letting $\Delta_1 \coloneqq \bigcup_{\Omega_i \in C_1} \Omega_i$ and $\Delta_2 \coloneqq \bigcup_{\Omega_i \in C_2} \Omega_i$, they test if P gives rise to a disjoint direct product decomposition by checking if $H = H|_{\Delta_1} \times H|_{\Delta_2}$. They also made a significant improvement to their algorithm by first considering the projections onto pairs of orbits and deciding if they are d.d.p. decomposable.

We would like to draw the reader's attention to an elementary result, which we shall repeatedly use later.

Lemma 3.1.1. *Let H be a subdirect product of the external direct product $G_1 \times G_2$. Then $H = G_1 \times G_2$ if and only if $1 \times G_2 \leq H$.*

Proof. The forward implication is clear. For the backward implication, for all $(g_1, g_2) \in$

H, since $(1, g_2) \in H$, we have $(g_1, 1) \in H$. The projection of H onto G_1 is the whole G_1, so $G_1 \times 1 \leq H$. Now since $G_1 \times 1$ and $1 \times G_2$ generate $G_1 \times G_2$ and are both contained in H, we have $H = G_1 \times G_2$. □

Recall that we regard a direct product of groups that have disjoint supports as a subgroup of the symmetric group over the disjoint union of the supports of the factors. For the rest of the chapter, we will use the following notation.

Notation 3.1.2. Let $H \leq S_n$. Let $\Omega_1, \Omega_2, \ldots, \Omega_k$ be the H-orbits. For $1 \leq i \leq k$, let $G_i := H|_{\Omega_i}$. We consider H as a subdirect product of $G := G_1 \times G_2 \times \ldots \times G_k \leq S_n$.
For $1 \leq i \leq k$, let $\pi_i : G \to G_i$ be defined by $h \mapsto h|_{\Omega_i}$. For a subset I of $\{1, 2, \ldots, k\}$, let $\Pi_I : G \to H|_{\cup_{i \in I} \Omega_i}$ be defined by $h \mapsto h|_{\cup_{i \in I} \Omega_i}$.
For all $1 \leq i \leq k$, let $\Delta_i := \bigcup\limits_{j \leq i} \Omega_j$.
Let $\bar{i}$ denote the set $\{1, 2, \ldots, i\}$.

It is important that the naming of the H-orbits are fixed in our algorithms. We maintain such consistency by fixing an order on the H-orbits. The choice of how we order them is unimportant. In our experiments, we chose to order the orbits by their smallest elements.

3.2 Disjoint direct product decomposition

In this section, we will first show that $H \leq S_n$ has a unique finest disjoint direct product decomposition. For $1 \leq i \leq k-1$, let N_{i+1} and θ_i be as in Theorem 1.4.3. In Section 3.2.1, we will see how the computation of disjoint direct product decompositions can be reduced to computing the N_{i+1} and the kernels of the θ_i. We then show that the N_{i+1} and $\mathrm{Ker}(\theta_i)$ can be efficiently computed in Section 3.2.2.

Definition 3.2.1. We say that two disjoint direct product decompositions $H = H_1 \times H_2 \times \ldots \times H_r$ and $H = K_1 \times K_2 \times \ldots \times K_s$ of $H \leq S_n$ are *equivalent* if the sets of sets $\{Supp(H_i) \mid 1 \leq i \leq r\}$ and $\{Supp(K_i) \mid 1 \leq i \leq s\}$ are the same.

Proposition 3.2.2. *Up to equivalence, there is a unique finest disjoint direct product decomposition of H.*

Proof. Let $H = H_1 \times H_2 \times \ldots \times H_r$ and $H = K_1 \times K_2 \times \ldots \times K_s$ be two inequivalent finest disjoint direct product decompositions of H. Since $\{Supp(H_i) \mid 1 \leq i \leq r\}$ and $\{Supp(K_i) \mid 1 \leq i \leq s\}$ forms partitions of Ω, for each H_i, there exists at least one K_j such that $Supp(H_i) \cap Supp(K_j) \neq \emptyset$. Now since the two decompositions are inequivalent, there exist $1 \leq i \leq r$ and $1 \leq j \leq s$ such that $Supp(H_i) \neq Supp(K_j)$ and $Supp(H_i) \cap Supp(K_j) \neq \emptyset$. Let $\Gamma := Supp(H_i)$ and $\Delta := Supp(K_j)$. We will show that $H_i = H_i|_{\Gamma \setminus \Delta} \times H_i|_{\Gamma \cap \Delta}$ is a disjoint direct product decomposition of H_i, which contradicts the fact that $H = H_1 \times H_2 \times \ldots \times H_r$ is a finest decomposition. Since H_i is a subdirect product of $H_i|_{\Gamma \setminus \Delta} \times H_i|_{\Gamma \cap \Delta}$, by the backward implication of Lemma 3.1.1

it suffices to show that $H_i|_{\Gamma\cap\Delta} \times 1_{\Gamma\setminus\Delta} \leq H_i$. We do so by showing that for all $h_i \in H_i$, there exists $h_i' \in H_i$ such that $h_i'|_{\Gamma\cap\Delta} = h_i|_{\Gamma\cap\Delta}$ and $h_i'|_{\Gamma\setminus\Delta} = 1$.
Let $h_i \in H_i$. Let $\hat{h_i} \in S_n$ satisfy $\hat{h_i}|_\Gamma = h_i$ and $\hat{h_i}|_{\overline{n}\setminus\Gamma} = 1$. By the forward implication of Lemma 3.1.1, $\hat{h_i} \in H$. Similarly, since K_j is a disjoint direct factor of H, there exists $h \in H$ such that $h|_{\overline{n}\setminus\Delta} = \hat{h_i}|_{\overline{n}\setminus\Delta}$ and $h|_\Delta = 1$. Then $h' := \hat{h_i}h^{-1}$ is an element of H such that

$$h'|_{\Gamma\cap\Delta} = (\hat{h_i}|_{\Gamma\cap\Delta})(h^{-1}|_{\Gamma\cap\Delta}) = \hat{h_i}|_{\Gamma\cap\Delta} = h_i|_{\Gamma\cap\Delta},$$

$$h'|_{\Gamma\setminus\Delta} = (\hat{h_i}|_{\Gamma\setminus\Delta})(h^{-1}|_{\Gamma\setminus\Delta}) = (\hat{h_i}|_{\Gamma\setminus\Delta})(\hat{h_i}^{-1}|_{\Gamma\setminus\Delta}) = 1,$$

and $h'|_{\overline{n}\setminus\Gamma} = 1$. Therefore, $h'|_\Gamma$ is an element of H_i such that $h'|_{\Gamma\cap\Delta} = h_i|_{\Gamma\cap\Delta}$ and $h'|_{\Gamma\setminus\Delta} = 1$. □

3.2.1 Computing the disjoint direct product decomposition

Recall Notation 3.1.2 and recall that we denote $\{1, 2, \ldots, i\}$ by $\overline{i}$. We will compute the finest disjoint direct product decomposition of H by iteratively computing the finest disjoint direct product decomposition of $\Pi_{\overline{i}}(H)$ for $1 \leq i \leq k$. In this subsection, we show for $1 \leq i < k$, how we can compute the finest disjoint direct product decomposition of $\Pi_{\overline{i+1}}(H)$ using the finest disjoint direct product decomposition of $\Pi_{\overline{i}}(H)$ and the groups N_{i+1} and homomorphisms θ_i from Theorem 1.4.3.

Since the support of each disjoint direct factor of a group is a union of (some of) its orbits, we will be computing certain partitions of $\overline{i}$ for each $1 \leq i \leq k$.

Definition 3.2.3. For $1 \leq i \leq k$, let $\mathcal{P}_i = \langle C_1 \mid C_2 \mid \ldots \mid C_r\rangle$ be the partition of $\overline{i}$ consisting of cells $C_j \subseteq \overline{i}$ for $1 \leq j \leq r$ such that $\Pi_{\overline{i}}(H) = \Pi_{C_1}(H) \times \Pi_{C_2}(H) \times \ldots \times \Pi_{C_r}(H)$ is the finest disjoint direct product decomposition of $\Pi_{\overline{i}}(H)$.

Observe that trivially, $\mathcal{P}_1 = \langle\{1\}\rangle$. Proposition 3.2.4 describes how we can compute $\mathcal{P}_{i+1}$ using $\mathcal{P}_i$ for $1 \leq i < k$. To simplify notation, from now on, for subsets $I, J \subseteq \overline{k}$ and for $h \in H$, we denote by $\Pi_I(h) \times 1_J$ the element $h' \in \Pi_{I\cup J}(H)$ such that $\Pi_I(h') = \Pi_I(h)$ and $\Pi_J(h') = 1$. Similarly, for $S \subseteq H$, we denote by $\Pi_I(S) \times 1_J$ the set $\{\Pi_I(h) \times 1_J \mid h \in S\}$.

Proposition 3.2.4. *Let $1 \leq i < k$. Let $\mathcal{P}_i = \langle C_1 \mid C_2 \mid \ldots \mid C_r\rangle$ be as in Definition 3.2.3 and let θ_i be as in Theorem 1.4.3. Let*

$$S := \{C_j \mid \Pi_{C_j}(H) \times 1_{\overline{i}\setminus C_j} \not\subseteq \mathrm{Ker}(\theta_i),\ \textit{for } 1 \leq j \leq r\}$$

and let $C = \bigcup_{C_j \in S} C_j \cup \{i+1\}$. Then

$$\Pi_{\overline{i+1}}(H) = \Pi_C(H) \times \bigtimes_{C_j \notin S} \Pi_{C_j}(H) \tag{3.1}$$

is the finest disjoint direct product decomposition of $\Pi_{\overline{i+1}}(H)$.

Hence the partition $\mathcal{P}_{i+1}$ from Definition 3.2.3 is the partition of $\overline{i+1}$ consisting of cell C and all other cells C_j of $\mathcal{P}_i$ such that $C_j \notin S$.

Proof. We will first show that Equation (3.1) is a disjoint direct product decomposition of $\Pi_{\overline{i+1}}(H)$, and then we will show that it is the *finest* disjoint direct product decomposition. The statement on $\mathcal{P}_{i+1}$ will then follow from Definition 3.2.3.
Since the factors in Equation (3.1) move disjoint sets of points, we show that Equation (3.1) is a disjoint direct product decomposition of $\Pi_{\overline{i+1}}(H)$, by showing that it is a direct product decomposition. Observe that $\Pi_{\overline{i+1}}(H)$ is a subdirect product of $\Pi_C(H) \times \bigtimes_{C_j \notin S} \Pi_{C_j}(H)$. Then by the backward implication of Lemma 3.1.1, it suffices to show that $1_C \times \bigtimes_{C_j \notin S} \Pi_{C_j}(H) \leq \Pi_{\overline{i+1}}(H)$. We will do so by showing that $\Pi_{C_j}(H) \times 1_{\overline{i+1} \setminus C_j} \leq \Pi_{\overline{i+1}}(H)$ for all cells C_j of $\mathcal{P}_i$ such that $C_j \notin S$.
Let $C_j \notin S$. Then $\theta_i(\Pi_{C_j}(H) \times 1_{\bar{i} \setminus C_j}) = N_{i+1}$. Take $R_{i+1} = \{1\}$ and let φ_i be as in Theorem 1.4.3. Then $\varphi_i(\Pi_{C_j}(H) \times 1_{\bar{i} \setminus C_j}) = \Pi_{C_j}(H) \times 1_{\overline{i+1} \setminus C_j} \leq \Pi_{\overline{i+1}}(H)$. Hence, Equation (3.1) gives a disjoint direct product decomposition of $\Pi_{\overline{i+1}}(H)$.

We will now show that Equation (3.1) is the *finest* disjoint direct product decomposition. As $C_j \notin S$ are cells of $\mathcal{P}_i$, the groups $\Pi_{C_j}(H)$ for $C_j \notin S$ are d.d.p. indecomposable, so it remains to show that $\Pi_C(H)$ is d.d.p. indecomposable. Observe that for $C_j \notin S$, since $\Pi_{C_j}(H)$ is a finest disjoint direct factor of $\Pi_{\overline{i+1}}(H)$, each $C_j \notin S$ is a cell of $\mathcal{P}_{i+1}$. We proceed as follows. We first show that C is a union of some of $U := \{C_1, C_2, \ldots, C_r, \{i+1\}\}$ and then show that $C_j \in S$ is in the same cell of $\mathcal{P}_{i+1}$ with $\{i+1\}$, from which we deduce that C is a cell of $\mathcal{P}_{i+1}$ and so $\Pi_C(H)$ is d.d.p. indecomposable.
To prove the first claim, we show that for all $u \in U$ such that $u \cap C \neq \emptyset$, we have $u \subseteq C$. This is trivially true for $u = \{i+1\}$. Let C_j be a cell of $\mathcal{P}_i$ such that $C_j \cap C \neq \emptyset$. We have shown that Equation (3.1) is a disjoint direct product decomposition, so the projection $\Pi_C(H)$ is a disjoint direct factor of $\Pi_{\overline{i+1}}(H)$, so $\Pi_C(H) \times 1_{\overline{i+1} \setminus C} \leq \Pi_{\overline{i+1}}(H)$. Then $\Pi_{C_j \cap C}(H) \times 1_{C_j \setminus C} \leq \Pi_{C_j}(H)$. Since $\Pi_{C_j}(H)$ is d.d.p. indecomposable, $C_j \cap C = C_j$, so $C_j \subseteq C$ and therefore C is a union of some of U.
Lastly, we show that each $C_j \in S$ is in the cell of $\mathcal{P}_{i+1}$ containing $i+1$. Let $C_j \in S$. Aiming for a contradiction, suppose that C_j and $i+1$ are in different cells of $\mathcal{P}_{i+1}$. Let φ_i be as in Theorem 1.4.3 and consider $L := \varphi_i(\Pi_{C_j}(H) \times 1_{\bar{i} \setminus C_j}) \subseteq \Pi_{\overline{i+1}}(H)$. Let D be a cell of $\mathcal{P}_{i+1}$ containing C_j. Then $\Pi_D(L) \times 1_{\overline{i+1} \setminus D} \leq \Pi_D(H) \times 1_{\overline{i+1} \setminus D}$ is contained in $\Pi_{\overline{i+1}}(H)$. Since $i+1 \notin D$, it follows that

$$\Pi_{\bar{i}}(L) \times 1_{i+1} = \Pi_D(L) \times 1_{\overline{i+1} \setminus D} \leq \Pi_{\overline{i+1}}(H).$$

Now since both L and $\Pi_{\bar{i}}(L) \times 1_{i+1}$ are contained in $\Pi_{\overline{i+1}}(H)$, the set $1_{\bar{i}} \times \pi_{i+1}(L)$ is also contained in $\Pi_{\overline{i+1}}(H)$. Therefore $\pi_{i+1}(L) \subseteq N_{i+1}$ and hence $\theta_i(\Pi_{C_j}(H) \times 1_{\bar{i} \setminus C_j}) = N_{i+1}$, a contradiction to the fact that $C_j \in S$. □

Proposition 3.2.4 will be used as the core of our algorithm for finding the finest disjoint direct factor decomposition in Section 3.3.

3.2.2 Orbit-ordered base and separable strong generating set

In this subsection, fix $1 \leq i < k$ and let N_{i+1}, θ_i and φ_i be as in Theorem 1.4.3. We will see how we can use some fundamental data structures associated to permutation groups to compute the N_{i+1} and find the cells C_j of $\mathcal{P}_i$ such that $\Pi_{C_j}(H) \times 1_{\bar{i} \setminus C_j} \not\subseteq \mathrm{Ker}(\theta_i)$.

Recall the definitions of base and strong generating sets from Definitions 1.5.1 and 1.5.3, and recall the sets Δ_i from Notation 3.1.2. We will be using bases that are orbit-ordered.

Definition 3.2.5. A base $B := (\beta_1, \beta_2, \ldots, \beta_m)$ of $H \leq S_n$ is *orbit-ordered* with respect to the ordering $\Omega_1, \Omega_2, \ldots, \Omega_k$ of the H-orbits if there exist $1 \leq j_1 \leq j_2 \leq \ldots \leq j_k \leq m$ such that for all $1 \leq i \leq k$, we have $H_{(\beta_1, \beta_2, \ldots, \beta_{j_i})} = H_{(\Delta_i)}$.

Remark 3.2.6. Recall from Notation 3.1.2 that we fixed an ordering Ω_1, Ω_2, $\ldots$, Ω_k of the H-orbits. Then the concatenation of $\Omega_1, \Omega_2, \ldots, \Omega_k$ is a (redundant) orbit-ordered base of H.

We can compute N_{i+1} from a strong generating set of H relative to an orbit-ordered base.

Lemma 3.2.7. *Let $B := (\beta_1, \beta_2, \ldots, \beta_m)$ be an orbit-ordered base of $H \leq S_n$ with respect to the ordering $\Omega_1, \Omega_2, \ldots, \Omega_k$. Let X be a strong generating set of H with respect to B. Then $N_{i+1} = \langle \pi_{i+1}(x) \mid x \in X \cap H_{(\Delta_i)} \rangle$ for all $1 \leq i < k$.*

Proof. Since B is orbit-ordered, there exists $1 \leq j \leq m$ such that $H_{(\beta_1, \beta_2, \ldots, \beta_j)} = H_{(\Delta_i)}$. As X is a strong generating set relative to B, we have $H_{(\Delta_i)} = \langle X \cap H_{(\Delta_i)} \rangle$. The result follows since $N_{i+1} = \pi_{i+1}(H_{(\Delta_i)})$. □

Recall that we compute the $\mathcal{P}_i$ iteratively. At the i-th iterative step, we will produce a strong generating set for H that is i-separable.

Definition 3.2.8. Let $\mathcal{P}_i$ be as in Definition 3.2.3. A strong generating set X of H with respect to an orbit-ordered base B is *i-separable* if for all $x \in X$ with non-trivial projection $\Pi_{\bar{i}}(x)$, there exists a unique cell C_j of $\mathcal{P}_i$ such that $\Pi_{\bar{i}}(x) \in \Pi_{C_j}(H) \times 1_{\bar{i} \setminus C_j}$. We say that X is *separable* if it is k-separable, where k is the number of orbits of H.

Note that since we have fixed an ordering of the H-orbits, by Proposition 3.2.2 the partitions $\mathcal{P}_i$ are unique, so i-separability depends only on i.

Example 3.2.9 (running example)**.** Let H be as in Example 1.4.5. Recall that we order the H-orbits by their smallest elements, so we have an ordering $\Omega_1, \Omega_2, \Omega_3, \Omega_4$. Then $B := (1, 4, 5, 7)$ is an orbit-ordered base since $H_{(\Delta_1)} = H_{(1)}$, $H_{(\Delta_2)} = H_{(1,4,5)}$, $H_{(\Delta_3)} = H_{(\Delta_4)} = H_{(1,4,5,7)}$. The generating set $X := \{x_1, x_2, x_3, x_4\}$ is a strong generating set of H with respect to the base B. So $\Pi_{\bar{2}}(H) = \langle (1,2,3),(4,5,6),(5,6) \rangle$ and hence $\mathcal{P}_2 = \langle \{1\} \mid \{2\} \rangle$. Then X is 2-separable since

$$\Pi_{\bar{2}}(x_1) \in \pi_1(H) \times 1_2 \quad \text{and} \quad \Pi_{\bar{2}}(x_2), \Pi_{\bar{2}}(x_3) \in 1_1 \times \pi_2(H) \quad \text{and} \quad \Pi_{\bar{2}}(x_4) = 1.$$

Similarly, we can show that $\mathcal{P}_3 = \langle \{1\} \mid \{2,3\} \rangle$. The set X is not 3-separable since $\pi_1(x_1)$ and $\Pi_{\{2,3\}}(x_1)$ are both non-trivial.

For each cell C_j of $\mathcal{P}_i$, we may compute $\theta_i(\Pi_{C_j}(H) \times 1_{\bar{i}\setminus C_j})$ using an i-separable strong generating set of H.

Lemma 3.2.10. *Let X be an i-separable strong generating set of H. Then for each cell C_j of $\mathcal{P}_i$, we have*

$$\theta_i(\Pi_{C_j}(H) \times 1_{\bar{i}\setminus C_j}) = \langle N_{i+1}\pi_{i+1}(x) \mid x \in X \textit{ such that } \Pi_{C_j}(x) \neq 1 \rangle.$$

Proof. Since X is i-separable, $\Pi_{C_j}(H) \times 1_{\bar{i}\setminus C_j}$ is generated by the $\Pi_{\bar{i}}(x)$ where $x \in X$ such that $\Pi_{C_j}(x) \neq 1$. So the result folllows from the definition of θ_i. □

Recall the sifting procedure and the definition of siftees from Definition 2.1.4. Note that g_s is a product of g and an element of H.

Lemma 3.2.11. *Let $g \in S_n$ and let g' be a* siftee *of g by H. Then there exists $h \in H$ such that $g = g'h$.*

Proof. Let B and the R_i be as in Definition 2.1.4. Then there exist $1 \leq s \leq m+1$ and $r_i \in R_i$ for all $1 \leq i \leq s-1$ such that $g' = g r_1^{-1} r_2^{-1} \dots r_{s-1}^{-1}$. Since the $R_i \subseteq H$, the result follows. □

Recall from Section 1.5 that a base $B = (\beta_1, \beta_2, \dots, \beta_m)$ of H defines a subgroup chain $H^{[1]} \geq H^{[2]} \geq \dots \geq H^{[m+1]}$ called a *stabiliser chain*, where $H^{[1]} := H$ and $H^{[i]} := H_{(\beta_1,\beta_2,\dots,\beta_{i-1})}$ for $2 \leq i \leq m+1$. The siftee of $g \in S_n$ by H obtained from a sifting procedure is not unique: it depends on the choice of the base B and the transversals R_i associated to the stabiliser chain defined by B.

In our algorithm, we will be sifting elements of H by stabilisers $H_{(\Delta_i)}$. Let $B := (\beta_1, \beta_2, \dots, \beta_m)$ be an orbit-ordered base of H with respect to the ordering $\Omega_1, \Omega_2, \dots, \Omega_k$. Then there exists $1 \leq j \leq m$ such that $(\beta_{j+1}, \beta_{j+2}, \dots, \beta_m)$ is a base of $H_{(\Delta_i)}$, with associated stabiliser chain $H^{[j+1]} \geq H^{[j+2]} \geq \dots \geq H^{[m+1]}$, which is a subchain of the stabiliser chain of H defined by B. Whenever we sift an element $h \in H$ by a stabiliser $H_{(\Delta_i)}$, we sift using the partial stabiliser chain defined by an orbit-ordered base B of H.

Definition 3.2.12. Let $B := (\beta_1, \beta_2, \dots, \beta_m)$ be an orbit-ordered base of H and let $1 \leq j \leq m$ such that $(\beta_{j+1}, \beta_{j+2}, \dots, \beta_m)$ is a base of $H_{(\Delta_i)}$. For $h \in H$, a *siftee of h by $H_{(\Delta_i)}$* is a siftee obtained by the sifting h using the stabiliser chain defined by base $(\beta_{j+1}, \beta_{j+2}, \dots, \beta_m)$.

For $x \in X$, we will be deciding whether $\pi_{i+1}(x) \in N_{i+1}$ by considering a siftee of x by $H_{(\Delta_i)}$.

Lemma 3.2.13. *Let B be an orbit-ordered base of $H \leq S_n$ with respect to the H-orbit ordering $\Omega_1, \Omega_2, \ldots, \Omega_k$. Let $x \in H$ and x' be a siftee of x by $H_{(\Delta_i)}$, where the sifting procedure uses the partial stabiliser chain of H defined by B, as in Definition 3.2.12. Then $\pi_{i+1}(x) \in N_{i+1}$ if and only if $\pi_{i+1}(x') = 1$.*

Proof. $\Leftarrow$: By Lemma 3.2.11, there exists $h \in H_{(\Delta_i)}$ such that $x = x'h$. Since $\pi_{i+1}(h) \in N_{i+1}$ and $\pi_{i+1}(x') = 1$, we have $\pi_{i+1}(x) = \pi_{i+1}(x')\pi_{i+1}(h) \in N_{i+1}$.
$\Rightarrow$: Since B is orbit-ordered, there exist $1 \leq t \leq u \leq m$ such that $H_{(\beta_1,\beta_2,\ldots,\beta_t)} = H_{(\Delta_i)}$ and $H_{(\beta_1,\beta_2,\ldots,\beta_u)} = H_{(\Delta_{i+1})}$. So $(\beta_{t+1}, \beta_{t+2}, \ldots, \beta_u)$ is a base of $\pi_{i+1}(H_{(\Delta_i)}) = N_{i+1}$. Since $\pi_{i+1}(x) \in N_{i+1}$, the sifting procedure does not terminate until after we have considered the image of β_u. More specifically, there exists $u \leq s \leq m+1$ and $r_i \in R_i$ for all $1 \leq i \leq s-1$ such that $x' = xr_{t+1}^{-1}r_{t+2}^{-1}\ldots r_{s-1}^{-1}$. Hence x' fixes $(\beta_{t+1}, \beta_{t+1}, \ldots, \beta_u)$. Therefore $\pi_{i+1}(x')$ is an element of N_{i+1} fixing the base of N_{i+1}, so $\pi_{i+1}(x') = 1$. □

3.3 Algorithm

In this section, we present an algorithm to compute the finest disjoint direct product decomposition of a given permutation group $H = \langle X \rangle \leq S_n$ and show that it has polynomial time complexity in terms of $|X|n$.

Recall Notation 3.1.2. For $1 \leq i \leq k$, let $\mathcal{P}_i$ be as in Definition 3.2.3. As the base case, we begin with $\mathcal{P}_1 := \langle\{1\}\rangle$ and we will compute $\mathcal{P}_{i+1}$ iteratively using Proposition 3.2.4.

Our algorithm to compute the finest disjoint direct product decomposition is presented in Algorithm 3. This algorithm computes a base and an initial strong generating set for H and then calls DDPD from Algorithm 4 repeatedly to calculate the $\mathcal{P}_i$.

Algorithm 3 Finding the finest disjoint direct product decomposition of H

```
Input: $H \leq S_n$ by a generating set $X$
Output: $\mathcal{P}_k$
1: Fix an ordering $\Omega_1, \Omega_2, \ldots, \Omega_k$ of the $H$-orbits
2: Find an orbit-ordered base $B$ of $H$ with respect to the ordering of $H$-orbits
3: Compute a strong generating set $X_1$ of $H$ relative to $B$
4: Initialise $\mathcal{P}_1 = \langle\{1\}\rangle$
5: for $i \in [1, 2, .., k-1]$ do
6:     $X_{i+1}, \mathcal{P}_{i+1} \leftarrow$ DDPD$(i, B, X_i, \mathcal{P}_i)$
7: end for
8: return $\mathcal{P}_k$
```

We give an example for Algorithm 4.

Example 3.3.1 (running example)**.** Consider H in Example 3.2.9. We have seen that $\mathcal{P}_2 = \langle\{1\} \mid \{2\}\rangle$ and X is 2-separable. To compute $\mathcal{P}_3$, initialise $S := \{\}$. Observe

Algorithm 4 Finding the finest disjoint direct product decomposition of $\Pi_{\overline{i+1}}(H)$

Input: Integer $1 \leq i \leq k$, base B of H, partition $\mathcal{P}_i = \langle C_1 \mid C_2 \mid \ldots \mid C_r \rangle$ of $\{1, 2, \ldots, k\}$ and an i-separable generating set X_i of H

Output: $\mathcal{P}_{i+1}$ and an $(i+1)$-separable generating set X_{i+1} of H

```
1: procedure DDPD($i$, $B$, $X_i$, $\mathcal{P}_i$)
2:     $S \leftarrow \{\}$
3:     $X_{i+1} \leftarrow \{\}$
4:     for $x \in X_i$ do
5:         if $x \notin H_{(\Delta_i)}$ then
6:             Find cell $C_j$ of $\mathcal{P}_i$ such that $\Pi_{\bar{i}}(x) \in \Pi_{C_j}(H) \times 1_{\bar{i} \setminus C_j}$
7:             $x' \leftarrow$ siftee of $x$ by $H_{(\Delta_i)}$ using the partial stabiliser chain defined by $B$
                                                                   ▷ see Definition 3.2.12
8:             Add $x'$ to $X_{i+1}$
9:             if $\pi_{i+1}(x') \neq 1$ then
10:                Add $C_j$ to $S$
11:            end if
12:        else
13:            Add $x$ to $X_{i+1}$
14:        end if
15:    end for
16:    $C \leftarrow (\bigcup_{C_j \in S} C_j) \cup \{i+1\}$
17:    $\mathcal{P}_{i+1} \leftarrow$ the partition consisting of cell $C$ and all other cells $C_j$ of $\mathcal{P}_i$ such that
               $C_j \notin S$
18:    return $X_{i+1}, \mathcal{P}_{i+1}$
19: end procedure
```

that $\Pi_{\bar{2}}(x_1) \in \Pi_1(H) \times 1_2$ and $x_1 \notin H_{(\Delta_2)}$. Sifting x_1 by $H_{(\Delta_2)}$ gives us a siftee $(1, 2, 3)$. Similarly, $\Pi_{\bar{2}}(x_2) \in \Pi_2(H) \times 1_1$ and $x_2 \notin H_{(\Delta_2)}$ with a siftee $(4, 5, 6)$. Now $\Pi_{\bar{2}}(x_3) \in \Pi_2(H) \times 1_1$ and $x_3 \notin H_{(\Delta_2)}$ has a siftee $(5, 6)(8, 9)(11, 12)$, which has a non-trivial projection on $\Omega_3 = \{7, 8, 9\}$. So we add $\{2\}$ to S. Since $x_4 \in H_{(\Delta_2)}$ we add x_4 to X_3. Therefore, we set $C = S \cup \{3\} = \{2, 3\}$, so $\mathcal{P}_3 = \langle \{1\} \mid \{2, 3\} \rangle$ and $X_3 = \{(1, 2, 3), (4, 5, 6), (5, 6)(8, 9)(11, 12), (7, 8, 9)(10, 11, 12)\}$.

Lemma 3.3.2. *For $1 \leq i \leq k$, the sets X_i computed in Algorithms 3 and 4 are strong generating sets of H with respect to the base $B = (\beta_1, \beta_2, \ldots, \beta_m)$ defined on line 2 of Algorithm 3.*

Proof. We proceed by induction on i. Clearly X_1 is a strong generating set of H with respect to B. Suppose that X_i is a strong generating set of H with respect to B. First note that since each $x \in X_i \cap H_{(\Delta_i)}$ is also in X_{i+1} and a siftee of $x \in X_i \setminus H_{(\Delta_i)}$ by $H_{(\Delta_i)}$ is not in $H_{(\Delta_i)}$, we have $X_i \cap H_{(\Delta_i)} = X_{i+1} \cap H_{(\Delta_i)}$. Let $0 \leq s \leq m$ and consider $H^{[s+1]} = H_{(\beta_1, \beta_2, \ldots, \beta_s)}$. Since B is an orbit-ordered base, either $H^{[s+1]} \subseteq H_{(\Delta_i)}$ or $H_{(\Delta_i)} \subseteq H^{[s+1]}$. For both cases, we will show that $H^{[s+1]} = \langle X_{i+1} \cap H^{[s+1]} \rangle$.

Suppose first that $H^{[s+1]} \subseteq H_{(\Delta_i)}$. Then

$$X_i \cap H^{[s+1]} = X_i \cap H_{(\Delta_i)} \cap H^{[s+1]} = X_{i+1} \cap H_{(\Delta_i)} \cap H^{[s+1]} = X_{i+1} \cap H^{[s+1]}.$$

Since X_i is a strong generating set, we have $H^{[s+1]} = \langle X_i \cap H^{[s+1]} \rangle = \langle X_{i+1} \cap H^{[s+1]} \rangle$. Suppose instead that $H_{(\Delta_i)} \subseteq H^{[s+1]}$. Certainly $\langle X_{i+1} \cap H^{[s+1]} \rangle \leq H^{[s+1]}$. To show the reverse containment, since X_i is a strong generating set, it suffices to show that $X_i \cap H^{[s+1]} \subseteq \langle X_{i+1} \cap H^{[s+1]} \rangle$. Let $x \in X_i \cap H^{[s+1]}$. If $x \in H_{(\Delta_i)}$, then by the construction of X_{i+1}, we have $x \in X_{i+1}$, hence $x \in X_{i+1} \cap H^{[s+1]}$. Else suppose that $x \in (X_i \cap H^{[s+1]}) \backslash H_{(\Delta_i)}$. Then there exists a siftee x' of x by $H_{(\Delta_i)}$ such that $x' \in X_{i+1}$. By Lemma 3.2.11, there exists $g \in H_{(\Delta_i)} \leq H^{[s+1]}$ such that $x = x'g$. Then $x' = xg^{-1} \in H^{[s+1]}$ and so $x' \in X_{i+1} \cap H^{[s+1]}$. Observe that

$$\begin{aligned} H_{(\Delta_i)} &= \langle X_i \cap H_{(\Delta_i)} \rangle && \text{since } X_i \text{ is a strong generating set} \\ &= \langle X_{i+1} \cap H_{(\Delta_i)} \rangle && \text{since } X_i \cap H_{(\Delta_i)} = X_{i+1} \cap H_{(\Delta_i)} \\ &\subseteq \langle X_{i+1} \cap H^{[s+1]} \rangle && \text{since } H_{(\Delta_i)} \subseteq H^{[s+1]}. \end{aligned}$$

So $g \in \langle X_{i+1} \cap H^{[s+1]} \rangle$ and hence $x \in \langle X_{i+1} \cap H^{[s+1]} \rangle$. □

Lemma 3.3.3. *Let $1 \leq i \leq k-1$. Let X_i be an i-separable strong generating set of H and let $\mathcal{P}_i = \langle C_1 \mid C_2 \mid \ldots \mid C_r \rangle$ be as in Definition 3.2.3. Let X_{i+1} and Q be the output of* $\mathrm{DDPD}(i, B, X_i, \mathcal{P}_i)$ *from Algorithm 4. Then*

1. *$\mathcal{P}_{i+1} = Q$.*

2. *X_{i+1} is an $(i+1)$-separable strong generating set of H.*

Hence Algorithm 4 is correct.

Proof. Part 1: We show that the set S constructed from Algorithm 4 is the same as the set in Proposition 3.2.4, from which the result will follow. Observe that we add C_j to S in line 10 if and only if there exists $x \in X_i$ with non-trivial projection $\Pi_{\bar{i}}(x)$ such that $\Pi_{\bar{i}}(x) \in \Pi_{C_j}(H) \times 1_{\bar{i}\backslash C_j}$ and $\pi_{i+1}(x') \neq 1$, where x' is a siftee of x by $H_{(\Delta_i)}$. Since X_i is i-separable, by Lemmas 3.2.10 and 3.2.13, such an $x \in X_i$ exists if and only if $\theta_i(\Pi_{C_j}(H) \times 1_{\bar{i}\backslash C_j}) \neq N_{i+1}$. That is, $\Pi_{C_j}(H) \times 1_{\bar{i}\backslash C_j} \nsubseteq \mathrm{Ker}(\theta_i)$. The result then follows from Proposition 3.2.4.
Part 2: Let $y \in X_{i+1}$. We show that if $\Pi_{\overline{i+1}}(y) \neq 1$, then there exists a unique cell C' of $\mathcal{P}_{i+1}$ such that $\Pi_{\overline{i+1}}(y) \in \Pi_{C'}(H) \times 1_{\overline{i+1}\backslash C'}$.
By the construction of X_{i+1}, either $y \in X_i \cap H_{(\Delta_i)}$ or there exists $x \in X_i \backslash H_{(\Delta_i)}$ such that y is a siftee of x by $H_{(\Delta_i)}$. Suppose first that $y \in X_i \cap H_{(\Delta_i)}$, so $\Pi_{\bar{i}}(y) = 1$. If $\Pi_{\overline{i+1}}(y) \neq 1$, then $\pi_{i+1}(y) \neq 1$ and so $\Pi_C(y) \neq 1$, where C is as defined in line 16. Since $\overline{i+1}\backslash C = \bar{i}\backslash C \subseteq \bar{i}$, we have $\Pi_{\overline{i+1}\backslash C}(y) = 1$. So C is the unique cell of $\mathcal{P}_{i+1}$ such that $\Pi_{\overline{i+1}}(y) = \Pi_C(y) \times 1_{\overline{i+1}\backslash C} \in \Pi_C(H) \times 1_{\overline{i+1}\backslash C}$.
Suppose now that y is a siftee of some $x \in X_i \backslash H_{(\Delta_i)}$ by $H_{(\Delta_i)}$. By Lemma 3.2.11, there

exists $g \in H_{(\Delta_i)}$ such that $x = yg$. Then since $\Pi_{\bar{i}}(g) = 1$, we have $\Pi_{\bar{i}}(x) = \Pi_{\bar{i}}(y)$. As X_i is i-separable, there exists exactly one cell C_j of $\mathcal{P}_i$ such that

$$\Pi_{\bar{i}}(y) = \Pi_{\bar{i}}(x) \in \Pi_{C_j}(H) \times 1_{\bar{i} \setminus C_j}.$$

If $C_j \notin S$, then C_j is a cell of $\mathcal{P}_{i+1}$ and so $\Pi_{\overline{i+1}}(y) \in \Pi_{C_j}(H) \times 1_{\overline{i+1} \setminus C_j}$. Otherwise if $C_j \in S$, then $\Pi_{\overline{i+1}}(y) \in \Pi_{C_j \cup \{i+1\}}(H) \times 1_{\bar{i} \setminus C_j}$. In particular, since $C_j \cup \{i+1\} \subseteq C$, we have $\Pi_{\overline{i+1}}(y) \in \Pi_C(H) \times 1_{\overline{i+1} \setminus C}$. □

Theorem 3.3.4. *Given $H \leq S_n$ by a generating set X, Algorithm 3 computes the finest disjoint direct product decomposition of H in time polynomial in $|X|n$.*

Proof. Let $Q \coloneqq \langle C_1 \mid C_2 \mid \ldots \mid C_r \rangle$ be the partition computed by Algorithm 3. Then by Part 1 of Lemma 3.3.3, $Q = \mathcal{P}_k$, and so $H = \Pi_{C_1}(H) \times \Pi_{C_2}(H) \times \ldots \times \Pi_{C_r}(H)$ is the finest disjoint direct product decomposition of H.
It remains to show that Algorithm 3 runs in polynomial time. By Proposition 2.1.2 and Theorem 2.1.7, lines 1 and 3 run in polynomial time. As in Remark 3.2.6, we can get an orbit-ordered base in polynomial time. Since Algorithm 4 is called at most $O(n)$ times, it suffices to show that Algorithm 4 runs in time polynomial in $|X|n$.
By Parts 2 and 3 of Theorem 2.1.9, lines 5 and 7 can be done in polynomial time. Line 6 can be done by iterating through each cell C_j of $\mathcal{P}_i$ and checking if $\Pi_{C_j}(x) \neq 1$. Since $\mathcal{P}_i$ has at most k cells, this can be done in $O(k)$ time. Therefore the result follows. □

Hence Theorem 3.0.2 now follows.

3.4 Experiments

In this section, we will investigate the practical performance of our algorithm for finding the finest disjoint direct product decomposition of a permutation group. As well as showing the performance of our algorithm, we will show how it can be used to improve the performance of a range of important group-theoretic problems. Our algorithm is implemented in GAP 4.11 [GAP20] [1].

We will test our algorithm on randomly created groups. The creator we use in this chapter is very straightforward and does not claim to produce all groups with equal probability. It is costly to attain better distribution as we require random normal subgroups and isomorphisms, hence this method is chosen for its simplicity and speed.

The creator takes three parameters, a transitive permutation group G and two integer constants r and s. The algorithm will produce a permutation group which is the disjoint direct product of r groups. Each of these direct factors is a subdirect product of G^s that is d.d.p. indecomposable. The algorithm runs in two stages.

[1]The implementation for the experiments in this section is done by Dr Christopher Jefferson, which better exploits the stabiliser chain data structure in GAP. An independent implementation by the author is included in the supplementary files. The difference in terms of the runtimes of the two implementations is minor.

The first stage is implemented by a function MakeSubdirect(G, s) which produces a random subdirect product of s copies of G that is d.d.p. indecomposable. MakeSubdirect(G, s) works by taking a random integer $i \in \{2, 3, \ldots, s\}$ and then taking the group H generated by i random elements of G^s. If H is d.d.p. indecomposable, and its projection onto each of the s copies of G is surjective, then it is returned, else this procedure repeats.

The second stage simply runs MakeSubdirect(G, s) r times, and takes the (disjoint) direct product of these r groups. Finally, we conjugate this group by a random permutation on the set of points moved by this group, so the decomposition does not follow the natural ordering of the integers.

Our algorithms are split into two sections. Firstly we compare our algorithm against the algorithm of Donaldson and Miller [DM09]. Secondly, to demonstrate that our algorithm has immediate practical value, we show how a few functions in GAP can easily be sped up by the knowledge of a disjoint direct product decomposition.

3.4.1 Comparison to Donaldson and Miller

We first compare to Donaldson and Miller's algorithm. Donaldson and Miller present two algorithms, an incomplete algorithm and a complete algorithm. We will not compare against their incomplete algorithm, as it is extremely fast but requires separable strong generating sets. Donaldson and Miller were unable to find a graph where the generating set of the automorphism group produced by Nauty [MP14] is not separable. We were able to find graphs where Nauty does not produce a separable generating set [2]. Also, the most advanced graph automorphism finders, such as Traces [MP14], perform random dives which produce random automorphisms. This means that generating sets produced by Traces will not, in general, be separable.

When implementing Donaldson and Miller's complete algorithm, we were forced to make some implementation choices, the most significant of which is the ordering in which the algorithm tries to partition the orbits (the first line of Algorithm 5 in [DM09]). We implemented this by trying the partitions in order of increasing sizes of the smaller part of the partition. This has the advantage that when the finest disjoint direct product decomposition has a factor with few orbits the algorithm will run relatively quickly, but it has no effect on the algorithm's worst case complexity.

We gather our results in Figure 3.1. Each row of Figure 3.1 gives the result for 10 instances of H, created as above. Each instance of H has rs orbits with the finest disjoint direct product decomposition consisting of r direct factors with s orbits each, and each transitive constituent of H is permutation isomorphic to G. For each G, r and s, we report the median time ("Median") and the number of completed instances ("#") of the 10 instances of H, using both Algorithm 5 of [DM09] ("Donaldson") and Algorithm 3 ("Our Algorithm"). The results are consistent with the theory. Since our

[2]One example can be found in the supplemental file `Chapter3-DDPD/nonSeparableExample.g`.

algorithm is polynomial, it scales much better than the algorithm in [DM09].

G	r	s	Donaldson		Our Algorithm	
			Median	#	Median	#
D_8	4	4	0.04	10	0.00	10
D_8	6	4	0.16	10	0.01	10
D_8	8	4	0.54	10	0.01	10
D_8	10	4	2.18	10	0.02	10
A_4	4	4	2.47	10	0.01	10
A_4	6	4	12.47	10	0.03	10
A_4	8	4	158.09	10	0.04	10
A_4	10	4	490.54	5	0.05	10
S_4	4	4	3.37	10	0.02	10
S_4	6	4	15.99	10	0.04	10
S_4	8	4	393.86	8	0.06	10
S_4	10	4	N/A	0	0.07	10

Figure 3.1: Comparison of DDPD algorithms

3.4.2 Application to GAP

In this section, we will see how some functions in GAP can easily be sped up by the knowledge of a disjoint direct product decomposition. This is not intended to be an exhaustive list, but simply to demonstrate how, for a selection of common problems, calculating a disjoint direct product decomposition can significantly improve performance. We experiment on the following GAP functions.

DerivedSubgroup - The derived subgroup of a group H is the direct product of the derived subgroup of each disjoint direct factor of H.

NrConjugacyClasses - The number of conjugacy classes of a group H is the product of the number of conjugacy classes of each disjoint direct factor of H.

CompositionSeries - A composition series of a group H can be constructed from composition series of the direct factors of H.

MinimalNormalSubgroups - If each disjoint direct factor of H is non-abelian, the minimal normal subgroups of a group H is the union of the minimal normal subgroups of each disjoint direct factor of H.

In our experiments, we run each row of our tables ten times. Each run is given a limit of 10 minutes and 4GB of memory. We give the median time in seconds (or N/A when less than 6 instances finished successfully). For the inner group G we consider the alternating group (A_n), symmetric group (S_n), dihedral groups (D_{2n}) of varying degree n, and also TransitiveGroup$(16, 712)$ ($Trans(16, 712)$) and TransitiveGroup$(16, 713)$ ($Trans(16, 713)$) from the Transitive Groups Library [Hul17].

Each row of the tables in Figures 3.2 to 3.5 gives the results for 10 random groups H, each with rs orbits, where the projection of H onto each orbit is permutation isomorphic to G, and H has the finest disjoint direct decomposition consisting of r direct factors with s orbits each. The columns "Full Group" and "Decomposed Group" refers to the computation with the original group H and the computation with the disjoint direct factors of H respectively. The column "Decomposition" refers to the computation of the finest disjoint direct product decomposition of H. For each of these columns, we report the median time (in seconds) required to compute the specified problems of the 10 instances under the subcolumns "Median", and the number of instances completed within the time and memory limits under the subcolumns "#".

Fro all of our experiments apart from CompositionSeries, the time taken to find the finest disjoint direct product decomposition and solve the problem on the decomposed group is always faster than solving the problem on the original full group. In the case of DerivedSubgroup (Figure 3.2), we speed up performance by up to a factor of 10. In the case of NrConjugacyClasses (Figure 3.3), CompositionSeries (Figure 3.4) and MinimalNormalSubgroups (Figure 3.5), we are able to solve problems which previously ran out of memory or time, in under a second.

G	r	s	Full Group		Decomposed Group		Decomposition	
			Median	#	Median	#	Median	#
D_8	12	4	0.33	10	0.00	10	0.20	10
D_8	16	4	1.07	10	0.00	10	0.55	10
D_8	20	4	2.53	10	0.00	10	1.20	10
A_4	12	4	2.34	10	0.01	10	0.28	10
A_4	16	4	7.53	10	0.01	10	0.66	10
A_4	20	4	19.76	10	0.01	10	1.64	10
S_4	12	4	7.36	10	0.01	10	0.89	10
S_4	16	4	27.89	10	0.02	9	2.34	10
S_4	20	4	65.85	10	0.02	10	5.37	10

Figure 3.2: Performance of DDPD for DerivedSubgroup

3.5 Conclusion and future work

In this chapter, we have shown that the finest disjoint direct product decomposition of a given group can be computed efficiently and can be used to speed up various permutation group problems. Moreover, as demonstrated in [DM09, Gra11], the disjoint direct product decomposition of a group has applications beyond computational group theory.

G	r	s	Full Group		Decomposed Group		Decomposition	
			Median	#	Median	#	Median	#
A_3	2	4	0.00	10	0.00	10	0.00	10
A_3	4	4	0.08	10	0.00	10	0.00	10
A_3	6	4	21.50	6	0.00	10	0.00	10
D_8	2	4	0.23	10	0.00	10	0.00	10
D_8	4	4	N/A	2	0.00	10	0.00	10
D_8	6	4	N/A	0	0.01	10	0.00	10
S_4	2	4	50.52	10	0.28	10	0.00	10
S_4	4	4	N/A	0	0.58	10	0.00	10
S_4	6	4	N/A	0	0.96	10	0.02	10

Figure 3.3: Performance of DDPD for NrConjugacyClasses

G	r	s	Full Group		Decomposed Group		Decomposition	
			Median	#	Median	#	Median	#
A_4	4	3	0.00	10	0.00	10	0.00	10
A_4	12	3	0.06	10	0.01	10	0.11	10
A_4	20	3	0.22	10	0.02	10	0.68	10
S_4	4	3	0.01	10	0.00	10	0.00	10
S_4	12	3	0.15	10	0.02	10	0.32	10
S_4	20	3	0.49	10	0.03	10	1.70	10
D_{32}	4	3	0.01	10	0.00	10	0.00	10
D_{32}	12	3	0.11	10	0.02	10	0.27	10
D_{32}	20	3	0.45	10	0.04	10	1.87	10
$Trans(16, 712)$	4	3	0.02	10	0.01	10	0.03	10
$Trans(16, 712)$	12	3	0.36	10	0.03	10	1.25	10
$Trans(16, 712)$	20	3	1.52	10	0.06	10	8.72	10
$Trans(16, 713)$	4	3	0.64	10	0.02	10	0.03	10
$Trans(16, 713)$	12	3	80.49	10	0.07	10	1.32	10
$Trans(16, 713)$	20	3	N/A	0	0.12	10	8.57	10

Figure 3.4: Performance of DDPD for CompositionSeries

G	r	s	Full Group		Decomposed Group		Decomposition	
			Median	#	Median	#	Median	#
D_8	4	4	1.88	10	0.03	10	0.00	10
D_8	6	4	N/A	4	0.04	10	0.02	10
D_8	8	4	N/A	0	0.06	10	0.12	10
D_8	10	4	N/A	0	0.07	10	0.35	10
A_4	4	4	1.60	10	0.23	10	0.01	10
A_4	6	4	6.49	10	0.50	10	0.02	10
A_4	8	4	27.66	10	0.47	10	0.20	10
A_4	10	4	57.10	10	0.55	10	0.39	10
S_4	4	4	4.59	10	0.40	10	0.01	10
S_4	6	4	20.44	10	0.58	10	0.04	10
S_4	8	4	102.73	10	0.80	10	0.75	10
S_4	10	4	254.37	10	1.23	10	1.09	10

Figure 3.5: Performance of DDPD for MinimalNormalSubgroups

While we show the disjoint direct product decomposition can be extremely useful, we are not suggesting it to be employed as an initial subprocedure of solving all the problems we use in our experiments. This is because adding this subprocedure will impose additional computation time on all calls of the problem that could be a waste of time if the group is d.d.p. indecomposable. Using efficient heuristics to only add this subprocedure for some groups still has the same problem with the added cost and raises an issue of determining the heuristics. Therefore, we propose that the computation of the finest disjoint direct product decomposition be available in GAP as a function, and leave it up to the user to decide if the decomposition would help the problem in hand.

An obvious piece of future work is to determine other problems, group theoretic or otherwise, that can benefit from the knowledge of its disjoint direct product decomposition. We believe that the disjoint direct product decomposition has more potential in groups arising from real world problems, as these are more likely to be highly intransitive.

Chapter 4

Normalisers of Highly Intransitive Groups

In this chapter, we will compute the normalisers $N_{S_n}(H)$ of intransitive groups $H \leq S_n$. We will see how certain structures of H can be used to reduce the search tree for computing $N_{S_n}(H)$. We do so in two ways: firstly we use the group structure to create conjugacy invariant functions and refiners (see Definitions 2.4.10 and 2.4.13), and secondly we see how these structures can be used to reduce the computation of $N_{S_n}(H)$ into smaller problems. We will also give an algorithm for computing normalisers $N_{S_n}(H)$ of intransitive $H \leq S_n$ with pairwise permutation isomorphic transitive constituents. The algorithm uses conjugacy invariant functions and refiners to produce a subgroup S of S_n containing $N_{S_n}(H)$.

More specifically, the structure of the chapter is as follows. In Section 4.1, we use the equivalence of orbits to give conjugacy invariant functions and refiners, and then reduce the computation of $N_{S_n}(H)$ to computing the normaliser of a group with a smaller degree. In Section 4.2, we first use disjoint direct product decompositions to give conjugacy invariant functions and refiners. If H has a disjoint direct product decomposition with more than two factors, we show that $N_{S_n}(H)$ can be computed by computing the normaliser of each disjoint direct factor, and then solving the conjugacy problem for each pair of the factors. In Section 4.3, we use the permutation isomorphism classes of certain projections to give some conjugacy invariant functions and refiners. We will introduce the class $\mathfrak{InP}(A)$ consisting of groups whose transitive constituents are all permutation isomorphic to a transitive group $A \leq S_m$, and then give a proper subgroup of S_n containing $N_{S_n}(H)$ for $H \leq S_n$ in class $\mathfrak{InP}(A)$. In Section 4.4, we give an algorithm which uses the conjugacy invariant functions and refiners from the previous sections to compute a group $S \leq S_n$ containing $N_{S_n}(H)$, where H is in $\mathfrak{InP}(A)$ for some transitive group A, and so $N_{S_n}(H) = N_S(H)$. Lastly in Section 4.5, we show that this subgroup $S \leq S_n$ can be used to speed up the computation of $N_{S_n}(H)$.

In this chapter, we will take the natural inclusion map of $\mathrm{Sym}(\Gamma)$ into $\mathrm{Sym}(\Delta)$ for all $\Gamma \subseteq \Delta$. For the rest of this chapter, let n be an integer and let $\Omega = \{1, 2, \ldots, n\}$.

4.1 Equivalent orbits

Recall the definitions of equivalent orbits and $\equiv_o$ from Definition 2.1.10. In this section, we show how we can use equivalent orbits to obtain conjugacy invariant functions and refiners. Then we show how we can use the equivalent orbits of $H \leq S_n$ to reduce the computation of $N_{S_n}(H)$ into computing the centraliser $C_{S_n}(H)$ and the normaliser of a group with a smaller degree.

4.1.1 Pruning with equivalent orbits

We start by showing that conjugation preserves the equivalence classes under $\equiv_o$.

Lemma 4.1.1. *Let $K, L \leq S_n$ such that $L = K^g$ for some $g \in S_n$. Then $\{\Omega_i \mid i \in I\}$ is an $\equiv_o$-class of K-orbits if and only if $\{\Omega_i^g \mid i \in I\}$ is an $\equiv_o$-class of L-orbits. Hence the normaliser $N_{S_n}(H)$ permutes the $\equiv_o$-classes of H-orbits.*

Proof. Let $i, j \in I$. Then Ω_i^g and Ω_j^g are orbits of L. We show that $\Omega_i^g \equiv_o \Omega_j^g$ using Lemma 2.1.12. Let $l \in L$ and let $k = l^{g^{-1}} \in K$. By the forward implication of Lemma 2.1.12, there exists an involution $\nu \in S_n$ with support $\Omega_i \cup \Omega_j$ such that for all $k' \in K$, we have $k'|_{\Omega_j} = (k'|_{\Omega_i})^\nu$. Then

$$(l|_{\Omega_i^g})^{g^{-1}\nu g} = ((k|_{\Omega_i})^g)^{g^{-1}\nu g} = (k|_{\Omega_i})^{\nu g} = (k|_{\Omega_j})^g = l|_{\Omega_j^g},$$

and $(l|_{\Omega_j^g})^{g^{-1}\nu g} = l|_{\Omega_i^g}$ similarly. Since ν has support $\Omega_i \cup \Omega_i$, its conjugate ν^g has support $\Omega_i^g \cup \Omega_i^g$. Therefore, by the backward implication of Lemma 2.1.12, we have $\Omega_i^g \equiv_o \Omega_i^g$. The fact that inequivalent orbits maps to non-equivalent orbits follows by symmetry. $\square$

Therefore we may use orbit equivalence to obtain a conjugacy invariant function and a conjugacy invariant refiner. Recall that we denote the set of all subgroups of S_n by $\mathcal{S}(S_n)$.

Lemma 4.1.2. *For $K \leq S_n$, let E_K be the function from the set of orbits of K to $\mathbb{N}$ such that $E_K(\Delta)$ gives the size of the equivalence class $[\Delta]_{\equiv_o}$.*

1. *Let Ψ be the function with domain $\mathcal{S}(S_n)$ such that $\Psi(K)$ gives the multiset $\{E_K(\Delta) \mid \text{for all orbits } \Delta \text{ of } K\}$. Then Ψ is a conjugacy invariant function.*

2. *Let Φ be the function with domain $\mathcal{S}(S_n) \times \Omega$ defined by $\Phi(K, \alpha) = E_K(\Delta)$ where Δ is the K-orbit containing α. Then Φ is a conjugacy invariant refiner.*

We denote Ψ and Φ by EquivOrbsCIF *and* EquivOrbsCIR *respectively.*

Proof. Let $K, L \leq S_n$ such that $L = K^g$ for some $g \in S_n$.
Part 1: By Lemma 4.1.1, g maps each $\equiv_o$-class of K-orbits to an $\equiv_o$-class of L-orbits with the same size. Hence $\Psi(L) = \Psi(K)$.

Part 2: Let $\alpha \in \Omega$ and let Δ be the K-orbit containing α. Then Δ^g is the orbit of L containing α^g. By the argument in Part 1, we have $E_L(\Delta^g) = E_K(\Delta)$ and hence $\Phi(K, \alpha) = \Phi(L, \alpha^g)$. $\square$

4.1.2 Degree reduction using equivalent orbits

In the last subsection, we saw how equivalent orbits are used to obtain a conjugacy invariant function Ψ and refiner Φ. In this subsection, we use equivalent orbits of $H \leq S_n$ to show that if $\equiv_o$ is not the equality relation then the normaliser $N_{S_n}(H)$ can be computed by computing $C_{S_n}(H)$ and the normaliser of a group with no equivalent orbits and a smaller degree. Hence when convenient, we may assume that H has no equivalent orbits.

We start by constructing a partition of some H-orbits based on the sizes of the $\equiv_o$-classes.

Definition 4.1.3. Let t, the $\mathcal{B}_i$, Ω_{ib} and $\overline{\psi_{ib}}$ for all $1 \leq i \leq t$ and $2 \leq b \leq |\mathcal{B}_i|$ be as in Proposition 2.1.13. Let $P_e(H)$ be the partition of $\Omega_{11}, \Omega_{21}, \dots, \Omega_{t1}$ such that Ω_{i1} and Ω_{j1} are in the same cell if and only if $|\mathcal{B}_i| = |\mathcal{B}_j|$. Let $\Gamma := \bigcup_{i=1}^{t} \Omega_{i1}$.

Definition 4.1.4. With the notation of Definition 4.1.3, let S be the stabiliser of $P_e(H)$ in $\mathrm{Sym}(\Gamma)$. Let the maps $\psi_{ij} : \Omega_{i1} \to \Omega_{ij}$ be as in Proposition 2.1.13, and let ψ_{i1} be the identity mapping. Let $\theta : N_S(H|_\Gamma) \to S_n$ be defined by

$$\text{for } \nu \in N_S(H|_\Gamma), \text{ if } \alpha \in \Omega_{ib} \text{ and } \Omega_{i1}^{\nu} = \Omega_{j1}, \text{ define } \alpha^{\theta(\nu)} = \alpha^{\overline{\psi_{ib}}^{-1}\nu\overline{\psi_{jb}}} \in \Omega_{jb}.$$

One can check that the $\theta(\nu)$ is indeed in S_n. Since the definition in Definition 4.1.4 is rather technical, we include an example here.

Example 4.1.5. Let $H = \langle (1,2)(3,4), (5,6)(7,8) \rangle$. Then $\Omega_{11} := \{1,2\}$ and $\Omega_{12} := \{3,4\}$ are equivalent H-orbits, with the map $\psi_{12} : \Omega_{11} \to \Omega_{12}$ defined by $1 \mapsto 3$ and $2 \mapsto 4$ as a bijection witnessing the orbit equivalence. Similarly $\Omega_{21} := \{5,6\}$ and $\Omega_{22} := \{7,8\}$ are equivalent H-orbits, and the map $\psi_{22} : \Omega_{21} \to \Omega_{22}$ defined by $5 \mapsto 7$ and $6 \mapsto 8$ witnesses the equivalence. So we have $\overline{\psi_{12}} = (1,3)(2,4)$ and $\overline{\psi_{22}} = (5,7)(6,8)$.
Let $\Gamma = \{1,2,5,6\}$, $S = \mathrm{Sym}(\Gamma)$ and $H|_\Gamma = \langle (1,2), (5,6) \rangle$. So $\nu := (1,6)(2,5) \in N_S(H|_\Gamma)$. Let $\theta : N_S(H|_\Gamma) \to S_n$ be as in Definition 4.1.4. Then $\theta(\nu)$ maps $1 \in \Omega_{11}$ to $1^{()*(1,6)(2,5)*()} = 6$. Similarly $2^{\theta(\nu)} = 5$, $5^{\theta(\nu)} = 2$ and $6^{\theta(\nu)} = 1$.
Now consider $3 \in \Omega_{12}$. Then $\theta(\nu)$ maps 3 to $3^{(1,3)(2,4)*(1,6)(2,5)*(5,7)(6,8)} = 8$. Similarly $4^{\theta(\nu)} = 7$, $7^{\theta(\nu)} = 4$ and $8^{\theta(\nu)} = 3$. So $\theta(\nu) = (1,6)(2,5)(3,8)(4,7)$.
Lastly observe that $\theta(\nu) \in N_{S_n}(H)$.

Proposition 4.1.6. *With the notation of Definition 4.1.3, let S be the stabiliser of $P_e(H)$ in $\mathrm{Sym}(\Gamma)$, and let $\theta : N_S(H|_\Gamma) \to S_n$ be as in Definition 4.1.4. Then $N_{S_n}(H) = \langle \mathrm{Im}(\theta), C_{S_n}(H) \rangle$.*

Proof. $\geq$: It suffices to show that $\mathrm{Im}(\theta) \subseteq N_{S_n}(H)$. Let $h \in H$ and $\nu \in N_S(H|_\Gamma)$. Since $(h|_\Gamma)^\nu \in H|_\Gamma$, there exists $h' \in H$ such that $h'|_\Gamma = (h|_\Gamma)^\nu$. We will show that $h^{\theta(\nu)} = h'$.
We first show that $h^{\theta(\nu)}|_\Gamma = h'|_\Gamma$. Let $\Omega_{i1}^\nu = \Omega_{j1}$ and let $\alpha \in \Omega_{i1}$. Then $\alpha^{\theta(\nu)} = \alpha^\nu \in \Gamma$ as $\overline{\psi_{i1}} = \overline{\psi_{j1}} = 1$. Therefore, $h^{\theta(\nu)}|_\Gamma = (h|_\Gamma)^\nu = h'|_\Gamma$.
Let $1 \leq i \leq t$ and $2 \leq b \leq |\mathcal{B}_i|$. Then by definition of $\overline{\psi_{ib}}$,

$$h^{\theta(\nu)}|_{\Omega_{ib}} = (h^{\theta(\nu)}|_{\Omega_{i1}})^{\overline{\psi_{ib}}} = (h'|_{\Omega_{i1}})^{\overline{\psi_{ib}}} = h'|_{\Omega_{ib}},$$

as required.
$\leq$: Let $g \in N_{S_n}(H)$. By Lemma 4.1.1, g permutes the equivalence classes of $\equiv_o$. By Proposition 2.1.13, $C_{S_n}(H)$ induces the full symmetric group on each equivalence class of orbits. So, there exists $c \in C_{S_n}(H)$ such that gc fixes Γ setwise. Then $(gc)|_\Gamma \in N_S(H|_\Gamma)$. Let $g' := \theta((gc)|_\Gamma) \in \mathrm{Im}(\theta)$ and $h \in H$. Then $(h|_\Gamma)^{gc} = (h|_\Gamma)^{g'}$ and so $h^{gc} = h^{g'}$. Then $gcg'^{-1} \in C_{S_n}(H)$ and hence $g \in \langle \mathrm{Im}(\theta), C_{S_n}(H)\rangle$. □

Hence, the computation of $N_{S_n}(H)$ can be reduced to computing $C_{S_n}(H)$ and the normaliser $N_S(H|_\Gamma)$ and its image under the map θ.

4.2 Disjoint direct product decompositions

Recall the definition of disjoint direct product decomposition from Definition 3.0.1. In this section, we use the disjoint direct product decompositions of groups to obtain a conjugation invariant function and conjugation invariant refiner. We will also show in Proposition 4.2.6 that we can compute the normaliser of a group with non-trivial disjoint direct product decomposition by computing the normaliser of each factor and solving the conjugacy problem for each pair of the factors.

4.2.1 Pruning with disjoint direct product decompositions

In this subsection, we see how the disjoint direct product decomposition gives us a means of proving that two groups are not conjugate in a symmetric group.

Lemma 4.2.1. *Let $K, L \leq S_n$ such that $L = K^g$ for some $g \in S_n$. If K_i is a finest disjoint direct factor of K, then K_i^g is a finest disjoint direct factor of L.*

Proof. We first show that K_i^g is a disjoint direct factor of L. Since K_i is a disjoint direct factor of K, there exists $J \leq K$ with support disjoint to $\mathit{Supp}(K_i)$ such that $K = K_i \times J$. Then $L = K_i^g \times J^g$. Since K_i^g and J^g have disjoint supports, this gives a disjoint direct product decomposition of L. Hence K_i^g is a disjoint direct factor of L.
It remains to show that K_i^g is d.d.p. indecomposable. Aiming for a contradiction, suppose that $K_i^g = J_1 \times J_2$ is a disjoint direct product decomposition. Then $K_i = J_1^{g^{-1}} \times J_2^{g^{-1}}$. Since J_1 and J_2 have disjoint supports, $J_1^{g^{-1}}$ and $J_2^{g^{-1}}$ also have disjoint

supports. Therefore this gives a disjoint direct product decomposition of K_i, which is a contradiction to the fact that K_i is a finest disjoint direct factor of K. □

Using the previous lemma, we get the following conjugacy invariant function and refiner:

Lemma 4.2.2. *Let N be the function with domain $\mathcal{S}(S_n)$ to $\mathbb{N}$ such that $N(K)$ is the number of orbits of K.*

1. *Let Ψ be the function with domain $\mathcal{S}(S_n)$ such that $\Psi(K)$ is the multiset*

   $$\{N(K_i) \mid \textit{for all finest disjoint direct factors } K_i \textit{ of } K\}.$$

   Then Ψ is a conjugacy invariant function.

2. *Let Φ be the function with domain $\mathcal{S}(S_n) \times \Omega$ defined by $\Phi(K, \alpha) = N(K_i)$, where K_i is the finest disjoint direct factor of K such that $\alpha \in Supp(K_i)$. Then Φ is a conjugacy invariant refiner.*

We denote Ψ and Φ by DDPDCIF *and* DDPDCIR *respectively.*

Proof. Let $K, L \leq S_n$ such that $L = K^g$ for some $g \in S_n$.
Part 1: Let K_i be a finest disjoint direct factor of K. Then by Lemma 4.2.1, K_i^g is a finest disjoint direct factor of L. Since conjugacy preserves orbit structures, the number of orbits of K_i is equal to the number of orbits of K_i^g. Therefore $\Psi(K) = \Psi(L)$.
Part 2: Let $\alpha \in \Omega$ and let K_i be a finest disjoint direct factor of K such that $\alpha \in$ *Supp*(K_i). Then $\alpha^g \in$ *Supp*(K_i^g) and by Lemma 4.2.1, K_i^g is a finest disjoint direct factor of L. As in the proof of Part 1, K_i and K_i^g have the same number of orbits, therefore $\Phi(K, \alpha) = \Phi(L, \alpha^g)$. □

We may obtain stronger conjugacy invariant functions and refiners by having $N(K)$ return the multiset $\{|\Delta| \mid \text{for all } K\text{-orbits } \Delta\}$. We can also make the conjugacy invariant refiner stronger by defining $\Phi(K, \alpha)$ to return a tuple $(N(K_i), s)$, where K_i is the finest disjoint direct factor of K such that $\alpha \in$ *Supp*(K_i) and s is the size of the K_i-orbit containing α. Additionally, we may use the orders of the disjoint direct product factors to obtain conjugacy invariant functions and refiners. None of these is included in our implementation.

4.2.2 Computing normalisers using disjoint direct product decompositions

In the last subsection, we showed that disjoint direct product decomposition can be used to prune and refine the search tree. In this subsection, we show that we can use the disjoint direct product decomposition of H to reduce the normaliser computation into polynomially many smaller problems. More specifically, we reduce the computation of

$N_{S_n}(H)$ into computing the normaliser of each factor and solving the conjugacy problem for each pair of factors.

For the rest of the subsection, let $H \leq S_n$ and let $H = H_1 \times H_2 \times \ldots \times H_r$ be the finest disjoint direct product decomposition of H. For $1 \leq i \leq r$, let $\Delta_i = \mathit{Supp}(H_i)$. We will consider each H_i as a subgroup of $\mathrm{Sym}(\Delta_i)$. Recall that we also identify $\mathrm{Sym}(\Delta_i)$ as a subgroup of $\mathrm{Sym}(\Gamma)$ for sets Γ containing Δ_i. Let $\Omega = \dot\bigcup_{i=1}^{r} \Delta_i$ be the support of H. We start by showing that certain elements of $N_{S_n}(H)$ give the conjugacy of certain pairs of disjoint direct factors of H, and vice versa.

Lemma 4.2.3. *Let $1 \leq i, j \leq r$ and let $\Delta = \Delta_i \dot\cup \Delta_j$. Then there exists $\nu \in N_{S_n}(H)$ such that $\Delta_i^\nu = \Delta_j$ if and only if H_i and H_j are conjugate in* $\mathrm{Sym}(\Delta)$.

Proof. $\Rightarrow$: Let $\nu \in N_{S_n}(H)$ such that $\Delta_i^\nu = \Delta_j$. Define g to be the involution in $\mathrm{Sym}(\Delta)$ such that $\alpha^g = \alpha^\nu$ for all $\alpha \in \Delta_i$. We will show that $H_i^g = H_j$.
Since $\mathit{Supp}(H_i) = \Delta_i$, we have $H_i^g = H_i^\nu$. By Lemma 4.2.1, the group H_i^ν is a finest disjoint direct factor of $H^\nu = H$ with support $\Delta_i^\nu = \Delta_j = \mathit{Supp}(H_j)$. Then by Proposition 3.2.2, the finest disjoint direct product decomposition of a group is unique, so $H_i^g = H_i^\nu \leq H_j$. Similarly since $(g^{-1})|_{\Delta_j} = (\nu^{-1})|_{\Delta_j}$, the group $H_j^{g^{-1}} = H_j^{\nu^{-1}}$ is a finest disjoint direct factor of $H^{\nu^{-1}} = H$ with support $\mathit{Supp}(H_i)$, so $H_j^{g^{-1}} \leq H_i$. Therefore $H_i^g = H_j$ and hence H_i and H_j are conjugate in $\mathrm{Sym}(\Delta)$.
$\Leftarrow$: Let $g \in \mathrm{Sym}(\Delta)$ such that $H_i^g = H_j$. Let $g' \in S_n$ be the involution with support Δ such that $\alpha^{g'} = \alpha^g$ for all $\alpha \in \Delta_i$. Then the conjugation by g' permutes H_i and H_j, and fixes all other H_s for $s \neq i, j$. Therefore $H^{g'} = H$. Hence g' is an element of $N_{S_n}(H)$ such that $\Delta_i^g = \Delta_j$. □

We gather the involutions witnessing the conjugacy of H_i and H_j in a group E. We will show in Proposition 4.2.6 that E and the $N_{\mathrm{Sym}(\Delta_i)}(H_i)$ generate $N_{S_n}(H)$.

Definition 4.2.4. Let $\sim_c$ be the equivalence relation on $H_1, H_2, \ldots, H_r$ such that $H_i \sim_c H_j$ if H_i and H_j are conjugate in $\mathrm{Sym}(\Delta_i \cup \Delta_j)$. Without loss of generality, let $[H_1]_{\sim_c}, [H_2]_{\sim_c}, \ldots, [H_t]_{\sim_c}$ be the equivalence classes of $\sim_c$.
For all $1 \leq i, j \leq r$ such that $H_i \sim_c H_j$, let c_{ij} be an involution in S_n with support $\Delta_i \cup \Delta_j$ such that $H_i^{c_{ij}} = H_j$. We take $c_{ii} = 1$ for all i.
For $1 \leq i \leq t$, let $E_i := \langle c_{ij} \mid H_j \in [H_i]_{\sim_c} \rangle$. Let $E := \langle E_i \mid 1 \leq i \leq t \rangle$.

We first show that the permutations of the disjoint factors H_i induced by conjugation by E form a direct product of symmetric groups.

Lemma 4.2.5. *Let E be as in Definition 4.2.4. Then for $\nu \in N_{S_n}(H)$, there exists $\epsilon \in E$ such that $H_i^\nu = H_i^\epsilon$ for all $1 \leq i \leq r$.*

Proof. Let $\sim_c$ be as in Definition 4.2.4. By the forward implication of Lemma 4.2.3, conjugation by ν permutes the finest disjoint direct factors. Therefore the permutation σ of the factors H_i induced by ν is an element of

$$D := \mathrm{Sym}([H_1]_{\sim_c}) \times \mathrm{Sym}([H_2]_{\sim_c}) \times \ldots \times \mathrm{Sym}([H_t]_{\sim_c}).$$

Let $1 \leq i \leq t$ and let E_i be as in Definition 4.2.4. Since $\mathrm{Sym}([H_i]_{\sim_c})$ is generated by $\{(H_i, H_j) \mid \text{for all } H_j \in [H_i]_{\sim_c}\}$ and each c_{ij} induces the permutation (H_i, H_j) of the disjoint factors, the permutations of the elements in $[H_i]_{\sim_c}$ induced by the conjugation of E_i form the symmetric group $\mathrm{Sym}([H_i]_{\sim_c})$.
As E is generated by the E_i, which have disjoint supports, the permutations of the factors H_i induced by the conjugation of E form the direct product D. Therefore, there exists $\epsilon \in E$ such that $H_i^\nu = H_i^\epsilon$ for all $1 \leq i \leq r$. □

Lastly we show that to compute $N_{S_n}(H)$, it suffices to compute $N_{\mathrm{Sym}(\Delta_i)}(H_i)$ for each $1 \leq i \leq r$ and the group E in Definition 4.2.4. We compute E by solving the conjugacy problem for disjoint direct factors H_i and H_j for all pairs $1 \leq i, j \leq r$.

Proposition 4.2.6. *Let $H \leq S_n$ and let $H = H_1 \times H_2 \times \ldots \times H_r$ be the finest disjoint direct product decomposition of H. For each $1 \leq i \leq r$, let $\Delta_i := Supp(H_i)$. Let E and t be as in Definition 4.2.4. Then*

$$N_{S_n}(H) = \langle N_{\mathrm{Sym}(\Delta_1)}(H_1) \times N_{\mathrm{Sym}(\Delta_2)}(H_2) \times \ldots \times N_{\mathrm{Sym}(\Delta_t)}(H_t), E\rangle,$$

where we identify the direct product as the corresponding subgroup of S_n.

Proof. $\geq$: As in the proof of the backwards implication of Lemma 4.2.3, each c_{ij} is in $N_{S_n}(H)$, so $E \leq N_{S_n}(H)$. Let $\nu \in N_{\mathrm{Sym}(\Delta_1)}(H_1) \times N_{\mathrm{Sym}(\Delta_2)}(H_2) \times \ldots \times N_{\mathrm{Sym}(\Delta_r)}(H_r)$. Then

$$H^\nu = H_1^\nu \times H_2^\nu \times \ldots \times H_r^\nu = H_1 \times H_2 \times \ldots \times H_r = H,$$

and so $\nu \in N_{S_n}(H)$.
$\leq$: Let $\nu \in N_{S_n}(H)$. Then by Lemma 4.2.5, there exists $\epsilon \in E$ such that $H_i^\nu = H_i^\epsilon$ for all $1 \leq i \leq r$. Since $\nu\epsilon^{-1}$ fixes each Δ_i setwise, we have $H_i^{(\nu\epsilon^{-1})|_{\Delta_i}} = H_i$ for all i. Therefore $(\nu\epsilon^{-1})|_{\Delta_i} \in N_{\mathrm{Sym}(\Delta_i)}(H_i)$. Hence $\nu\epsilon^{-1} \in N_{\mathrm{Sym}(\Delta_1)}(H_1) \times N_{\mathrm{Sym}(\Delta_2)}(H_2) \times \ldots \times N_{\mathrm{Sym}(\Delta_r)}(H_r)$.
It remains to show that $N_{\mathrm{Sym}(\Delta_1)}(H_1) \times N_{\mathrm{Sym}(\Delta_2)}(H_2) \times \ldots \times N_{\mathrm{Sym}(\Delta_r)}(H_r)$ is contained in $\langle N_{\mathrm{Sym}(\Delta_1)}(H_1) \times N_{\mathrm{Sym}(\Delta_2)}(H_2) \times \ldots \times N_{\mathrm{Sym}(\Delta_t)}(H_r), E\rangle$. We will show that $N_{\mathrm{Sym}(\Delta_j)}(H_j) \leq \langle N_{\mathrm{Sym}(\Delta_i)}(H_i), E_i\rangle$ for all $H_j \in [H_i]_{\sim_c}$. Let $h_i \in H_i$ and let $c_{ij} \in E_i$ be as in Definition 4.2.4. Then there exists $h_j \in H_j$ such that $h_i^{c_{ij}} = h_j$. Let $\nu_j \in N_{\mathrm{Sym}(\Delta_j)}(H_j)$. Then $h_i^{c_{ij}\nu_j c_{ij}} = h_j^{\nu_j c_{ij}} \in H_j^{c_{ij}} = H_i$. Therefore $c_{ij}\nu_j c_{ij} \in N_{\mathrm{Sym}(\Delta_i)}(H_i)$ and hence $\nu_j \in \langle N_{\mathrm{Sym}(\Delta_i)}(H_i), E_i\rangle$. □

4.3 Permutation isomorphism of projections

In this section, we first show how the permutation isomorphism classes of certain projections can be used to give conjugacy invariant functions and refiners for computing $N_{S_n}(H)$. We then define the class $\mathfrak{InP}(A)$ consisting of groups whose transitive constituents are all permutation isomorphic to A. Lastly in Proposition 4.3.10, we give the unique smallest subgroup of S_n containing $N_{S_n}(H)$ for all $H \in \mathfrak{InP}(A)$ with k orbits. Note that some results in this section generalise and describe some ideas presented in [Hul05, Section 11].

4.3.1 Pruning with permutation isomorphism of projections

Recall the definition of permutation isomorphism from Definition 1.1.5. By Proposition 1.1.7, permutation isomorphism is preserved under conjugation. In this subsection, we present some conjugacy invariant functions and refiners resulting from permutation isomorphism classes of certain projections.

We will start by considering the projections of groups on their orbits. Recall that we identify $\mathrm{Sym}(\Gamma)$ as a subgroup of $\mathrm{Sym}(\Delta)$ for all sets Γ and Δ such that $\Gamma \subset \Delta$.

Lemma 4.3.1. *Let K and L be subgroups of S_n such that $L = K^g$ for some $g \in S_n$. Let Ω_1 and Ω_2 be K-orbits. Then $K|_{\Omega_1}$ and $K|_{\Omega_2}$ are permutation isomorphic if and only if $L|_{\Omega_1^g}$ and $L|_{\Omega_2^g}$ are permutation isomorphic.*

Proof. By Proposition 1.1.7, $K|_{\Omega_1}$ and $K|_{\Omega_2}$ are permutation isomorphic if and only if there exists $\sigma \in \mathrm{Sym}(\Omega_1 \cup \Omega_2)$ such that $(K|_{\Omega_1})^\sigma = K|_{\Omega_2}$. Then

$$(L|_{\Omega_1^g})^{g^{-1}\sigma g} = (K|_{\Omega_1})^{gg^{-1}\sigma g} = (K|_{\Omega_1})^{\sigma g} = (K|_{\Omega_2})^g = L|_{\Omega_2^g}.$$

Hence by the backward implication of Proposition 1.1.7, $L|_{\Omega_1^g}$ and $L|_{\Omega_2^g}$ are permutation isomorphic. The converse follows similarly. □

So we may obtain a conjugacy invariant function and refiner using the permutation isomorphism classes of the transitive constituents of groups.

Lemma 4.3.2. *For $K \leq S_n$, let ψ_K be the function from the set of orbits of K to $\mathbb{N}$ such that $\psi_K(\Delta)$ is the number of K-orbits Δ' such that $K|_\Delta$ and $K|_{\Delta'}$ are permutation isomorphic.*

1. *Let Ψ be the function with domain $\mathcal{S}(S_n)$ such that $\Psi(K)$ is the multiset*

$$\{\psi_K(\Delta) \mid \textit{for all orbits } \Delta \textit{ of } K\}.$$

Then Ψ is a conjugacy invariant function.

2. *Let Φ be the function with domain $\mathcal{S}(S_n) \times \Omega$ such that $\Psi(K, \alpha) = \psi_K(\Delta)$ where Δ is the K-orbit containing α. Then Φ is a conjugacy invariant refiner.*

Proof. Let $K, L \leq S_n$ such that $L = K^g$ for some $g \in S_n$.
Part 1: Let Δ be a K-orbit. Then Δ^g is a L-orbit. We shall show that $\psi_K(\Delta) = \psi_L(\Delta^g)$, from which the result follows.
Let Δ' be a K-orbit. Then by Lemma 4.3.1, $K|_\Delta$ is permutation isomorphic to $K|_{\Delta'}$ if and only if $L|_{\Delta^g}$ is permutation isomorphic to $L|_{\Delta'^g}$. Thus $\psi_K(\Delta) = \psi_L(\Delta^g)$.

Part 2: Let $\alpha \in \{1, 2, \ldots, n\}$ and let Δ be the K-orbit containing α. Then Δ^g is the L-orbit containing α^g. By the arguments above, $\psi_K(\Delta) = \psi_L(\Delta^g)$ and so $\Psi(K, \alpha) = \Psi(L, \alpha^g)$. □

Note that there is no known efficient algorithm to test permutation isomorphisms. In practice, if the orbits are small, the GAP function TransitiveIdentification is quick and can be used for permutation isomorphism testing. For a transitive group G, the image TransitiveIdentification(G) returns an integer i such that G is permutation isomorphic to the i-th group in the transitive group library with degree $|Supp(G)|$ [Hul17]. Hence two transitive permutation groups H and G are permutation isomorphic if and only if $|Supp(H)| = |Supp(G)|$ and TransitiveIdentification(H) = TransitiveIdentification(G). However, note that TransitiveIdentification is based on existing classifications of transitive groups, and so it is limited by the degree of the input group.

Therefore the following result follows from Lemma 4.3.2.

Corollary 4.3.3. *Let Ψ be the function with domain $\mathcal{S}(S_n)$ where $\Psi(K)$ is the multiset of tuples*

$$\{(|\Gamma|, \textsc{TransitiveIdentification}(K|_\Gamma)) \mid \textit{for all orbits } \Gamma \textit{ of } K\}.$$

Then Ψ is a conjugacy invariant function.

For larger orbits, this method is not feasible. Since the order of two permutation isomorphic groups is the same, we give the following conjugacy invariant function and refiner. Note that the following result is elementary and it need not be proved via permutation isomorphisms.

Corollary 4.3.4. *For all subgroups K of S_n, let ψ_K be the function from the set of orbits of K to $\mathbb{N}$ such that $\psi_K(\Delta)$ is the number of K-orbits Δ' such that $|\Delta| = |\Delta'|$ and $|(K|_\Delta)| = |(K|_{\Delta'})|$. Let Ψ and Φ be as constructed in Lemma 4.3.2. Then Ψ is a conjugacy invariant function and Φ is a conjugacy invariant refiner.*

In this thesis, we will mostly consider groups with pairwise permutation isomorphic transitive constituents. So the conjugacy invariant functions and refiners above are of no use to us. We will instead consider certain stabilisers of the projections onto the unions of pairs of orbits to obtain a conjugacy invariant refiner.

Lemma 4.3.5. *Let Ψ be as in Corollary 4.3.3. Let Φ be the function with domain $\mathcal{S}(S_n) \times \Omega$ such that $\Phi(K, \alpha)$ gives the multiset of Ψ-images of pointwise stabilisers $\{\Psi((K|_{\Omega_i \cup \Omega_j})_{(\Omega_j)}) \mid 1 \le j \le k, j \ne i\}$ if $\alpha \in \Omega_i$, where $\Omega_1, \Omega_2, \ldots, \Omega_k$ are orbits of K. Then Φ is a conjugacy invariant refiner.*
We will denote Φ by ProjOnOrbitPairsCIR.

Proof. Let $K, L \leq S_n$ such that $L = K^g$ for some $g \in S_n$. Let $\Omega_1, \Omega_2, \ldots, \Omega_k$ be the orbits of K. Let $\alpha \in \Omega$ and let $1 \leq i \leq k$ such that $\alpha \in \Omega_i$. Let $1 \leq j \leq k$ such that $j \neq i$. Then $((K|_{\Omega_i \cup \Omega_j})_{(\Omega_j)})^g = (L|_{\Omega_i^g \cup \Omega_j^g})_{(\Omega_j^g)}$, and so by Corollary 4.3.3, we have $\Psi((K|_{\Omega_i \cup \Omega_j})_{(\Omega_j)}) = \Psi((L|_{\Omega_i^g \cup \Omega_j^g})_{(\Omega_j^g)})$.
Therefore the multisets

$$\{\Psi((K|_{\Omega_i \cup \Omega_j})_{(\Omega_j)}) \mid 1 \leq j \leq k, j \neq i\} \text{ and } \{\Psi((L|_{\Omega_i^g \cup \Omega_j^g})_{(\Omega_j^g)}) \mid 1 \leq j \leq k, j \neq i\}$$

are equal. Since Ω_i^g is the orbit of L containing α^g, the multisets above are $\Phi(K, \alpha)$ and $\Phi(L, \alpha^g)$ respectively so $\Phi(K, \alpha) = \Phi(L, \alpha^g)$. □

Note that we may also consider the projection on the unions of more than two orbits. However this is not implemented.

4.3.2 Normalisers of H in $\mathfrak{InP}(A)$

By Lemma 4.3.1, the normaliser $N_{S_n}(H)$ permutes the permutation isomorphic transitive constituents of H. In this subsection we introduce the class $\mathfrak{InP}(A)$ for transitive group $A \leq S_m$, which consists of all groups whose transitive constituents are all permutation isomorphic to A. We will also give a subgroup of S_n containing $N_{S_n}(H)$ for $H \in \mathfrak{InP}(A)$. Such a subgroup is used for computing normalisers in [Hul05, Section 11]. Here, we will give more details of the construction of such a subgroup.

Definition 4.3.6. Let $A \leq S_m$ be transitive. Let $\mathfrak{InP}(A)$ be the class consisting of all groups $H \leq S_n$ with k orbits $\Omega_1, \Omega_2, \ldots, \Omega_k$ such that each projection $H|_{\Omega_i}$ is permutation isomorphic to A. Notice that $n = mk$.

An alternative way of seeing the groups in $\mathfrak{InP}(A)$ is as subdirect products of A^k. For the rest of the section, we will use the following notation.

Notation 4.3.7. Let $H \in \mathfrak{InP}(A)$ be a subgroup of S_n. For $1 \leq i \leq k$, let $G_i = H|_{\Omega_i}$. For $2 \leq i \leq k$, let $\phi_i : \Omega_1 \to \Omega_i$ be a bijection witnessing the permutation isomorphism from G_1 to G_i. So G_i is permutation isomorphic to A for all i.
Let $G = G_1 \times G_2 \times \ldots \times G_k$, identified as a subgroup of S_n. Then H is a subdirect product of G.

Next we define a subgroup $L \leq S_n$ and analyse its structure in a technical lemma. We will use this lemma later to show that L contains $N_{S_n}(H)$. Recall that for all $\Gamma \subset \Delta$, we identify $\mathrm{Sym}(\Gamma)$ as a subgroup of $\mathrm{Sym}(\Delta)$ using the natural inclusion.

Lemma 4.3.8. *Let $H \in \mathfrak{InP}(A)$ with the notation from Notation 4.3.7, for $1 \leq i \leq k$, let $N_i := N_{\mathrm{Sym}(\Omega_i)}(G_i)$. For each $2 \leq i \leq k$, let $\overline{\phi_i}$ be the involution in S_n with support $\Omega_i \cup \Omega_j$ such that $\phi_i(\alpha) = \alpha^{\overline{\phi_i}}$ for all $\alpha \in \Omega_i$. Let $B := \langle N_1, N_2, \ldots, N_k \rangle \leq S_n$ and let $K := \langle \overline{\phi_j} \mid 2 \leq j \leq k \rangle \leq S_n$. Let $L := \langle B, K \rangle$.*

1. *Let $\xi : K \to S_k$ be the permutation representation of K on $\{\Omega_1, \Omega_2, \ldots, \Omega_k\}$. Then ξ is faithful and $K \cong S_k$.*

2. *L is permutation isomorphic to $N_1 \wr S_k$ in its imprimitive action.*

3. *$N_{S_n}(G) = L$.*

Proof. Part 1: The mapping ξ is a homomorphism since for all $\kappa_1, \kappa_2 \in K$ and $1 \leq i \leq k$,

$$\Omega_{i\xi(\kappa_1)\xi(\kappa_2)} = (\Omega_{i\xi(\kappa_1)})^{\kappa_2} = (\Omega_i^{\kappa_1})^{\kappa_2} = \Omega_i^{\kappa_1\kappa_2} = \Omega_{i\xi(\kappa_1\kappa_2)}.$$

For injectivity, let $\kappa \in K$ such that $\xi(\kappa) = 1$. Then $\Omega_i^{\kappa} = \Omega_i$, for all $1 \leq i \leq k$. We first show that κ fixes Ω_1 pointwise. Let $\alpha \in \Omega_1$. As K is generated by the $\overline{\phi_i}$, we can write κ as a product of the $\overline{\phi_i}$. Since for each $2 \leq i \leq k$, the orbit Ω_i is moved only by $\overline{\phi_i}$, there exist (not neccessarily distinct) $I_1, I_2, \ldots, I_r$ such that $\alpha^{\kappa} = \alpha^{\overline{\phi_{I_1}}^2 \overline{\phi_{I_2}}^2 \cdots \overline{\phi_{I_r}}^s} = \alpha^{\overline{\phi_{I_r}}^s}$. Then as $\alpha^{\kappa} \in \Omega_1$, the integer s is even. So $\alpha^{\kappa} = \alpha$.
Let $\alpha \in \Omega_i$ for $i \neq 1$. Then there exists $\beta \in \Omega_1$ such that $\alpha^{\overline{\phi_i}} = \beta$. Since $\overline{\phi_i}$ is an involution, we have $\alpha^{\kappa} = \beta^{\kappa\overline{\phi_i}} = \beta^{\overline{\phi_i}} = \alpha$. So κ also fixes Ω_i pointwise. Hence ξ is injective and is therefore faithful.
The mapping ξ is surjective since $\xi(\overline{\phi_i}) = (1, i)$ for all $2 \leq i \leq k$, and $\{(1, i) \mid 2 \leq i \leq k\}$ generates S_k. Therefore $K \cong S_k$.
Part 2: We first show that the Ω_i form a block system for L. Since N_i fixes Ω_i setwise and K permutes the Ω_i, it suffices to show that L is transitive. Let $\alpha, \beta \in \Omega$ and let $1 \leq i, j \leq k$ such that $\alpha \in \Omega_i$ and $\beta \in \Omega_j$. Then $\alpha^{\overline{\phi_i}}, \beta^{\overline{\phi_j}} \in \Omega_1$, therefore since A is assumed to be transitive, there exists some $g \in G$ such that $\alpha^{\overline{\phi_i}g} = \beta^{\overline{\phi_j}}$. So $\overline{\phi_i}g\overline{\phi_j} \in \langle G, K\rangle \leq L$ maps α to β, therefore L is transitive.
Consider the kernel J of the action of L on these blocks. Since K acts faithfully on the blocks, $J \leq \langle N_1, N_2, \ldots, N_k\rangle$. Conversely, since each N_i fixes the blocks, the group $\langle N_1, N_2, \ldots, N_k\rangle \leq J$. So $J = \langle N_1, N_2, \ldots, N_k\rangle \cong N_1 \times N_2 \times \ldots \times N_k$. Hence, up to isomorphism, $L \leq (N_1 \times N_2 \times \ldots \times N_k) \rtimes S_k$. But as $S_k \cong K \leq L$, it follows that $L = (N_1 \times N_2 \times \ldots \times N_k) \rtimes S_k \cong N_1 \wr S_k$.
Part 3: We first show that $G \trianglelefteq L$. Since $G_i \leq N_{\mathrm{Sym}(\Omega_i)}(G_i)$ for all $1 \leq i \leq k$, we have $G \leq L$. To show that G is normal in L, let $g \in G$. Then we can write g as the product of $g_1, g_2, \ldots, g_k$, where each $g_i \in G_i$. Let $1 \leq i \leq k$ and let $n_i \in N_i$. Then there exists $g_i' \in G_i$ such that $g_i' = g_i^{n_i}$. Since $n_i|_{\Omega_j} = 1$ for all other $j \neq i$, we have $g^{n_i} = g_1 \ldots g_{i-1}g_i'g_{i+1} \ldots g_k \in G_1 \times G_2 \times \ldots \times G_k = G$, and so $G^{n_i} = G$.
Let $2 \leq i \leq k$. We now show that $G^{\overline{\phi_i}} = G$. Since ϕ_i is a bijection witnessing the permutation isomorphism between G_1 and G_i, as in the proof of Proposition 1.1.7, $G_1^{\overline{\phi_i}} = G_i$. So there exists $g_1' \in G_1$ and $g_i' \in G_i$ such that $g_i' = g_1^{\overline{\phi_i}}$ and $g_1' = g_i^{\overline{\phi_i}}$. Since $\overline{\phi_i}$ fixes all points outside $\Omega_1 \cup \Omega_i$, we have that

$$g^{\overline{\phi_i}} = g_1^{\overline{\phi_i}} g_2^{\overline{\phi_i}} \ldots g_k^{\overline{\phi_i}} = g_1' g_2 \ldots g_{i-1} g_i' g_{i+1} \ldots g_k \in G_1 \times G_2 \times \ldots \times G_k = G.$$

Since the n_i and the $\overline{\phi_i}$ generate L, we have $G \trianglelefteq L$.
Lastly we show that $N_{S_n}(G) = L$. $\geq$: Since $G \trianglelefteq L \leq S_n$ and $N_{S_n}(G)$ is the largest subgroup of S_n in which G is normal, we have $L \leq N_{S_n}(G)$.
$\leq$: First observe that since $L \leq N_{S_n}(G)$, the normaliser $N_{S_n}(G)$ is transitive. Then as $G \trianglelefteq N_{S_n}(G)$, by Proposition 1.1.14, the orbits $\Omega_1, \Omega_2, \dots, \Omega_k$ of G form a system of blocks of $N_{S_n}(G)$. Let $g \in G$, and let $\nu \in N_{S_n}(G)$. Then ν permutes the blocks $\Omega_1, \Omega_2, \dots, \Omega_k$. Since the induced action of K on the Ω_i is isomorphic to S_k, there exists $\kappa \in K$ such that $\Omega_i^\nu = \Omega_i^\kappa$ for all $1 \leq i \leq k$. Then $\nu\kappa^{-1}$ fixes each Ω_i setwise, and so $(\nu\kappa^{-1})|_{\Omega_i} \in N_i$ for $1 \leq i \leq k$. Hence $\nu\kappa^{-1} \in \langle N_1, N_2, \dots, N_k \rangle = B$. Therefore $\nu \in \langle B, K \rangle = L$. □

Let $H \in \mathfrak{InP}(A)$. We will show that L contains $N_{S_n}(H)$ in Proposition 4.3.10, which is the main result of this subsection.

Lemma 4.3.9. *Let T_1,T_2,…,T_k be transitive groups with pairwise disjoint supports. Let S be the symmetric group on the disjoint union of the supports of the T_i. Let $T := T_1 \times T_2 \times \ldots \times T_k$, identified as a subgroup of S. Let A be a subdirect product of T and let C be a group such that $T \leq C \leq S$. Then $N_C(A) \leq N_C(T)$.*

Proof. Let $\nu \in N_C(A)$ and $t = (1, \dots, 1, t_i, 1, \dots, 1) \in T$ for some $1 \leq i \leq k$ and $t_i \in T_i$. We show that $t^\nu \in T$. Then as

$$\{(1, \dots, 1, t_i, 1, \dots, 1) \mid \text{for all } t_i \in T_i \text{ and } 1 \leq i \leq k\}$$

generate T, we have $\nu \in N_C(T)$.
Since A is a subdirect product of T, the projection of A onto T_i is T_i, so there exists $(a_1, a_2, \dots, a_k) \in A$ such that $a_i = t_i$. Since $N_C(A)$ permutes the orbits of A, there exists $1 \leq j \leq k$ such that $a_i^\nu \in T_j$. Then

$$t^\nu = (1, \dots, 1, a_i^\nu, 1 \dots, 1) \in 1 \times \ldots \times 1 \times T_j \times 1 \times \ldots \times 1 \leq T.$$

□

Proposition 4.3.10. *Let H be as in Notation 4.3.7 and let L be as in Lemma 4.3.8. Then $N_{S_n}(H) \leq L$.*

Proof. Let G be as in Notation 4.3.7. As H is a subdirect product of G, by Lemma 4.3.9, $N_{S_n}(H) \leq N_{S_n}(G)$. Then by Part 3 of Lemma 4.3.8, $N_{S_n}(H) \leq L$. □

Lastly, since $L \cong N_1 \wr S_k$, it is useful to think of a normalising element $g \in N_{S_n}(H)$ as an element of the wreath product.

Lemma 4.3.11. *Let $H = \langle X \rangle$ be as in Notation 4.3.7. Assume that we have B, K and ξ as in Lemma 4.3.8. Let $g \in N_{S_n}(H)$. Then in polynomial time, we can compute $\nu \in B$ and $\kappa \in K$ such that $g = \nu\kappa$.*

Proof. By Proposition 4.3.10, we have $g \in L$, so g induces a permutation of the orbits $\Omega_1, \Omega_2, \ldots, \Omega_k$ of H. Let $w \in S_k$ such that $\Omega_i^g = \Omega_{i^w}$ for $1 \leq i \leq k$. Let ξ be as in Lemma 4.3.8. Then $\nu := g\xi^{-1}(w)$ fixes each H-orbits, so $\nu \in B$. By Part 9 of Theorem 2.1.9, $\xi^{-1}(w) \in K$ can be computed in polynomial time. □

4.4 Algorithm

Let $H \leq S_n$ be in class $\mathfrak{InP}(A)$ for some transitive group $A \leq S_m$. Using some of the conjugacy invariant functions and refiners defined in this chapter, we compute a subgroup S of S_n containing $N_{S_n}(H)$, then $N_{S_n}(H) = N_S(H)$. We will demonstrate that such an S can be useful for reducing the computation time for $N_{S_n}(H)$ in Section 4.5.

We start by showing how we can use a conjugacy invariant function to construct a partition preserved by $N_{S_n}(H)$.

Lemma 4.4.1. *Let $H \in \mathfrak{InP}(A)$ have orbits $\Omega_1, \Omega_2, \ldots, \Omega_k$. Let Ψ be a conjugacy invariant function. Let P be a partition of $\{\Omega_1, \Omega_2, \ldots, \Omega_k\}$ such that Ω_i and Ω_j are in the same cell if and only if $\Psi(H_{(\Omega_i)}) = \Psi(H_{(\Omega_j)})$. Then $N_{S_n}(H)$ stabilises the partition P.*

Proof. Let Ω_i and Ω_j be orbits of H in different cells of P. Then $\Psi(H_{(\Omega_i)}) \neq \Psi(H_{(\Omega_j)})$. By the definition of a conjugacy invariant function, $\Psi(H_{(\Omega_i)})$ and $\Psi(H_{(\Omega_j)})$ are not conjugate in S_n. Then by Lemma 2.4.9 there are no $\nu \in N_{S_n}(H)$ such that $\Omega_i^\nu = \Omega_j$. □

Similarly, we can obtain a partition stabilised by $N_{S_n}(H)$ using a conjugacy invariant refiner:

Lemma 4.4.2. *Let $H \in \mathfrak{InP}(A)$, with $H \leq S_n$. Let Φ be a conjugacy invariant refiner. Let P be a partition of $\{1, 2, \ldots, n\}$ such that α and β are in the same cell if and only if $\Phi(H, \alpha) = \Phi(H, \beta)$. Then $N_{S_n}(H)$ stabilises partition P.*

Proof. Let α and β be points in $\{1, 2, \ldots, n\}$ in different cells of P. Then we have $\Phi(H, \alpha) \neq \Phi(H, \beta)$. By the definition of a conjugacy invariant refiner, there are no $\nu \in S_n$ such that $H^\nu = H$ and $\alpha^\nu = \beta$. □

Let Φ be the conjugacy invariant refiner ProjOnOrbitPairsCIR (Lemma 4.3.5), EquivOrbsCIR (Part 2 of Lemma 4.1.2) or DdpdCIR (Part 2 of Lemma 4.2.2). We show that we can in fact obtain a partition of the H-orbits.

Lemma 4.4.3. *Let Φ be the conjugacy invariant refiner* ProjOnOrbitPairsCIR, EquivOrbsCIR *or* DdpdCIR. *Let $K \leq S_n$. If α and β are in the same K-orbit, then $\Phi(K, \alpha) = \Phi(K, \beta)$.*

Proof. If Φ is the conjugacy invariant refiner EquivOrbsCIR or ProjOnOrbit-PairsCIR, then the results follows from the definition of Φ. Else let Φ be as in DdpdCIR. If α and β are in the same K-orbit, then they are moved by the same disjoint direct factor of K. So $\Phi(K, \alpha) = \Phi(K, \beta)$. □

Corollary 4.4.4. *Let $H \leq S_n$ with orbits $\Omega_1, \Omega_2, \ldots, \Omega_k$. Let Φ be conjugacy invariant refiner* EquivOrbsCIR*,* DdpdCIR *or* ProjOnOrbitPairsCIR*. For $1 \leq i \leq k$, fix a point $\alpha_i \in \Omega_i$. Let P be a partition of $\{\Omega_1, \Omega_2, \ldots, \Omega_k\}$ such that Ω_i and Ω_j are in the same cell if and only if $\Phi(H, \alpha_i) = \Phi(H, \alpha_j)$. Then $N_{S_n}(H)$ stabilises partition P.*

Now let P be a partition of the H-orbits $\Omega_1, \Omega_2, \ldots, \Omega_k$ stabilised by $N_{S_n}(H)$. Let $L \cong N_{\mathrm{Sym}(\Omega_1)}(H|_{\Omega_1}) \wr S_k$ be as in Lemma 4.3.8. We will compute a subgroup of L stabilising P which contains $N_{S_n}(H)$.

Lemma 4.4.5. *Let P be a partition of the H-orbits $\Omega_1, \Omega_2, \ldots, \Omega_k$ stabilised by $N_{S_n}(H)$. Let P' be the partition of $\{1, 2, \ldots k\}$ such that $1 \leq i, j \leq k$ are in the same cell of P' if and only if Ω_i and Ω_j are in the same cell of P. Let $U \leq S_k$ be stabiliser of P' in S_k. Let ξ and B be as in Lemma 4.3.8, and let $S = \langle B, \xi^{-1}(U)\rangle$. Then $N_{S_n}(H) \leq S$.*

Proof. Let $\nu \in N_{S_n}(H)$. Then by Proposition 4.3.10, the normaliser $N_{S_n}(H)$ is contained in $L = \langle B, \xi^{-1}(S_k)\rangle$. Since $N_{S_n}(H)$ permutes the H-orbits, the element ν induces a permutation $\sigma \in S_k$ on the Ω_i, where $\Omega_i^\nu = \Omega_{i^\sigma}$. Then as $N_{S_n}(H)$ stabilises P, the permutation σ stabilises P', therefore $\sigma \in U$.
Let $s := \xi^{-1}(\sigma) \in S$. Since s and ν induces the same permutation σ on the H-orbits, we have νs^{-1} fixes each Ω_i setwise. Then for each $1 \leq i \leq k$, the projection $(\nu s^{-1})|_{\Omega_i}$ normalises $H|_{\Omega_i}$, so $\nu s^{-1} \in B \leq S$. Therefore $\nu \in S$. □

Hence, given a group $H = \langle X\rangle \leq S_n$ in class $\mathfrak{InP}(A)$ for some transitive group $A \leq S_m$, we may compute the normaliser $N_{S_n}(H)$ the following way:

1. Fix an labelling of the H-orbits $\Omega_1, \Omega_2, \ldots, \Omega_k$.
2. Compute $L \cong N_{S_m}(A) \wr S_k$ as in Lemma 4.3.8.
3. Using Corollary 4.4.4, compute partitions P_{g0}, P_{e0} and P_{d0} using the conjugacy invariant refiners ProjOnOrbitPairsCIR (Lemma 4.3.5), EquivOrbsCIR (Part 2 of Lemma 4.1.2) and DdpdCIR (Part 2 of Lemma 4.2.2) respectively.
4. Using Lemma 4.4.1, compute partitions P_{e1} and P_{d1} using the conjugacy invariant functions EquivOrbsCIF (Part 1 of Lemma 4.1.2) and DdpdCIF (Part 1 of Lemma 4.2.2) respectively.
5. Compute the meet P of the partitions $P_{e0}, P_{e1}, P_{d0}, P_{d1}$ and P_{g0}.
6. Compute the group S stabilising P as in Lemma 4.4.5, so $N_{S_n}(H) \leq S$.
7. Compute $N_S(H)$.

4.5 Results and discussion

In this section, we compare the performance of the algorithm in Section 4.4 against the generic normaliser computation function in GAP.

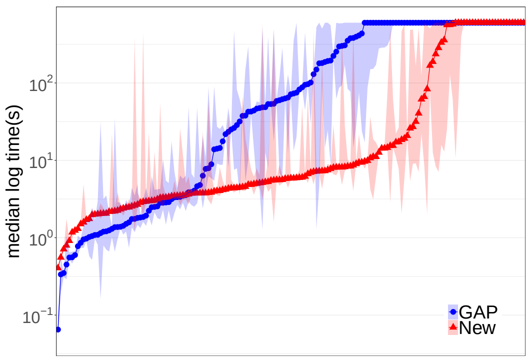
We consider groups in class $\mathfrak{InP}(A)$ for all transitive group A of degree between 2 and 10 inclusive. For each such $A \leq S_m$, we consider 10 random groups $H \leq S_n$ in $\mathfrak{InP}(A)$ with 8 orbits and 10 random groups $H \leq S_n$ in $\mathfrak{InP}(A)$ with 10 orbits. Each of these H with $k \in \{8, 10\}$ orbits is generated in the following way. Let $G \leq S_{mk}$ be the group in $\mathfrak{InP}(A)$ with orbits $\Omega_1, \Omega_2, \ldots, \Omega_k$ such that $G = G|_{\Omega_1} \times G|_{\Omega_1} \times \ldots \times G|_{\Omega_k}$, identified as a subgroup of S_{mk}. Choose a random integer s between 1 and k, and generate a set S of random elements of G of size s. While $\langle S \rangle$ is not in $\mathfrak{InP}(A)$, we add more random elements of G to S. Lastly we set $H = \langle S \rangle$.

For each $H \leq S_n$, we compute its normaliser $N_{S_n}(H)$ using both the NORMALIZER function in GAP and the algorithm in Section 4.4, with a 10 minutes timeout. The result is presented in Figure 4.1. For each method, we give the median, lower quartile and upper quartile computation time for each $H \in \mathfrak{InP}(A)$ with k orbits. The results are ordered independently by increasing median time for GAP and the new algorithm. For each degree of A, we also report the ratio of the completed instances over the total number of instances. Finally, we also give the scatter heatplots of the computation of each $N_{S_n}(H)$ using the two algorithms.

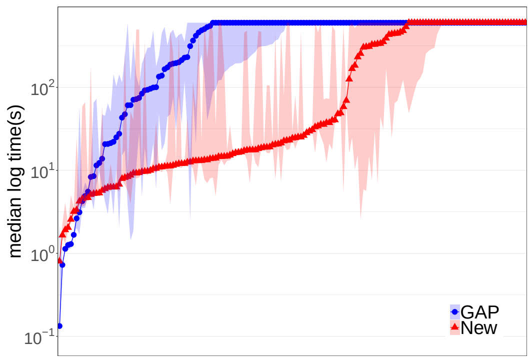
Results show that the number of instances exceeding the timeout significantly decreases when using the new algorithm. Of the instances where the GAP function is fast ($\leq 10s$), the new algorithm may be slower but remains quite fast. For all instances where GAP requires more than $100s$, the new algorithm always performs better. In particular, there are many instances where GAP exceeds the time limits but the time taken by the new algorithm remains quite low.

However, we have only experimented on groups with 8 or 10 orbits. It is unclear if the initial computation of the overgroup of $N_{S_n}(H)$ will payoff for groups with a larger number of orbits. On the other hand, we could use the conjugation invariant functions and refiners during the search, as described in Section 2.4.2, instead of using them to compute an overgroup of $N_{S_n}(H)$. We also note that there could be more sophisticated methods for constructing the subdirect products H which are not implemented here.

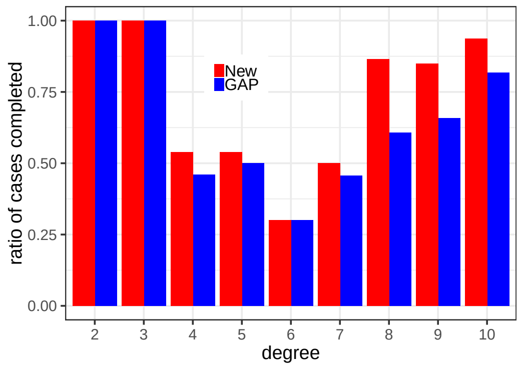
For future work, it would be interesting to directly compare the effectiveness of these conjugacy invariant functions and refiners against each other, and against the methods implemented in GAP, such as those using orbital graphs. It would also be useful to determine and characterise the properties of the intransitive groups H that give slow normaliser computation, to obtain better global pruning methods.



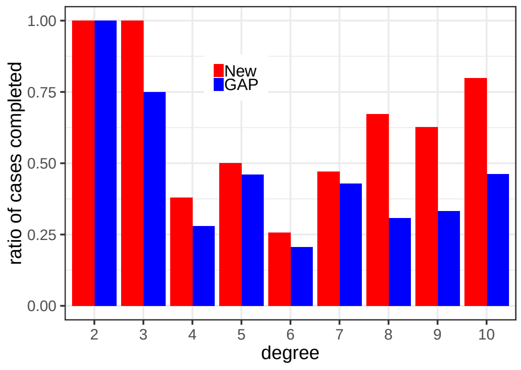
(a) Median, lower and upper quartile log computation time (*logs*) for each G for $k = 8$.



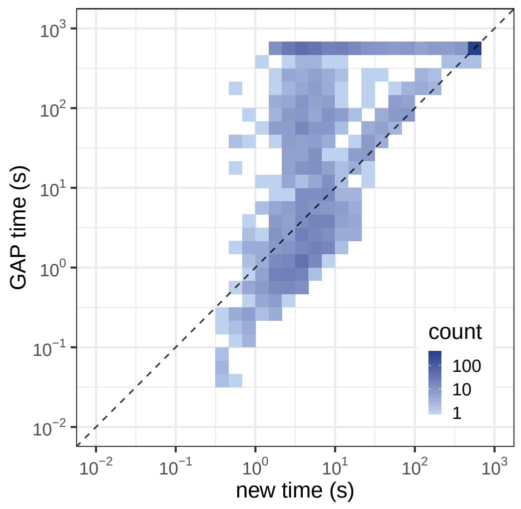
(b) Median, lower and upper quartile log computation time (*logs*) for each G for $k = 10$.



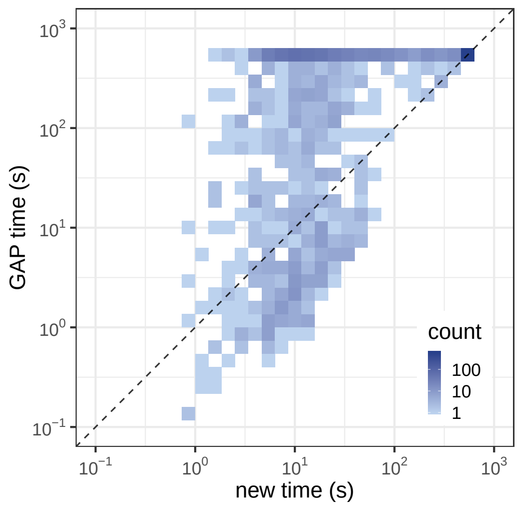
(c) Ratio of completed cases for each degree for $k = 8$.



(d) Ratio of completed cases for each degree for $k = 10$.



(e) Scatter heatplot of all instances for $k = 8$.



(f) Scatter heatplot of all instances for $k = 10$.

Figure 4.1: Computation time for computing normalisers of 10 random groups with k orbits in class $\mathfrak{InP}(G)$, for each transitive group G of degree between 2 and 10, with 10 minutes timeout.

Chapter 5

Normalisers of Groups In Class $\mathfrak{InP}(C_p)$

In this chapter we will compute $N_{S_n}(H)$ for $H \in \mathfrak{InP}(C_p)$, where C_p is the cyclic group on p points. We believe the class $\mathfrak{InP}(C_p)$ is likely to be one of the hardest cases of the Norm-Sym problem, as current methods are very slow. Indeed, the implementation in the computer algebra system GAP [GAP20] struggles to solve the Norm-Sym problem for groups in $\mathfrak{InP}(C_p)$ even when H has a small degree (median time of more than 10 minutes for degree less than 30 and $p = 2$, see Section 5.6).

In terms of complexity, the decision variant of Norm-Sym, that is, deciding whether two groups $H, K \in \mathfrak{InP}(C_p)$ are conjugate in S_n, is polynomially equivalent to deciding whether two codes over $\mathbb{F}_p$ are monomially equivalent (see Definition 5.1.12, Theorem 5.1.15). This is known to be at least as hard as graph isomorphism [PR97]. By [SS13, Theorem 1], the monomial equivalence of codes of length k over $\mathbb{F}_p$ reduces to computing the permutation equivalence of codes of length $(p-1)k$ over $\mathbb{F}_p$. By Babai's simply exponential bound on the permutation equivalence of codes [BCGQ11], deciding the monomial equivalence of codes of length k over $\mathbb{F}_p$ is bounded by $2^{O((p-1)k)}$, assuming constant time field operations. However, it is not known whether the conjugacy problem in symmetric groups is polynomial-time equivalent to Norm-Sym [Luk93]. This chapter will focus on developing an effective practical algorithm to solve Norm-Sym for groups in $\mathfrak{InP}(C_p)$.

In this chapter, using methods for computing automorphisms of linear codes, we give a new faster algorithm to solve the Norm-Sym problem for $H \in \mathfrak{InP}(C_p)$. In Section 5.6, we provide evidence that our new algorithm performs far better than the one currently implemented in GAP. As the simply exponential algorithm in [Wie19] is not implemented, the fastest implemented algorithm to compute $N_{S_n}(H)$ is a backtrack search algorithm, and so has a runtime of $2^{O(n \log n)}$. Using methods based on [Feu09], we shall bound the complexity of the Norm-Sym problem for $H \in \mathfrak{InP}(C_p)$ by $2^{O((n/p) \log (n/p))}$. This gives a better complexity for the case when p is large, but we will show our algorithm is better in practice for all values of p. Using dual codes (see Definition 5.2.1), we may bound the depth of the search tree in S_n by $n/(2p)$, so we get the following result, which will follow immediately from Proposition 5.2.3 and Theorem 5.2.22.

Theorem 5.0.1. *The* Norm-Sym *problem for H in class $\mathfrak{InP}(C_p)$ can be solved in time* $\min\left(2^{O(\frac{n}{p}\log\frac{n}{p})}, 2^{O(\frac{n}{2p}\log n)}\right)$.

The chapter structure is as follows. In Section 5.1, we show that the Norm-Sym problem for $H \in \mathfrak{InP}(C_p)$ is polynomially equivalent to the problem of computing a certain group of automorphisms of a code over $\mathbb{F}_p$. In Section 5.2, we shall prove Theorem 5.0.1. In Section 5.3, we present several pruning techniques for computing $N_{S_n}(H)$ for $H \in \mathfrak{InP}(C_p)$. We will describe our algorithm to compute $N_{S_n}(H)$ for $H \in \mathfrak{InP}(C_p)$ in Section 5.4. In Section 5.5, we shall see how the method can be extended to compute the normaliser $N_{S_n}(H)$ for $H \in \mathfrak{InP}(D_{2p})$, the class of groups with intransitive projections permutation isomorphic to dihedral groups of order $2p$, where p is an odd prime. Finally, in Section 5.6, we will give the runtimes of our implementations and the one in GAP.

5.1 Normaliser of $H \in \mathfrak{InP}(C_p)$ and code automorphisms

Let p be prime. We will be computing $N_{S_n}(H)$ for H in class $\mathfrak{InP}(C_p)$. First we show that we can decide if a given group $H \leq S_n$ is in the class $\mathfrak{InP}(C_p)$.

Lemma 5.1.1. *Let $H = \langle X\rangle \leq S_n$. Then in polynomial time, we can decide if $H \in \mathfrak{InP}(C_p)$.*

Proof. Let $\Omega_1, \Omega_2, \ldots, \Omega_k$ be the orbits of H. Then $H \in \mathfrak{InP}(C_p)$ if and only if $|\Omega_i| = p$ and $H|_{\Omega_i}$ has order p for all $1 \leq i \leq k$. Hence the result follows from Part 1 of Theorem 2.1.9. □

For the rest of this section, we shall assume the following.

Definition 5.1.2. Let $n = pk$ and let $H \leq S_n$ be in class $\mathfrak{InP}(C_p)$. Let $\Omega_1, \Omega_2, \ldots, \Omega_k$ be the orbits of H, where the orbits are ordered such that $|H| = p^s = |(H|_{\cup_{i\leq s}\Omega_i})|$. For $1 \leq i \leq k$, let $G_i := H|_{\Omega_i}$. Let $G := G_1 \times G_2 \times \ldots \times G_k \leq S_n$, called the *enveloping group* of H.
Let g_1 be a permutation in $\mathrm{Sym}(\Omega_1)$ generating G_1, so $G_1 = \langle g_1\rangle$. For $1 \leq i \leq k$, let $\phi_i : \Omega_1 \to \Omega_i$ be a bijection witnessing the permutation isomorphism between G_1 and G_i, as in Definition 1.1.5, and let $g_i = g_1^{\overline{\phi_i}}$, where $\overline{\phi_i}$ is an involution as in Notation 2.1.11.

Finally note that the permutations g_i and the maps ϕ_i can be computed in polynomial time.

Lemma 5.1.3. *Let $H = \langle X\rangle \in \mathfrak{InP}(C_p)$ with k orbits. Then in polynomial time, we can compute a generating set $\{g_1, g_2, \ldots, g_k\}$ of the enveloping group G of H and* *bijections ϕ_i for $2 \le i \le k$ witnessing the permutation isomorphism between $\langle g_1 \rangle$ and $\langle g_i \rangle$.*

Proof. For H-orbit Ω_i, we have $H|_{\Omega_i} = \langle (x|_{\Omega_i}) \mid x \in X \rangle$, which can be computed in polynomial time. As $H|_{\Omega_i}$ is cyclic, any non-trivial element of $H|_{\Omega_i}$ generates $H|_{\Omega_i}$. It is elementary that the conjugacy of two permutations in the symmetric group can be computed in polynomial time, so the bijections ϕ_i can be computed in polynomial time. □

Since we will mostly work with H as a linear code rather than as a permutation group, in Section 5.1.1, we set up an isomorphism γ from H to a linear code. Then in Section 5.1.2, we show that computing $N_{S_n}(H)$ for $H \in \mathfrak{InP}(C_p)$ is polynomially equivalent to computing the monomial automorphism group of $\gamma(H)$.

5.1.1 Representing H as a linear code

In this subsection, we will set up an isomorphism γ between H and a subspace of $\mathbb{F}_p^k$.

Definition 5.1.4. Let G be the enveloping group of H, with generators $g_1, g_2, \dots g_k$, as in Definition 5.1.2. For $g \in G$, there exist integers $r_1,r_2,\dots,r_k$, where $0 \le r_i \le p-1$, such that $g = g_1^{r_1} g_2^{r_2} \dots g_k^{r_k}$. Let γ be the isomorphism from G to $(\mathbb{F}_p^+)^k$ defined by

$$\gamma(g_1^{r_1} g_2^{r_2} \dots g_k^{r_k}) = (r_1,r_2,\dots,r_k).$$

Next we give some matrix notation which we will use throughout the chapter.

Notation 5.1.5. Denote the set of all $s \times k$ matrices over the field $\mathbb{F}_p$ by $M(s,k,\mathbb{F}_p)$. For $M \in M(s,k,\mathbb{F}_p)$, we denote by $M_{i,*}$ and $M_{*,j}$ the i-th row and the j-th column of M respectively.
For a tuple I of distinct elements of $\{1,2,\dots,s\}$, we denote by $M_{I,*}$ the matrix of dimension $|I| \times k$ such that the i-th row of $M_{I,*}$ is $M_{I_i,*}$. Similarly for tuple J of distinct elements of $\{1,2,\dots,k\}$, we denote by $M_{*,J}$ the matrix of dimension $s \times |J|$ such that the j-th column of $M_{*,J}$ is M_{*,J_j}.
We denote the row space of M by $\langle M \rangle$.

A subspace of $\mathbb{F}_p$ can also be regarded as a linear code.

Definition 5.1.6. A *linear code* C of length k and rank s over $\mathbb{F}_p$ is a subspace of $\mathbb{F}_p^k$ with dimension s, which we denote by $C \le \mathbb{F}_p^k$. A *generator matrix* M of C is a matrix in $M(k,s,\mathbb{F}_p)$ such that the rows form a basis for C. A generator matrix $M \in M(k,s,\mathbb{F}_p)$ of C is in *standard form* if its first s columns form the identity matrix I_s of size s.

Remark 5.1.7. We shall assume that all linear codes are given by generator matrices. Since we can row reduce a matrix in polynomial time [BF73] and the row space of a matrix remains the same after row-reductions, we may assume that any generator matrix is row-reduced. Since the rank of a matrix M with k columns is at most k, we shall also assume that a given generator matrix of a code has at most k rows. For more information on linear codes, refer to, for example, [vL99].

Note that $\gamma(H)$ is a subspace of $\gamma(G) = \mathbb{F}_p^k$, so $\gamma(H)$ is linear code of length k over $\mathbb{F}_p$.

5.1.2 The normaliser of H as an automorphism group of a linear code

The main result of this subsection is Theorem 5.1.15, where we show that computing $N_{S_n}(H)$ for $H \in \mathfrak{InP}(C_p)$ is polynomially equivalent to computing a certain group of automorphisms of a code over $\mathbb{F}_p$.

Denote by $\mathbb{F}_p^*$ the multiplicative group of $\mathbb{F}_p$. We start by specialising Lemma 4.3.8 to H in $\mathfrak{InP}(C_p)$.

Proposition 5.1.8. *Let H and g_i be as in Definition 5.1.2. Let L, K, B and the N_i be as in Lemma 4.3.8. Let $r \in \mathbb{F}_p^*$ be primitive. For $1 \leq i \leq k$, let $c_i \in \mathrm{Sym}(\Omega_i)$ such that $c_i^{-1} g_i c_i = g_i^r$. Then $B = \langle g_1, g_2, \dots g_k, c_1, c_2, \dots, c_k \rangle$ and so*

$$\begin{aligned} N_{S_n}(H) &\leq L = \langle B, K \rangle \\ &= \langle g_1, g_2, \dots g_k, c_1, c_2, \dots, c_k, K \rangle \\ &\cong (C_p \rtimes C_{p-1}) \wr S_k. \end{aligned}$$

Proof. We will show that $N_i = \langle g_i, c_i \rangle$ and then the result follows from Proposition 4.3.10.
$\geq$: Since $G_i \leq N_{\mathrm{Sym}(\Omega_i)}(G_i)$, we have $g_i \in N_{\mathrm{Sym}(\Omega_i)}(G_i)$. By the definition of c_i, we have $g_i^{c_i} \in G_i \leq N_{\mathrm{Sym}(\Omega_i)}(G_i)$, and so $c_i \in N_{\mathrm{Sym}(\Omega_i)}(G_i)$.
$\leq$: Let $\nu \in N_{\mathrm{Sym}(\Omega_i)}(G_i)$ and let $g_i \in G_i$. Observe first that since the elements in $\mathrm{Aut}(G_i)$ are fully defined by the image of g_i, and there are $p-1$ choices for this image, $|\mathrm{Aut}(G_i)| \leq p-1$. Now since c_i induces an automorphism α of G_i of order $p-1$, we have $\mathrm{Aut}(G_i) \cong \langle c_i \rangle \cong C_{p-1}$. Then there exists $d \in \langle c_i \rangle$ such that $g_i^d = g_i^\nu$. Since $C_{\mathrm{Sym}(\Omega_i)}(G_i) = G_i$, we have $\nu \in \langle g_i, c_i \rangle$. □

Next, we define the actions of certain subgroups of $GL_k(p)$ on $\gamma(H)$, and set up more notation we will be using later.

Definition 5.1.9. Let D and P be subgroups of $GL_k(p)$ consisting of all invertible diagonal matrices and permutation matrices of $GL_k(p)$ respectively. Let $W := DP$. The groups D, P and W act on the vector space $\mathbb{F}_p^k$ by right multiplication, and the action of W is called the *monomial action*.

Observe that $D \cong (\mathbb{F}_p^*)^k$, $P \cong S_k$ and $W = D \rtimes P \cong \mathbb{F}_p^* \wr S_k$.

Definition 5.1.10. We denote by $diag(t_1, t_2, \ldots, t_k)$ the element of D with diagonal entries $t_1, t_2, \ldots, t_k \in \mathbb{F}_p^*$. So for $M \in M(s, k, \mathbb{F}_p)$ and $1 \leq j \leq k$, we have $(M diag(t_1, t_2, \ldots, t_k))_{*,j} = M_{*,j} t_j$.
Let $K \cong S_k$ be as in Lemma 4.3.8. Define a homomorphism $\rho : K \to P$ by $\rho(\kappa)_{i,j} = 1$ if and only if $\Omega_j = \Omega_i^\kappa$. So for $M \in M(s, k, \mathbb{F}_p)$ and $1 \leq i \leq k$, we have $(M\rho(\kappa))_{*,j} = M_{*,i}$ where $\Omega_j = \Omega_i^\kappa$.

So the right multiplication of M by an element of W first multiplies each column of M by some non-zero scalar and then permutes the columns of M. We may decompose $w \in W$ as follows.

Lemma 5.1.11. *Let $w \in W$. Then in polynomial time, we can find $d \in D$ and $g \in P$ such that $w = dg$.*

Proof. For $1 \leq i \leq k$, let t_i be the non-zero entry of $w_{i,*}$. Let $d = diag(t_1, t_2, \ldots, t_k) \in D$. Observe that $g := d^{-1}w \in P$, so $w = dg$. □

Definition 5.1.12. Let C be a code of length k over $\mathbb{F}_p$. A *monomial automorphism* of C is an element of W that setwise stabilises C. The *monomial automorphism group* of C is the group of all monomial automorphisms of C, denoted by $\mathrm{MAut}(C)$. Two codes $C, C' \leq \mathbb{F}_p^k$ are *monomially equivalent* if there exists $w \in W$ such that $Cw = C'$.

Recall the isomorphism γ from Definition 5.1.4. We will show that computing $N_{S_n}(H)$ is polynomially equivalent to computing the monomial automorphism group of $\gamma(H)$.

Let $L = B \rtimes K$ be as in Proposition 5.1.8. We shall show that the conjugation action of $L/G \cong C_{p-1} \wr S_k$ on G is equivalent to the action of $W \cong C_{p-1} \wr S_k$ on $\mathbb{F}_p^k$. Recall that for all subsets Δ of Ω, we embed $\mathrm{Sym}(\Delta)$ into $\mathrm{Sym}(\Omega)$ by fixing all points in $\Omega \backslash \Delta$.

Lemma 5.1.13. *Let $L = B \rtimes K \leq S_n$ be as in Proposition 5.1.8. Let $\sigma : B \to D$ map $\nu \mapsto diag(t_1, t_2, \ldots, t_k)$, where $\nu^{-1} g_i \nu = g_i^{t_i}$ for $1 \leq i \leq k$. Define a map $\Xi : L \to W$ by writing each $l \in L$ as $\nu\kappa$ for some $\nu \in B$ and $\kappa \in K$ and mapping*

$$l = \nu\kappa \mapsto \sigma(\nu)\rho(\kappa).$$

Then the following statements hold.

1. *Ξ is an epimorphism.*
2. *For all $g \in G$ and for all $l \in L$, we have $\gamma(g^l) = \gamma(g)\Xi(l)$.*
3. *$\mathrm{Ker}(\Xi) = G$.*
4. *$\Xi(N_{S_n}(H)) = \mathrm{MAut}(\gamma(H))$, and so $N_{S_n}(H)$ is the full pre-image of $\mathrm{MAut}(\gamma(H))$ under Ξ.*

Proof. Part 1: To show that Ξ is a homomorphism, let $l_1 = \nu_1\kappa_1$ and $l_2 = \nu_2\kappa_2$ be elements of L, where $\nu_1, \nu_2 \in B$ and $\kappa_1, \kappa_2 \in K$. Then

$$\Xi(l_1l_2) = \Xi(\nu_1\nu_2^{\kappa_1^{-1}}\kappa_1\kappa_2) = \sigma(\nu_1\nu_2^{\kappa_1^{-1}})\rho(\kappa_1\kappa_2) = \sigma(\nu_1)\sigma(\nu_2^{\kappa_1^{-1}})\rho(\kappa_1)\rho(\kappa_2).$$

For $1 \leq j \leq k$, let $1 \leq r \leq k$ be such that $\Omega_j^{\kappa_1} = \Omega_r$, and let $t_r \in \mathbb{F}_p^*$ be such that $\sigma(\nu_2)_r = t_r$. Then using the actions noted in Definition 5.1.10,

$$(M\rho(\kappa_1)\sigma(\nu_2)\rho(\kappa_1)^{-1})_{*,j} = (M\rho(\kappa_1)\sigma(\nu_2))_{*,r} = (M\rho(\kappa_1))_{*,r}t_r = M_{*,j}t_r.$$

Since $g_j^{\kappa_1\nu_2\kappa_1^{-1}} = g_r^{\nu_2\kappa_1^{-1}} = g_r^{t_r\kappa_1^{-1}} = g_j^{t_r\kappa_1^{-1}}$, we have

$$(M\sigma(\nu_2^{\kappa_1^{-1}}))_{*,j} = M_{*,j}\sigma(\nu_2^{\kappa_1^{-1}}))_j = M_{*,j}t_r = (M\rho(\kappa_1)\sigma(\nu_2)\rho(\kappa_1)^{-1})_{*,j}.$$

So $\sigma(\nu_2^{\kappa_1^{-1}}) = \sigma(\nu_2)^{\rho(\kappa_1)^{-1}}$. Therefore $\Xi(l_1l_2) = \sigma(\nu_1)\rho(\kappa_1)\sigma(\nu_2)\rho(\kappa_2) = \Xi(l_1)\Xi(l_2)$ and hence Ξ is a homomorphism. Since σ is an epimorphism, Ξ is also an epimorphism.
Part 2: Let $g \in G$ and let $l = \nu\kappa$ with $\nu \in B$ and $\kappa \in K$. Then there exist $r_i \in \mathbb{F}_p$ and $t_i \in \mathbb{F}_p^*$ such that $g|_{\Omega_i} = g_i^{r_i}$ and $g_i^{\nu} = g_i^{t_i}$. Let $\Omega_i^{\kappa} = \Omega_j$. Then $(g^l)|_{\Omega_j} = ((g^{\nu})|_{\Omega_i})^{\kappa} = (g_i^{r_it_i})^{\kappa}$. Let the maps $\phi_i : \Omega_1 \to \Omega_i$ be as in Definition 5.1.2, and let $\overline{\phi_i}$ be as in Notation 2.1.11. Since K is generated by the $\overline{\phi_i}$ and the conjugation by $\overline{\phi_i}$ swaps g_1 and g_i, we have $g_i^{\kappa} = g_j$. So $(g^{\nu\kappa})|_{\Omega_j} = g_j^{r_it_i}$, and $\gamma(g^l)_j = r_it_i$.
On the other hand,

$$(\gamma(g)\Xi(l))_j = (\gamma(g)\sigma(\nu)\rho(\kappa))_j = (\gamma(g)\sigma(\nu))_i = r_it_i = \gamma(g^l)_j.$$

Part 3: $\leq$: Let $l = \nu\kappa \in L$ such that $\Xi(l)$ is the k-dimensional identity matrix I_k, where $\nu \in B$ and $\kappa \in K$. Then $\rho(\kappa) = \sigma(\nu)^{-1}\Xi(l) = \sigma(\nu)^{-1} \in D \cap P = 1$. So $\kappa = 1$ and hence $l = \nu \in B$. For $h \in H$, we have $\gamma(h^l) = \gamma(h)\Xi(l) = \gamma(h)$, and so $h^l = h$. Hence $l \in C_{S_n}(H)$. Now since $l \in B$ fixes the H-orbits and $C_{S_n}(H) \leq C_p \wr S_k$, it follows that $l \in G$.
$\geq$: Let $g \in G \leq B$. Since G is abelian, $g^{-1}g_ig = g_i$ for all $1 \leq i \leq k$. So $\sigma(g) = I_k$.
Part 4: Let $h \in H$ and let $l \in N_{S_n}(H)$. Then by Part 2, $\gamma(h)\Xi(l) = \gamma(h^l) \in \gamma(H)$ and so $\Xi(l) \in \mathrm{MAut}(\gamma(H))$. Conversely for $\alpha \in \mathrm{MAut}(\gamma(H))$, since Ξ is surjective, there exists $l \in L$ such that $\Xi(l) = \alpha$. By Part 2, $\gamma(h^l) = \gamma(h)\alpha \in \gamma(H)$ for all $h \in H$. So $h^l \in H$ and hence $l \in N_{S_n}(H)$. Therefore $\Xi(N_{S_n}(H)) = \mathrm{MAut}(\gamma(H))$. □

Lastly we show that the Norm-Sym problem for $H \in \mathfrak{InP}(C_p)$ is polynomial time equivalent to the MAut problem, which we shall define now.

Problem 5.1.14 (MAut)**.** Let M be a generator matrix for a code C. Compute a generating set of $\mathrm{MAut}(C)$.

Recall from Remark 5.1.7 that each code of length k is given by a generator matrix with at most k rows. So the input size for MAut is polynomial in n/p, where we do not count the $\log p$ representations of each entry of the matrix.

Theorem 5.1.15. *Fix a prime p. Then the* NORM-SYM *problem for $H = \langle X \rangle \in \mathfrak{InP}(C_p)$ with $H \leq S_n$ is polynomially equivalent to the* MAUT *problem for $C \leq \mathbb{F}_p^{n/p}$.*

Proof. To show the forward reduction, let G be the enveloping group of H. Then by Lemma 5.1.3, we can compute G and the maps ϕ_i in polynomial time. Hence we can compute the map $\gamma : G \to \mathbb{F}_p^k$ as in Definition 5.1.4. Then we can obtain a generator matrix M of $\gamma(H)$ in polynomial time by finding a basis for $\langle \gamma(x) \mid x \in X \rangle$. Assume that we can compute a generating set Y for $\mathrm{MAut}(\gamma(H))$. Since $\mathrm{MAut}(\gamma(H)) \leq W$ and W has a permutation representation in S_{pk}, by Theorem 2.1.1, we assume that $|Y| \leq pk = n$. Then by Lemma 5.1.13, $N_{S_n}(H) = \langle \{\Xi^{-1}(\alpha) \mid \alpha \in Y\}, G \rangle$, where each $\Xi^{-1}(\alpha)$ denotes an element l of L such that $\Xi(l) = \alpha$. By Part 9 of Theorem 2.1.9, $\Xi^{-1}(\alpha)$ can be computed in polynomial time.
For the backward reduction, let $C \leq \mathbb{F}_p^{n/p}$ be a linear code given by a generator matrix $M \in M(s, k, \mathbb{F}_p)$. Let $g_i = (p(i-1)+1, \ldots, pi) \in S_n$ for $1 \leq i \leq n/p$ and let $G = \langle g_1, g_2, \ldots, g_k \rangle$. Let γ be as in Definition 5.1.4 and let $H = \langle \gamma^{-1}(M_{i,*}) \mid 1 \leq i \leq s \rangle$. Assume that we have a generating set Y for $N_{S_n}(H)$. By Theorem 2.1.1, further assume that the set has size polynomial in n. Then by Lemma 5.1.13, $\mathrm{MAut}(C) = \Xi(N_{S_n}(H)) = \langle \Xi(g) \mid g \in Y \rangle$, which can be computed in polynomial time by Part 8 of Theorem 2.1.9. □

5.2 Complexity results

Let $L = B \rtimes K$ be as in Proposition 5.1.8. In Section 5.2.1, we show that for $H \in \mathfrak{InP}(C_p)$, the normaliser $N_{S_n}(H)$ can be computed in time $2^{O(\frac{n}{2p}\log n)}$. Then in Section 5.2.2, we give an efficient algorithm for computing the finest disjoint direct product decomposition of a given group in class $\mathfrak{InP}(C_p)$, which underpins the procedures in the later subsections. In Section 5.2.3, we show that for each non-trivial $\kappa \in K$, if there exists $\nu \in B$ such that $\nu\kappa \in N_{S_n}(H)$, then we can find a witness ν in polynomial time. In Section 5.2.4, we first show that $N_B(H)$ can be computed in polynomial time, then we see how the results in this section come together to reduce the search space by searching for normalising elements in $K \cong S_k$ instead of $L \cong (C_p \rtimes C_{p-1}) \wr S_k$, and hence prove Theorem 5.0.1.

5.2.1 Limiting the depth of the search tree

In this subsection, we show how we can reduce the depth of the search tree using a possibly smaller group $H^\perp$, which we shall define next.

Definition 5.2.1. The *dual* code $C^\perp$ of a code C over $\mathbb{F}_p$ is the subset $\{x \in \mathbb{F}_p^k \mid x \cdot c = 0 \text{ for all } c \in C\}$, where '$\cdot$' denotes the standard dot product. Let $H^\perp := \gamma^{-1}(\gamma(H)^\perp) \leq S_n$.

So for $c \in \mathbb{F}_p^k$, we have $\gamma^{-1}(c) \in H^\perp$ if and only if $\gamma(h) \cdot c = 0$ for all $h \in H$.

We shall show that there exists a bijection between $N_{S_n}(H)$ and $N_{S_n}(H^\perp)$. Recall from Proposition 5.1.8 that $N_{S_n}(H) \le L$.

Lemma 5.2.2. *Let $l = \nu\kappa \in L$, with $\nu \in B$ and $\kappa \in K$. Then $l \in N_{S_n}(H)$ if and only if $\nu^{-1}\kappa \in N_{S_n}(H^\perp)$.*

Proof. Let epimorphism $\Xi : L \to W$ be as in Lemma 5.1.13.
$\Rightarrow$: Assume that $l \in N_{S_n}(H)$. Let $d \in H^\perp$. Let $\sigma(\nu) = diag(t_1,t_2,\dots,t_k) \in D$ and $\gamma(d) = (s_1,s_2,\dots,s_k) \in \mathbb{F}_p^k$. We show that $d^{\nu^{-1}\kappa} \in H^\perp$ by showing that $\gamma(h)\cdot\gamma(d^{\nu^{-1}\kappa}) = 0$ for all $h \in H$.
Let $h \in H$. Since $l \in N_{S_n}(H)$, there exists $g \in H$ such that $g^l = h$. Let $\gamma(g) = (r_1,r_2,\dots,r_k) \in \mathbb{F}_p^k$. By the definition of $H^\perp$, we have $\gamma(g)\cdot\gamma(d) = \sum_{i=1}^{k} r_i s_i = 0$. Then

$$0 = \sum_{i=1}^{k} r_i s_i = \sum_{i=1}^{k} r_i t_i s_i t_i^{-1} = (\gamma(g)\sigma(\nu)) \cdot (\gamma(d)\sigma(\nu^{-1})).$$

Since P permutes the coordinates of $\mathbb{F}_p^k$,

$$\gamma(g)\Xi(l) \cdot \gamma(d)\Xi(\nu^{-1}\kappa) = \gamma(g)\sigma(\nu)\rho(\kappa) \cdot \gamma(d)\sigma(\nu^{-1})\rho(\kappa) = 0.$$

Then by Part 2 of Lemma 5.1.13,

$$\gamma(h) \cdot \gamma(d^{\nu^{-1}\kappa}) = \gamma(g^l) \cdot \gamma(d^{\nu^{-1}\kappa}) = (\gamma(g)\Xi(l)) \cdot (\gamma(d)\Xi(\nu^{-1}\kappa)) = 0.$$

$\Leftarrow$: Since $(H^\perp)^\perp = H$, by the forward implication, $\nu\kappa = (\nu^{-1})^{-1}\kappa \in N_{S_n}(H)$. □

Now, we show that $N_{S_n}(H)$ for $H \in \mathfrak{InP}(C_p)$ can be computed in time $2^{O(n/2p \log n)}$.

Proposition 5.2.3. *Let $H = \langle X \rangle \le S_n$ such that $H \in \mathfrak{InP}(C_p)$ with enveloping group G. Let $\gamma : G \to \mathbb{F}_p^k$ be as in Definition 5.1.4 and let $m := \min\{\dim\gamma(H), k - \dim\gamma(H)\}$. Then $N_{S_n}(H)$ can be computed in time $2^{O(m \log n)}$. Hence $N_{S_n}(H)$ can be computed in time $2^{O(\frac{n}{2p}\log n)}$.*

Proof. By Lemma 5.1.1, we can check that $H \in \mathfrak{InP}(C_p)$ in polynomial time. Let $s := \dim\gamma(H)$. We shall show that $N_{S_n}(H)$ can be computed in time $O(n^s)$, then by Lemmas 4.3.11 and 5.2.2, $N_{S_n}(H)$ can be computed in time $O(n^m)$, hence proving the first assertion. From here, since $\dim(\gamma(H)^\perp) + \dim(\gamma(H)) = k$, the minimum $\min\{\dim\gamma(H), k - \dim\gamma(H)\}$ is at most $k/2$, the last assertion follows.
Row reduction can be done in polynomial time [BF73], so we can obtain a row-reduced generator matrix M of $\gamma(H)$ in polynomial time. Then we may reorder the orbits of H such that the matrix M is in standard form.
If $w \in \mathrm{MAut}(\gamma(H))$ then $M' := Mw = Mdiag(v_1, v_2, \dots, v_k)\rho(\kappa)$ satisfies $\langle M \rangle = \langle M' \rangle$. Since M is in standard form, the elements of $\langle M \rangle$ are uniquely determined by their

first s coordinates. So to test whether $w \in \mathrm{MAut}(\gamma(H))$, it suffices to describe which columns of M are mapped onto the first s coordinates of M' and with which scalars, then the rest of the action of w is uniquely determined up to orbit equivalence. We find all such $w \in \mathrm{MAut}(\gamma(H))$ by considering all choices of $(1^{\kappa^{-1}}, 2^{\kappa^{-1}}, \ldots, s^{\kappa^{-1}}) = J$ and $(v_i)_{i \in J} \in (\mathbb{F}_p^*)^s$.
For each such J and $(v_i)_{i \in J}$, we first define M' in the following way. We take M' to be the matrix in $M(s, k, \mathbb{F}_p)$ such that, for $1 \leq i \leq s$, its i-column $M'_{*,i}$ is $v_{i^{\kappa^{-1}}} M_{*,i^{\kappa^{-1}}}$. Then each row of M' can be extended to a row of length k using the fact that $\langle M' \rangle \leq \langle M \rangle$. Observe that if $w \in \mathrm{MAut}(\gamma(H))$ then $(Mw)_{*,j}$, the j-column of the product Mw, is a scalar multiple of an M-column. Therefore we test whether the choice of J and $(v_i)_{i \in J}$ yields an element of $\mathrm{MAut}(\gamma(H))$ by testing, for all remaining columns $M'_{*,j}$ of M' where $j > s$, if there exists a remaining column $M_{*,j^{\kappa^{-1}}}$ of M and a scalar $v_{j^{\kappa^{-1}}} \in \mathbb{F}_p^*$ such that $M'_{*,j} = v_{j^{\kappa^{-1}}} M_{*,j^{\kappa^{-1}}}$.
If for some j there are no such $M_{*,j^{\kappa^{-1}}}$ and $v_{j^{\kappa^{-1}}}$, then by our observation on $(Mw)_{*,j}$, the choice of J and $(v_i)_{i \in J}$ do not extend to an element of $\mathrm{MAut}(\gamma(H))$, and we move on to the next choice of J and $(v_i)_{i \in J}$. If for some j, there are more than one possible $j^{\kappa^{-1}}$, we choose either. If we have succeeded in finding such $M_{*,j^{\kappa^{-1}}}$ and $v_{j^{\kappa^{-1}}}$ for all $j > s$, then

$$(M diag(v_{1^{\kappa^{-1}}}, \ldots, v_{k^{\kappa^{-1}}})\rho(\kappa))_{*,j} = v_{j^{\kappa^{-1}}} M_{*,j^{\kappa^{-1}}} = M'_{*,j} \quad \text{for all } 1 \leq j \leq k,$$

and so by letting $w = diag(v_{1^{\kappa^{-1}}}, \ldots, v_{k^{\kappa^{-1}}})\rho(\kappa)$, we have $Mw = M'$ and so $w \in \mathrm{MAut}(\gamma(H))$. This step can be done in polynomial time by considering all $k(p-1) \leq n$ choices of $v_{j^{\kappa^{-1}}} M_{*,j^{\kappa^{-1}}}$ for each $j > s$.

Let Y be the set of all elements w of $\mathrm{MAut}(\gamma(H))$ found as above. Since there are $(p-1)^s k^s \leq n^s$ choices for J and $(v_i)_{i \in J}$, and the procedure described above can be done in polynomial time, Y can be computed in $O(n^s)$. We show that $N_{S_n}(H) = \langle C_{S_n}(H), \{\Xi^{-1}(y) \mid y \in Y\} \rangle$, where $\Xi^{-1}(y)$ is an element of L with Ξ-image y.
It is clear that $N_{S_n}(H)$ contains $\langle C_{S_n}(H), \{\Xi^{-1}(y) \mid y \in Y\} \rangle$. For the forward containment, let $\nu \in N_{S_n}(H)$. Then there exists $y \in Y$ such that $M\Xi(\nu) = My$ and hence $M\Xi(\nu)y^{-1} = M$. Then by Part 2 of Lemma 5.1.13, $\nu\Xi^{-1}(y^{-1})$ centralises H. By Part 3 of Lemma 5.1.13, $\nu\Xi^{-1}(y^{-1}) \in \langle G, C_{S_n}(H) \rangle = C_{S_n}(H)$. Therefore $N_{S_n}(H) = \langle C_{S_n}(H), \{\Xi^{-1}(y) \mid y \in Y\} \rangle$.
By Theorem 2.1.16 and Part 9 of Theorem 2.1.9, $N_{S_n}(H)$ can be computed in $O(n^s)$ time. □

While searching for elements of $\mathrm{MAut}(\gamma(H))$ in $W \cong \mathbb{F}_p^* \wr S_k$, the above result limits the depth of the search tree. In practice, we search for $\mathrm{MAut}(\gamma(H))$ in $P \cong S_k$, as we shall see in Section 5.2.4.

5.2.2 Disjoint direct product decomposition

Recall the definition of disjoint direct product decomposition from Definition 3.0.1. We showed in Theorem 3.3.4 that the finest disjoint direct product of a given permutation group can be computed in polynomial time. Recall γ from Definition 5.1.4. Then $H = R_1 \times R_2 \times \ldots \times R_r$ is a disjoint direct product decomposition if and only if $\gamma(H) = \gamma(R_1) + \gamma(R_2) + \ldots + \gamma(R_r)$ and the $\gamma(R_i)$ have pairwise disjoint supports. In this subsection, we will show that the disjoint direct decomposition of a group in class $\mathfrak{InP}(C_p)$ can be computed more efficiently, and calculate its complexity in more detail. We will also introduce some data structures to be used in the later sections.

Notation 5.2.4. Denote the tuple $(1, 2, \ldots, i)$ by $\bar{i}$.

For the rest of this subsection, we assume the following.

Notation 5.2.5. Let $M \in M(s, k, \mathbb{F}_p)$ be a generator matrix of $\gamma(H)$ in standard form. For $1 \leq i \leq s$, let $x_i := \gamma^{-1}(M_{i,*})$, so that we may replace the original generating set of H by $X = \{x_1, x_2, \ldots, x_s\}$.

First, we see how we may detect certain disjoint direct factors of H from M. For a tuple J of distinct elements of $\{1, 2 \ldots, k\}$, let $\Omega_J = \cup_{j \in J} \Omega_j$.

Lemma 5.2.6. *Let $I = \{1 \leq i \leq s \mid M_{i,J} \neq 0\}$. Then there exist R_1, R_2 such that $H = R_1 \times R_2$ and $Supp(R_1) = \Omega_J$ and $Supp(R_2) = \Omega \backslash \Omega_J$ if and only if $M_{I,\bar{k} \backslash J}$ is a zero matrix.*

Proof. $\Leftarrow$: Let $R_1 = \langle x_i \mid i \in I \rangle$ and $R_2 = \langle x_i \mid i \notin I \rangle$. Since the x_i generate H and H is abelian, we have $H = R_1 \times R_2$. Then as $M_{I,\bar{k} \backslash J} = 0$, the support $Supp(R_1) \subseteq \Omega_J$. By the definition of I, we have $M_{\bar{s} \backslash I, J} = 0$, and so $Supp(R_2) \subseteq \Omega \backslash \Omega_J$. Therefore R_1 and R_2 have disjoint supports and hence $H = R_1 \times R_2$ is a disjoint direct product decomposition.
$\Rightarrow$: Let $i \in I$. As $H = R_1 \times R_2$, there exist $r_1 \in R_1$ and $r_2 \in R_2$ such that $x_i = r_1 r_2$. Note that r_1 is non-trivial by the definition of I. We will show that $M_{i,\bar{k} \backslash J} = 0$ by showing that $M_{i,*} = \gamma(r_1)$. As M is in standard form, each element of $\langle M \rangle$ is fully determined by its restriction to the first s positions, so it suffices to show that $M_{i,\bar{s}} = \gamma(r_1)_{\bar{s}}$.
Since r_2 fixes all points in Ω_J, we have that $x_i|_{\Omega_J} = r_1|_{\Omega_J}$. Similarly, $x_i|_{\Omega \backslash \Omega_J} = r_2|_{\Omega \backslash \Omega_J}$. So

$$M_{i,J} = \gamma(r_1)_J \text{ and } M_{i,\bar{k} \backslash J} = \gamma(r_2)_{\bar{k} \backslash J}. \tag{5.1}$$

In particular, we have $M_{i,\bar{s} \cap J} = \gamma(r_1)_{\bar{s} \cap J}$. Hence to show that $M_{i,\bar{s}} = \gamma(r_1)_{\bar{s}}$, it remains to show that $M_{i,\bar{s} \backslash J} = \gamma(r_1)_{\bar{s} \backslash J}$.
Since M is in standard form, there exists a unique $1 \leq t \leq s$ such that $M_{i,t} \neq 0$ and $M_{i,\bar{s} \backslash t} = 0$. Suppose first that $t \notin J$. We will show that $M_{i,*} = \gamma(r_2)$, which contradicts the definition of I as $\gamma(r_2)_J = 0$. Since $\bar{s} \cap J \subseteq \bar{s} \backslash t$, we have $M_{i,\bar{s} \cap J} = 0$.

Then since r_2 fixes Ω_J pointwise, $M_{i,\overline{s}\cap J} = 0 = \gamma(r_2)_{\overline{s}\cap J}$. By Equation (5.1), we have $M_{i,\overline{s}\setminus J} = \gamma(r_2)_{\overline{s}\setminus J}$, and so $M_{i,\overline{s}} = \gamma(r_2)_{\overline{s}}$. Since each element of $\langle M\rangle$ is fully determined by its restriction to the first s positions, $M_{i,*} = \gamma(r_2)$, a contradiction.
Therefore $t \in J$, so $\overline{s}\setminus J \subseteq \overline{s}\setminus t$ and hence $M_{i,\overline{s}\setminus J} = 0$. Since r_1 fixes all points outside Ω_J, it follows that $\gamma(r_1)_{\overline{s}\setminus J} = 0 = M_{i,\overline{s}\setminus J}$. So $M_{i,\overline{s}} = \gamma(r_1)_{\overline{s}}$ and hence $M_{i,\overline{k}\setminus J} = 0$. $\square$

Recall the new generating set X of H from Notation 5.2.5. We shall show in Lemma 5.2.8 that the finest disjoint direct product of H can be obtained by computing a certain partition of X.

Definition 5.2.7. Let $a \in \mathbb{N}$ and let $v \in \mathbb{F}_p^a$. Then the *support* of v is $Supp(v) := \{1 \leq i \leq a \mid v_i \neq 0\}$. Let V be a subspace of $\mathbb{F}_p^a$, then its support $Supp(V) = \cup_{v\in V} Supp(v)$.

Lemma 5.2.8. *Let $\mathcal{P} = \langle C_1 \mid C_2 \mid \ldots \mid C_r\rangle$ be the finest partition of $\{x_1, x_2, \ldots, x_s\}$ such that for all distinct C_a and C_b, we have*

$$\textit{for all } x \in C_a \textit{ and } y \in C_b, Supp(x) \cap Supp(y) = \emptyset. \tag{5.2}$$

For each cell C_i of $\mathcal{P}$, let $R_i = \langle x \in C_i\rangle$. Then $H = R_1 \times R_2 \times \ldots \times R_r$ is the finest disjoint direct product decomposition of H.

Proof. Since H is abelian and $\{x_1, x_2, \ldots, x_s\}$ generates H, we have $H = R_1 \times R_2 \times \ldots \times R_r$. Since each pair of cells in $\mathcal{P}$ satisfies Equation (5.2), the groups R_i have pairwise disjoint supports. So $H = R_1 \times R_2 \times \ldots \times R_r$ is a disjoint direct product decomposition. It remains to show that it is the finest disjoint direct product decomposition.
Let $1 \leq i \leq r$. Suppose that $R_i = R_{i1} \times R_{i2}$ is a disjoint direct product decomposition of R_i. Let $J_1 := Supp(\gamma(R_{i1}))$ and let $I_1 := \{1 \leq i \leq s \mid M_{i,J_1} \neq 0\}$. Then by Lemma 5.2.6, $M_{I_1,\overline{k}\setminus J_1}$ is a zero matrix. So $X_{I1} := \{x_t \mid t \in I_1\}$ has support $Supp(R_{i1})$. Similarly, there exists a subset $X_{I2} \subseteq C_i$ with support $Supp(R_{i2})$. So $Supp(x) \cap Supp(y) = \emptyset$ for all $x \in X_{I1}$ and $y \in X_{I2}$, which contradicts the assumption that $\mathcal{P}$ is the finest partition which satisfies Equation (5.2). $\square$

We shall compute the finest disjoint direct product decomposition of H by computing the finest partition $\mathcal{P}$ of $\{x_1, x_2, \ldots, x_s\}$ as in Lemma 5.2.8. We will do so by iteratively computing certain partitions $\mathcal{P}_j$ of $\{x_1, x_2, \ldots, x_s\}$, for $1 \leq j \leq k$, which we will now define.

Notation 5.2.9. For $1 \leq j \leq k$, let $\Delta_j = \cup_{i=1}^{j}\Omega_i$ and let $\mathcal{P}_j$ be the finest partition of $\{x_1, x_2, \ldots, x_s\}$ such that for all distinct cells C_a and C_b of $\mathcal{P}_j$, we have

$$\text{for all } x \in C_a \text{ and } y \in C_b, Supp(x|_{\Delta_j}) \cap Supp(y|_{\Delta_j}) = \emptyset. \tag{5.3}$$

Remark 5.2.10. We note the following.

1. Since $Supp(x|_{\Delta_{j-1}}) \cap Supp(y|_{\Delta_{j-1}}) \subseteq Supp(x|_{\Delta_j}) \cap Supp(y|_{\Delta_j})$, each cell of $\mathcal{P}_j$ is a union of cells of $\mathcal{P}_{j-1}$.

2. Recall that we assume that M is in standard form. So for $j \leq s$, the partition $\mathcal{P}_j$ is the trivial partition consisting of all singletons.

3. If $j = k$, then $\Delta_j = \Omega$, so $\mathcal{P}_k$ is the partition $\mathcal{P}$ from Lemma 5.2.8.

Next, we show how we can compute the $\mathcal{P}_j$ for $s < j \leq k$.

Lemma 5.2.11. *Let $s \leq j < k$. Let $\mathcal{P}_j = \langle C_1 \mid C_2 \mid \dots \mid C_r \rangle$ be as in Notation 5.2.9. Let*

$$J := \{C_a \mid 1 \leq a \leq r \text{ and there exists } x \in C_a \text{ such that } x|_{\Omega_{j+1}} \neq 1\}$$

and let $C_J = \cup_{C_a \in J} C_a$. Let $\mathcal{P}'$ be the partition with cells C_J and all C_a for $C_a \notin J$. Then $\mathcal{P}_{j+1} = \mathcal{P}'$.

Proof. First, we verify that $\mathcal{P}'$ satisfies Equation (5.3). Let C'_a and C'_b be distinct cells of $\mathcal{P}'$. Let $x_u \in C'_a$ and $x_v \in C'_b$. We show that $Supp(x_u|_{\Delta_{j+1}}) \cap Supp(x_v|_{\Delta_{j+1}}) = \emptyset$ by showing that $Supp(M_{u,\overline{j+1}}) \cap Supp(M_{v,\overline{j+1}}) = \emptyset$. Suppose first that $C'_a, C'_b \notin J$, then C'_a and C'_b are cells of $\mathcal{P}_j$. So $Supp(M_{u,\bar{j}}) \cap Supp(M_{v,\bar{j}}) = \emptyset$. By the definition of J, we have $M_{u,j+1} = M_{v,j+1} = 0$, and so indeed $Supp(M_{u,\overline{j+1}}) \cap Supp(M_{v,\overline{j+1}}) = \emptyset$.
Suppose now that $C'_a \notin J$ and $C'_b = C_J$. Then $M_{u,j+1} = 0$, and so $Supp(M_{u,\overline{j+1}}) \cap Supp(M_{v,\overline{j+1}}) = Supp(M_{u,\bar{j}}) \cap Supp(M_{v,\bar{j}})$. Since C_J is a union of cells of $\mathcal{P}_j$, there exists a cell C of $\mathcal{P}_j$ such that $x_v \in C$. Then by the definition of $\mathcal{P}_j$, we have that $Supp(M_{u,\bar{j}}) \cap Supp(M_{v,\bar{j}}) = \emptyset$. So $\mathcal{P}'$ indeed satisfies Equation (5.3).

It remains to show that $\mathcal{P}'$ is the finest possible partition of the generators. Let C'_a be a cell of $\mathcal{P}'$. We show that there does not exists a proper subset $T' \subset C'_a$ such that $Supp(M_{u,\overline{j+1}}) \cap Supp(M_{v,\overline{j+1}}) = \emptyset$ for all $x_u \in T'$ and $x_v \in C'_a \backslash T'$. Aiming for a contradiction, assume that such a T' exists.
Suppose first that $C'_a \notin J$. Then we have $Supp(M_{u,\overline{j+1}}) = Supp(M_{u,\bar{j}})$ for all $x_u \in C'_a$. So $Supp(M_{u,\bar{j}}) \cap Supp(M_{v,\bar{j}}) = \emptyset$ for all $x_u \in T'$ and $x_v \in C'_a \backslash T'$. Since C'_a is a cell of $\mathcal{P}_j$, this contradicts that $\mathcal{P}_j$ is the finest partition satisfying Equation (5.3).
So $C'_a = C_J$. By the definition of J, for all $C \in J$, there exists $x_u \in C$ such that $M_{u,j+1} \neq 0$. Observe that T' either contains all such x_u or none of them. Without loss of generality, suppose that T' contains all such x_u. So $T' \cap C \neq \emptyset$ for all $C \in J$. By Remark 5.2.10, $T' \cap C$ is a union of cells of $\mathcal{P}_j$, so the entire cell C is contained in T. Hence $C_J \subseteq T$. Since $T' \subseteq C_J$, we have $T' = C_J$, which is a contradiction. □

Hence, we have the main result of this subsection.

Proposition 5.2.12. *Let H be as in Definition 5.1.2. Let $M \in M(s, k, \mathbb{F}_p)$ be a given generator matrix of $\gamma(H)$ in standard form. Then the partitions $\mathcal{P}_j$ from Notation 5.2.9 and so the partition $\mathcal{P}$ can be computed in $O(s\alpha(s)k)$ time, where α is the inverse Ackermann function.*
Hence the disjoint direct product decomposition of H can be computed in $O(s\alpha(s)k)$ time.

Proof. We compute the disjoint direct product decomposition of H by computing the partition $\mathcal{P}$ as in Lemma 5.2.8. The partition $\mathcal{P}$ is computed by iteratively computing the $\mathcal{P}_j$ in Notation 5.2.9 using Lemma 5.2.11. As in Remark 5.2.10, we initialise $\mathcal{P}_s$ to be the trivial partition consisting of all singletons. We evoke the procedure in Lemma 5.2.11 $O(k)$ times to compute the partitions $\mathcal{P}_j$ for each $s < j \leq k$. It remains to show that the procedure in Lemma 5.2.11 can be computed in time $O(s\alpha(s))$.
To compute $\mathcal{P}_{j+1}$, we merge cells of $\mathcal{P}_j$ if the cells contain a permutation x such that $x|_{\Omega_{j+1}} \neq 1$. Checking if the projection of each x on Ω_{j+1} is trivial can be done in constant time by checking if the corresponding row of M has 0 at the $(j+1)$-st position. We construct $\mathcal{P}_{j+1}$ using the Union-Find data structure, which has a worst case complexity of $O(s\alpha(s))$ [TvL84]. $\square$

5.2.3 Normalising elements that permute the H-orbits

Let $L = B \rtimes K$ be as in Proposition 5.1.8 and recall that $N_{S_n}(H) \leq L$. In Proposition 5.2.18, we show that given a non-trivial element $\kappa \in K$, in polynomial time, we can decide if there exists an element of $N_{S_n}(H)$ which induces the permutation κ on the H-orbits.

Lemma 5.2.13. *Let $F := GL_s(p) \times D$. Define an action of F on $M(s, k, \mathbb{F}_p)$ by*

$$M^{(R,d)} = R^{-1}Md \quad \text{for all } R \in GL_s(p) \text{ and } d \in D.$$

Let $\kappa \in K \leq S_n$. Let $M, M' \in M(s, k, \mathbb{F}_p)$ be generator matrices of $\gamma(H)$ and $\gamma(H^{\kappa^{-1}})$ respectively. Then there exists $\nu \in B$ such that $\nu\kappa \in N_{S_n}(H)$ if and only if there exists $f \in F$ such that $M^f = M'$.

Proof. Let $\nu \in B$. Then $\nu\kappa \in N_{S_n}(H)$ if and only if $H^\nu = H^{\kappa^{-1}}$, which means that H and $H^{\kappa^{-1}}$ are in the same B-orbit. By Lemma 5.1.13, this is equivalent to $\gamma(H)$ and $\gamma(H^{\kappa^{-1}})$ being in the same D-orbit.
Observe that $\langle M \rangle = \langle M' \rangle$ if and only if M and M' are in the same orbit under left multiplication by $GL_s(p)$. Therefore H and $H^{\kappa^{-1}}$ are in the same B-orbit if and only if M and M' are in the same F-orbit. $\square$

We decide if M and M' are in the same F-orbit by comparing their orbit representatives. Algorithm 1 of [Feu09] computes such representatives for matrices $A \in M(s, k, \mathbb{F}_p)$. Since we will be proving its complexity, we will present our minor variation of the algorithm here.

We first define an ordering on $M(s, k, \mathbb{F}_p)$. Let $A, A' \in M(s, k, \mathbb{F}_p)$. Let $A^R_{*,i}$ denote $A_{*,i}$ reversed. Then we define $A \prec A'$ if there exists j such that $A_{*,i} = A'_{*,i}$ for $1 \leq i < j$ but $A^R_{*,j} <_{lex} A'^R_{*,j}$, where $<_{lex}$ denotes the lexicographic ordering. We define the orbit representative of A under F to be the least image under the ordering $\prec$. Recall that we denote the tuple $(1, 2, \ldots, i)$ by $\bar{i}$.

In Feulner's paper, he computes a series of partitions called the *support partitions*, which we now define.

Definition 5.2.14. Let $A \in M(s, k, \mathbb{F}_p)$. For all $1 \le j \le k$, the *j-support partition* $\mathcal{Q}_j$ of A is the finest partition of $\{1, 2, \dots, s\}$ such that for $1 \le i \le j$, there exists a unique cell C of $\mathcal{Q}_j$ such that $Supp(A_{*,i}) \subseteq C$.

We show that the support partitions and the partitions P_j from Notation 5.2.9 are equivalent.

Lemma 5.2.15. *Let $\mathcal{Q}_j$ be as in Definition 5.2.14. For $1 \le i \le s$, let $x_i := \gamma^{-1}(A_{i,*})$ and let P_j be as in Notation 5.2.9. For each cell C of $\mathcal{Q}_j$, let $\tilde{C} := \{x_u \mid u \in C\}$. Let $\tilde{\mathcal{Q}}_j$ be the partition of $\{x_1, x_2, \dots, x_s\}$ consisting of such cells $\tilde{C}$. Then $\tilde{\mathcal{Q}}_j = \mathcal{P}_j$.*

Proof. Let C_a and C_b be distinct cells of $\mathcal{Q}_j$. Let $\Delta_j = \cup_{i=1}^{j} \Omega_i$. First we show that $Supp(x_u|_{\Delta_j}) \cap Supp(x_v|_{\Delta_j}) = \emptyset$ for all $x_u \in \tilde{C}_a$ and $x_v \in \tilde{C}_b$ by showing that $Supp(M_{u,\bar{j}}) \cap Supp(M_{v,\bar{j}}) = \emptyset$.
Let $u \in C_a$ and $v \in C_b$. If there exists $t \in Supp(M_{u,\bar{j}}) \cap Supp(M_{v,\bar{j}})$, then $M_{u,t}$ and $M_{v,t}$ are both non-zero. So $Supp(M_{*,t}) \cap C_a \neq \emptyset$. Then by the definition of $\mathcal{Q}_j$, we have $Supp(M_{*,t}) \subseteq C_a$. Similarly, $Supp(M_{*,t}) \subseteq C_b$, which is a contradiction.

Next we show that $\tilde{\mathcal{Q}}_j$ is the finest possible. Suppose that there exists $T \subset \tilde{C}_a$ such that $Supp(x_u|_{\Delta_j}) \cap Supp(x_v|_{\Delta_j}) = \emptyset$ for all $x_u \in T$ and for all $x_v \in \tilde{C}_a \backslash T$. Let $C_{a1} := \{1 \le u \le s \mid x_u \in T\}$ and $C_{a2} := \{1 \le u \le s \mid x_u \in \tilde{C}_a \backslash T\}$ be disjoint subsets of $\tilde{C}_a$. Define R_j to be the partition of $\bar{s}$ consisting of cells C_{a1}, C_{a2} and all other cells C_b of $\mathcal{Q}_j$ where $b \neq a$. Then R_j is a partition finer than $\mathcal{Q}_j$. We shall show that for all $1 \le i \le j$, there exists a unique cell C of R_j such that $Supp(M_{*,i}) \subseteq C$, which contradicts the finest property of $\mathcal{Q}_j$.
Let $1 \le i \le j$. Then by the definition of $\mathcal{Q}_j$, there exists a cell C of $\mathcal{Q}_j$ such that $Supp(M_{*,i}) \subseteq C$. If $C \neq C_a$, then we are done. Suppose now that $C = C_a$. If $Supp(M_{*,i})$ intersects both C_{a1} and C_{a2} non-trivially, then there exists $u \in C_{a1}$ and $v \in C_{a2}$ such that both $M_{u,i}$ and $M_{v,i}$ are non-zero. This implies that $Supp(M_{u,\bar{j}}) \cap Supp(M_{v,\bar{j}}) \neq \emptyset$, which contradicts the definition of T. Therefore, either $Supp(M_{*,i}) \subseteq C_{a1}$ or $Supp(M_{*,i}) \subseteq C_{a2}$. □

Next, we will present a simplified version of Algorithm 1 in [Feu09]. The differences are as follows.

- Feulner considers codes over arbitrary finite fields, but we restrict to fields of prime order.
- We assume that A is in standard form. Feulner did not make this assumption.
- For $1 \le j \le k$, let $D^{[j]} \cong (\mathbb{F}_p^*)^j$ be the subgroup of D consisting of matrices $diag(t_1, t_2, \dots t_j, 1, 1, \dots, 1)$ for $t_1, t_2, \dots t_j \in \mathbb{F}_p^*$. Algorithm 1 in [Feu09] computes

the orbit representative of A under the action of $GL_s(p) \times D^{[j+1]}$ given an orbit representative of A under the action of $GL_s(p) \times D^{[j]}$. We compute the F-orbit representative of A directly.

- Feulner's algorithm performs row reduction, computes the partition $\mathcal{Q}_{j+1}$ and computes the orbit representative of A under the action of $GL_s(p) \times D^{[j+1]}$ simultaneously. We first row reduce A and compute all $\mathcal{Q}_j$.

Lemma 5.2.16. *Let $A \in M(s, k, \mathbb{F}_p)$ be such that the first s columns are linearly independent. Let $f \in F$ be such that A^f is row-reduced. Then $A^f \preceq A$ and so A^f is the orbit representative of A under the action of $GL_s(p) \times D^{[s]}$.*

Proof. Since the first s columns of A are linearly independent, $(A^f)_{*,\bar{s}}$ is an identity matrix. Since the s lexicographically smallest vectors of $\mathbb{F}_p^s$ are $(A^f)^R_{*,1}$, $(A^f)^R_{*,2}$, …, $(A^f)^R_{*,s}$, we have $A^f \preceq A$. □

Algorithm 5 Computing the orbit representative of $A \in M(s, k, \mathbb{F}_p)$ under the action of F

Input: $A \in M(s, k, \mathbb{F}_p)$, where the first s columns are linearly independent
Output: F-orbit representative A' of A and $f \in F$ such that $A' = Af$

```
 1: Let $A'$ be a row-reduced copy of $A$                                   ▷ $O(sk^2)$
 2: Get the $j$-support partition $\mathcal{Q}_j$ of $A'$ for $1 \le j \le k$
                                  ▷ $O(s\alpha(s)k)$ by Proposition 5.2.12 and Lemma 5.2.15
 3: for $j \in [s, s+1, \dots, k-1]$ do                                     ▷ $O(k)$ times
 4:     for $i \in [s, s-1, \dots, 1]$ do                                   ▷ $O(s)$ times
 5:         $C \leftarrow$ cell of $\mathcal{Q}_j$ such that $i \in C$      ▷ Finding $C$ is $O(s)$
 6:         $u \leftarrow A'_{i,j}$
 7:         $T \leftarrow \{t \in C \mid A'_{t,j+1} \neq 0\}$               ▷ Getting $T$ is $O(s)$
 8:         if $u \neq 0$ and $T = \emptyset$ then
 9:             for $r \in C$ do                                            ▷ $O(s)$ times
10:                 Multiply row $A'_{r,*}$ by $u^{-1}$                     ▷ $O(k)$ multiplications
11:             end for
12:             for all $1 \le t \le j$ do                                  ▷ $O(k)$ times
13:                 if $\exists c \in C$ such that $A'_{c,t} \neq 0$ then   ▷ $O(s)$ time
14:                     Multiply column $A'_{*,t}$ by $u$                   ▷ $O(s)$ multiplications
15:                 end if
16:             end for
17:         end if
18:     end for
19: end for
20: return $A'$ and $f \in F$ such that $A' = A^f$
```

Theorem 5.2.17. *Let $A \in M(s, k, \mathbb{F}_p)$ be such that the first s columns are linearly independent. Then Algorithm 5 returns the orbit representative of A under F in time $O(s^2k^2)$.*

Proof. The correctness follows from Lemma 5.2.16 and [Feu09]. We can write A in reduced row echelon form in time $O(sk^2)$ using Gauss-Jordan elimination [BF73]. By Proposition 5.2.12 and Lemma 5.2.15, the $\mathcal{Q}_j$ can be computed in $O(s\alpha(s)k)$ time, where α is the inverse Ackerman function. We store each partition $\mathcal{Q}_j$ by storing an array L_j such that for $i \leq s$, we have that i is in cell $C_{L_j[i]}$ of $\mathcal{Q}_j$, where $L_j[i]$ denotes the i-th element of L_j. Computing each L_j takes $O(s)$ time.
Then we can get C of line 5 by computing the positions of $C_{L_j[i]}$ in L_j, which can be done in $O(s)$ time. Since C has at most size s, obtaining T can be done in $O(s)$ time. Similarly, line 13 is of $O(s)$ complexity. Hence the algorithm has complexity $O(sk^2 + s\alpha(s)k + sk + ks(s + s + sk + k(s + s))) = O(s^2k^2)$. □

Lastly, we see how we use Algorithm 5 to decide if there exists an element of $N_{S_n}(H)$ which induces a given permutation on the H-orbits.

Proposition 5.2.18. *Let $H \leq S_n$ such that $H \in \mathfrak{InP}(C_p)$. Let $L = B \rtimes K$ be as in Proposition 5.1.8 and let $\kappa \in K$. Assume that we have generator matrices $M, M' \in M(s, k, \mathbb{F}_p)$ for $\gamma(H)$ and $\gamma(H^{\kappa^{-1}})$. Then in $O(s^2k^2)$, we can determine if there exists $\nu \in B$ such that $\nu\kappa \in N_{S_n}(H)$, and output such a ν if it exists.*

Proof. By Lemma 5.2.13, such a ν exists if and only if M and M' are in the same F-orbit and so have the same F-orbit representative. By Theorem 5.2.17, elements (R_1, d_1) and (R_2, d_2) of F mapping M and M' respectively to their orbit representative under F can be computed in in $O(s^2k^2)$.
If the orbit representatives are different, then no such ν exists, so we assume that M and M' have the same orbit representative. Let σ be as in Lemma 5.1.13. Since $M^{(R,d_1)(R',d_2)^{-1}} = M'$, we have $\langle M\rangle d_1d_2^{-1} = \langle M'\rangle$. Therefore $H^{\sigma^{-1}(d_1d_2^{-1})} = H^{\kappa^{-1}}$ and hence $\sigma^{-1}(d_1d_2^{-1})\kappa \in N_{S_n}(H)$. So $\nu = \sigma^{-1}(d_1d_2^{-1})$. □

5.2.4 Computing $N_{S_n}(H)$ by searching in K

Recall $L = B \rtimes K$ from Proposition 5.1.8. In this subsection, we will first show in Proposition 5.2.21 that given the finest disjoint direct product decomposition of H, we can efficiently compute $N_B(H)$. Then in Theorem 5.2.22, we see how the results in earlier subsections come together to show that $N_{S_n}(H)$ for $H \in \mathfrak{InP}(C_p)$ can be computed in time $O((\frac{n}{p})!)$.

We start by showing that $C_B(H)$ is the enveloping group G of H.

Lemma 5.2.19. *Let $H \in \mathfrak{InP}(C_p)$ and let G be the enveloping group of H. Let B be as in Proposition 5.1.8. Then $C_B(H) = G$.*

Proof. $\leq$: Let $\Omega_1, \Omega_2, \ldots, \Omega_k$ be the orbits of H. Since B fixes each H-orbit setwise, we have $C_B(H) \leq C_{\mathrm{Sym}(\Omega_1)}(H|_{\Omega_1}) \times C_{\mathrm{Sym}(\Omega_2)}(H|_{\Omega_2}) \times \ldots \times C_{\mathrm{Sym}(\Omega_k)}(H|_{\Omega_k}) = G$.
$\geq$: Let $\{g_1, g_2, \ldots, g_k\}$ be a generating set of $G \leq B$. Then as H and G are abelian, each g_i centralises H. □

Let X be a generating set of H corresponding to the rows of M in standard form, as in Notation 5.2.5. Next we show that each element of $N_B(H)$ conjugates elements of X with non-disjoint supports to the same power.

Lemma 5.2.20. *Let $\nu \in N_B(H)$. Let $x_i, x_j \in X$ be such that $Supp(x_i) \cap Supp(x_j) \neq \emptyset$. Then there exists $1 \leq u \leq p-1$ such that $x_i^\nu = x_i^u$ and $x_j^\nu = x_j^u$.*

Proof. Since $x_i^\nu \in H$, the image $\gamma(x_i^\nu) \in \langle M \rangle$ and hence $\gamma(x_i^\nu)$ is a linear combination of the rows of M. Since $M_{*,\overline{s}}$ is an identity matrix, $M_{i,i} \neq 0$ and $M_{i,l} = 0$ for all $1 \leq l \leq s$ such that $l \neq i$. Then since ν setwise stabilises each H-orbit, $\gamma(x_i^\nu)_i \neq 0$ and $\gamma(x_i^\nu)_l = 0$ for all $1 \leq l \leq s$ where $l \neq i$. Hence $\gamma(x_i^\nu)$ is a multiple of $M_{i,*} = \gamma(x_i)$ and therefore there exists $t \in \mathbb{F}_p^*$ such that $\gamma(x_i^\nu) = \gamma(x_i)t$. Then $x_i^\nu = x_i^t$. Similarly, there exists t' such that $x_j^\nu = x_j^{t'}$. We shall show that $t = t'$.
Let Ω_u be an H-orbit such that $\Omega_u \subseteq Supp(x_i) \cap Supp(x_j)$. Let $g_u = g_1^{\overline{\phi_u}}$, as in Definition 5.1.2. Let $1 \leq r_u \leq p-1$ be such that $x_i|_{\Omega_u} = g_u^{r_u}$. Then since $x_i^\nu = x_i^t$, we have

$$(g_u^{r_u})^{\nu|_{\Omega_u}} = (x_i|_{\Omega_u})^{\nu|_{\Omega_u}} = (x_i^\nu)|_{\Omega_u} = (x_i^t)|_{\Omega_u} = (x_i|_{\Omega_u})^t = g_u^{r_u t}.$$

Hence $g_u^{\nu|_{\Omega_u}} = g_u^t$. Similarly, since $\Omega_u \subseteq Supp(x_j)$, we have that $g_u^{\nu|_{\Omega_u}} = g_u^{t'}$. Therefore $t = t'$. □

Now we show that given the finest disjoint direct product decomposition of H, we can compute $N_B(H)$ in polynomial time. Recall that for $\Delta \subseteq \Omega$, we identify $\mathrm{Sym}(\Delta)$ as a subgroup of $\mathrm{Sym}(\Omega)$.

Proposition 5.2.21. *Let $H \in \mathfrak{InP}(C_p)$ with orbits $\Omega_1, \Omega_2, \ldots, \Omega_k$. Let G be the enveloping group of H and let $H = R_1 \times R_2 \times \ldots \times R_r$ be the finest disjoint direct product decomposition of H. Let B be the group generated by $N_{\mathrm{Sym}(\Omega_i)}(H|_{\Omega_i})$ for all $1 \leq i \leq k$. Then $N_B(H)$ can be computed in polynomial time.*

Proof. Let $g_1, g_2, \ldots, g_k$ be the standard generators of G, as in Definition 5.1.2. Let $t \in \mathbb{F}_p^*$ be primitive. For $1 \leq i \leq r$, let $\Gamma_i = Supp(R_i)$ and $\eta_i \in \mathrm{Sym}(\Gamma_i)$ such that $g_j^{\eta_i} = g_j^t$ for all g_j such that $Supp(g_j) \subseteq \Gamma_i$. We first show that

$$N_B(H) = \langle G, \eta_1, \eta_2, \ldots, \eta_r \rangle. \tag{5.4}$$

$\geq$: By Lemma 5.2.19, $G = C_B(H) \leq N_B(H)$. To show that each η_i is in $N_B(H)$, let $h \in H$. Then we can write $h = h_1 h_2 \ldots h_r$ with $h_j \in R_j$, and so $h^{\eta_i} = h_1^{\eta_i} h_2^{\eta_i} \ldots h_r^{\eta_i}$. If $i \neq j$, then since h_j and η_i have disjoint supports, $h_j^{\eta_i} = h_j$. Therefore $h^{\eta_i} \in H$ if and only if $h_i^{\eta_i} \in R_i$.
If $\Omega_u \subseteq \Gamma_i$, then $h_i|_{\Omega_u} = g_u^r$ for some r, and so

$$(h_i^{\eta_i})|_{\Omega_u} = (h_i|_{\Omega_u})^{\eta_i|_{\Omega_u}} = (g_u^r)^{\eta_i|_{\Omega_u}} = g_u^{rt} = (h_i|_{\Omega_u})^t.$$

Therefore $h_i^{\eta_i} = h_i^t \in R_i$.
$\leq$: Let $\nu \in N_B(H)$. For $1 \leq i \leq r$, let $\nu_i = \nu|_{\Gamma_i}$. We show that $\nu_i \in \langle G, \eta_i \rangle$ for all i, from which the containment follows.
By Lemma 5.2.20, there exists $t^a \in \mathbb{F}_p^*$ such that $x^{\nu_i} = x^{t^a}$ for all $x \in X \cap R_i$. So $x^{\nu_i} = x^{t^a} = x^{\eta_i^a}$. Therefore $\nu_i \in \langle C_B(R_i), \eta_i^a \rangle \leq \langle G, \eta_i \rangle$.

Now that we have proved Equation (5.4), for the complexity result, by Lemma 5.1.3, the g_i can be computed in polynomial time. Since conjugacy of permutations in the symmetric group is in polynomial time, each η_i can be computed in polynomial time. Therefore $N_B(H)$ can be computed in polynomial time. □

Lastly, we show that to compute $N_{S_n}(H)$ for $H \in \mathfrak{InP}(C_p)$, it suffices to search in K. Our implementation to compute $N_{S_n}(H)$ will use the procedure described in the following proof.

Theorem 5.2.22. *The* Norm-Sym *problem for $H = \langle X \rangle \leq S_n$ where $H \in \mathfrak{InP}(C_p)$ can be computed in time $2^{O(\frac{n}{p} \log \frac{n}{p})}$.*

Proof. By Lemma 5.1.1, we can recognise that $H \in \mathfrak{InP}(C_p)$ in polynomial time. By Lemma 5.1.3, in polynomial time, we can compute the generators $g_1, g_2, \ldots, g_k$ of G and hence obtain the map $\gamma : G \to \mathbb{F}_p^k$ as in Definition 5.1.4. Since row reduction can be computed in polynomial time [BF73], we may obtain a generator matrix M of $\gamma(H) = \langle \gamma(x) \mid x \in X \rangle$ in standard form in polynomial time. By Proposition 5.2.12, we can compute the finest disjoint direct product decomposition of H in polynomial time. Next we compute $C_{S_n}(H)$ and $N_B(H)$, which can be computed in polynomial time by Theorem 2.1.16 and Proposition 5.2.21 respectively. Initialise N as $\langle C_{S_n}(H), N_B(H) \rangle$. For each $\kappa \in K$, using Proposition 5.2.18, in polynomial time, we determine if there exists $\nu \in B$ such that $\nu\kappa \in N_{S_n}(H)$. If such a ν exists, we update N as $\langle N, \nu\kappa \rangle$. The procedure above can be done in time $O((\frac{n}{p})!) = 2^{O(\frac{n}{p} \log \frac{n}{p})}$.
By the end of the procedure, we have $N \leq N_{S_n}(H)$. To show $N_{S_n}(H) \leq N$, let $g \in N_{S_n}(H)$. Then since $g \in L = B \rtimes K$, by Lemma 4.3.11, we can find $\kappa \in K$ and $\nu \in B$ such that $g = \nu\kappa$. If $\kappa = 1$, then $g \in N_B(H)$ and so $g \in N$. Suppose now that $\kappa \neq 1$. Using Proposition 5.2.18, we would have found an element $\nu' \in B$ such that $\nu'\kappa \in N_{S_n}(H)$, so $\nu'\kappa \in N$. Then $\nu\kappa(\nu'\kappa)^{-1} = \nu\nu'^{-1} \in N_B(H) \leq N$. Therefore $g = \nu\kappa \in N$. □

Theorem 5.0.1 now follows. Note that Proposition 5.2.3 gives better complexity than Theorem 5.2.22 when $p \leq k$. However, the algorithm for Proposition 5.2.3 requires the checking of all elements of $(\mathbb{F}_p^*)^m$, which we do not have useful pruning for. When p and m are large, this becomes infeasible (see Figure 5.2). Our implementation will instead follow the method described in Theorem 5.2.22.

5.3 Pruning techniques

Recall that we compute $N_{S_n}(H)$ using backtrack search. In this section, we present some pruning techniques which improve the performance of normaliser algorithms, arising from the view of $H \in \mathfrak{InP}(C_p)$ as a linear code over $\mathbb{F}_p$. We will give more details of how we apply these results to computing $N_{S_n}(H)$ in Section 5.4. By Lemma 5.2.2, the pruning tests in this section can be used on $H^\perp$ as well.

First, we show that the stabilisers and the equivalence of orbits can be computed using M. Recall $\equiv_o$ from Definition 2.1.10.

Lemma 5.3.1. *Let β be a point in a H-orbit Ω_j. Let $M \in M(s,k,\mathbb{F}_p)$ be a generator matrix for $\gamma(H)$. If $M_{*,j}$ has a unique position i with non-zero entry, then $\gamma(H_{(\beta)}) = \langle M_{\overline{s}\setminus i,*}\rangle$.*

Proof. $\geq$: By our assumption on $M_{*,j}$, we have $M_{\overline{s}\setminus i,j}$ is the zero vector, so $\gamma^{-1}(M_{\overline{s}\setminus i,*})$ fixes β.
$\leq$: Let $h \in H_{(\beta)}$. Then $\gamma(h)$ is a linear combination of rows of M and so there exists $a_1, a_2, \ldots, a_s \in \mathbb{F}_p$ such that $\gamma(h) = \sum_{i=1}^{s} a_i M_{i,*}$. Since $H|_{\Omega_i}$ is regular, $(H|_{\Omega_i})_{(\beta)} = 1$, so $\gamma(h)_i = 0$. Since $M_{i,*}$ is the only row of M with non-zero entry in the j-th position, $a_i = 0$, and so $\gamma(h) \in \langle M_{\overline{s}\setminus i,*}\rangle$. □

Note that the assumption on $M_{*,j}$ in the previous lemma can be achieved by performing row operations. Hence the stabiliser of any point can be computed this way.

Lemma 5.3.2. *Let M be the row-reduced generator matrix of $\gamma(H)$. Then $\Omega_i \equiv_o \Omega_j$ if and only if there exists $d \in \mathbb{F}_p^*$ such that $M_{*,j} = dM_{*,i}$.*

Proof. $\Leftarrow$: Recall the p-cycles g_i from Definition 5.1.2. Fix $\alpha \in \Omega_i$ and $\beta \in \Omega_j$. Define a mapping $\psi : \Omega_i \to \Omega_j$ by setting $\psi(\alpha^{g_i^t}) = \beta^{g_j^{dt}}$ for $0 \leq t \leq p-1$. Since $\langle g_i\rangle$ and $\langle g_j^d\rangle$ are isomorphic and regular, ψ is a bijection. We show that ψ satisfies Definition 2.1.10 and so $\Omega_i \equiv_o \Omega_j$.
Let $h \in H$ and $\delta \in \Omega_i$. Then $\delta = \alpha^{g_i^r}$ for some r. Let $\gamma(h)_i = v_i$. Then $\gamma(h)_j = dv_i$ and so $h|_{\Omega_i} = g_i^{v_i}$ and $h|_{\Omega_j} = g_j^{dv_i}$. Hence

$$\psi(\delta^h) = \psi(\delta^{g_i^{v_i}}) = \psi(\alpha^{g_i^{r+v_i}}) = \beta^{g_j^{d(r+v_i)}} = (\beta^{g_j^{dr}})^{g_j^{dv_i}} = \psi(\alpha^{g_i^r})^{g_j^{dv_i}} = \psi(\delta)^h.$$

$\Rightarrow$: By Lemma 2.1.12, there exists an involution $l \in S_n$ with support $\Omega_i \cup \Omega_j$ such that $h|_{\Omega_j} = (h|_{\Omega_i})^l$, for all $h \in H$. Let $h \in H$ and $v_i = \gamma(h)_i$ and $v_j = \gamma(h)_j$. Then $h|_{\Omega_i} = g_i^{v_i}$ and $h|_{\Omega_j} = g_j^{v_j}$. Then $(g_i^l)^{v_i} = (g_i^{v_i})^l = (h|_{\Omega_i})^l = h|_{\Omega_j} = g_j^{v_j}$. Let $g_i^l = g_j^d$. Then $g_i^{dv_i} = g_j^{v_j}$ and so $v_j = dv_i$. □

Hence we can determine if $\Omega_i \equiv_o \Omega_j$ by checking the equality of $M_{t_i,i}^{-1}M_{*,i}$ and $M_{t_j,j}^{-1}M_{*,j}$, where t_i and t_j are the first non-zero positions of $M_{*,i}$ and $M_{*,j}$ respectively. This can be done in time $O(sk + sk^2)$.

Definition 5.3.3. Two columns $M_{*,i}$ and $M_{*,j}$ of M are *equivalent* if $\Omega_i \equiv_o \Omega_j$. We denote this by $M_{*,i} \equiv_o M_{*,j}$.

Next, we show how we can use linearly dependent columns of M together with Lemma 2.4.16 to prune the search tree.

Definition 5.3.4. A set V of linearly dependent vectors are *minimally* linearly dependent if no proper subset $U \subset V$ is linearly dependent.

Notation 5.3.5. Let J be a subset of $\{1, 2, \ldots, k\}$. Denote the union $\cup_{j\in J}\Omega_j$ by Ω_J.

Lemma 5.3.6. *Let I and J be subsets of $\{1, 2, \ldots, k\}$. Let $g \in N_{S_n}(H)$ such that $\Omega_I^g = \Omega_J$. Then the column rank of $M_{*,I}$ is equal to the column rank of $M_{*,J}$. Hence*

1. *the columns of $M_{*,I}$ are linearly independent if and only if the columns of $M_{*,J}$ are linearly independent;*

2. *the columns of $M_{*,I}$ are minimally linearly dependent if and only if the columns of $M_{*,J}$ are minimally linearly dependent.*

Proof. Part 1: Let r_I and r_J be the column ranks of $M_{*,I}$ and $M_{*,J}$ respectively. Since $H|_{\Omega_I}$ and $H|_{\Omega_J}$ are conjugate in $\mathrm{Sym}(\Omega_I \cup \Omega_J)$, we have $p^{r_I} = |(H|_{\Omega_I})| = |(H|_{\Omega_J})| = p^{r_J}$, so $r_I = r_J$. The result follows since the columns of $M_{*,I}$ are linearly independent if and only if $|(H|_{\Omega_I})| = p^{|J|}$.
Part 2: We show that the columns of $M_{*,I}$ are minimally linearly dependent if and only if $|(H|_{\Omega_I})| = p^{|I|-1}$. Suppose that the columns of $M_{*,I}$ are minimally linearly dependent. Let $i \in I$. Since $M_{*,i}$ is a linear combination of columns of $M_{*,I\setminus i}$, we have $|\langle M_{*,I}\rangle| = |\langle M_{*,I\setminus i}\rangle|$. By the definition of a minimally dependent set, the columns of $M_{*,I\setminus i}$ are linearly independent, so $|(H|_{\Omega_I})| = |\langle M_{*,I}\rangle| = |\langle M_{*,I\setminus i}\rangle| = p^{|I|-1}$. □

Let C be a code. Let the *weight* $wt(c)$ of $c \in C$ be the number of $1 \le i \le k$ such that $c_i \neq 0$. and let w_i be the number of $c \in C$ of weight i. Then the *weight enumerator* of C is the polynomial $\sum_{i=1}^{k} w_i x^i$.

Lemma 5.3.7. 1. *Let $\Delta, \Gamma \subseteq \Omega$ be such that $|H_{(\Delta)}| = |H_{(\Gamma)}| = p^t$, say. Let $M_{(\Delta)}, M_{(\Gamma)} \in M(t, k, \mathbb{F}_p)$ be generator matrices of $H_{(\Delta)}$ and $H_{(\Gamma)}$ respectively. If $H_{(\Delta)}$ and $H_{(\Gamma)}$ are conjugate in S_n, then there exists a bijection δ between the $\equiv_o$-classes of columns of $M_{(\Delta)}$ and of columns of $M_{(\Gamma)}$ that preserves the size of the classes.*

2. *Let $Q, Q' \le H$ such that $Q' = Q^r$ for some $r \in S_n$. Then $\gamma(Q)$ and $\gamma(Q')$ have the same weight enumerator.*

3. *Let T be the set of minimum weight vectors in $\gamma(H)$. Then $N_{S_n}(H)$ setwise stabilises $\gamma^{-1}(T)$.*

Proof. Part 1: Follows from Lemma 4.1.1, and $\Omega_i \equiv_o \Omega_j$ if and only if $M_{*,i} \equiv_o M_{*,j}$.
Part 2: We show that the number of cycles in $h \in H$ is equal to the weight of $\gamma(h)$. Then the result follows since conjugation preserves cycle structures.
By the definition of γ, we have $\gamma(h)_i \neq 0$ if and only if $h|_{\Omega_i} \neq 1$. Since $h|_{\Omega_i}$ is either trivial or a p-cycle, $\gamma(h)_i \neq 0$ if and only if $h|_{\Omega_i}$ is a p-cycle.
Part 3: Let $\nu \in N_{S_n}(H)$. Since conjugation preserves cycle structure, for all $t \in T$, we have that$wt(\gamma(t^\nu)) = wt(\gamma(t))$, and so $\gamma^{-1}(t^\nu) \in \gamma^{-1}(T)$. □

By Lemma 2.4.9, the search tree can be pruned by determining when two stabilisers are not conjugate in S_n. Parts 1 and 2 show how we can decide the non-conjugacy of stabilisers. Note that since there is no known polynomial-time algorithm for computing the weight enumerator, we use a simple heuristic to determine when we use Part 2 (see Algorithm 10). We find a set stabilised by $N_{S_n}(H)$ using Part 3, which can be used to further reduce our search tree (see Algorithm 7).

Lastly, we give a technical lemma which can be used to prune the search tree at depth $m > s$, where $|H| = p^s$.

Lemma 5.3.8. *Let $M \in M(s, k, \mathbb{F}_p)$ be a generator matrix of $\gamma(H)$ in standard form. Fix $m \in \{s, \ldots, k\}$ and let $J = \{1, \ldots, m\} \dot\cup \{t\} \subseteq \{1, 2, \ldots, k\}$. Let $f : J \to \{1, 2, \ldots, k\}$ be an injection. If there exists $1 \leq i \leq s$ such that*

$$M_{i,t^f} \neq 0 \text{ and } M_{i,j^f} M_{j,t} = 0 \quad \text{for all } j \in J \cap \{1, 2, \ldots, s\}, \tag{5.5}$$

then there does not exist $\eta \in N_{S_n}(H)$ such that $\Omega_j^\eta = \Omega_{j^f}$ for all $j \in J$.

Proof. Aiming for a contradiction, assume that such an i and η exist. Then by Part 4 of Lemma 5.1.13, $\Xi(\eta) \in \mathrm{MAut}(\gamma(H))$. So there exist $d = diag(v_1, v_2, \ldots, v_k) \in D$ and $q \in P$ such that $\Xi(\eta) = dq$. Let $M' = Mdq$. Then as in Definition 5.1.10, for all $j \in J$ and $1 \leq i \leq s$,

$$M'_{i,j} = v_{j^f} M_{i,j^f}. \tag{5.6}$$

In particular, $M'_{i,t} = v_{t^f} M_{i,t^f}$, which is non-zero.
Since $dq \in \mathrm{MAut}(M)$, each row of M' is a linear combinations of the rows of M. Since M is in standard form, $M'_{i,*} = \sum_{u=1}^{s} M'_{i,u} M_{u,*}$. Then

$$M'_{i,*} = \sum_{u=1}^{s} M'_{i,u} M_{u,*} = \sum_{u=1}^{s} v_{u^f} M_{i,u^f} M_{u,*},$$

where the second equality follows from Equation (5.6). Therefore by Equation (5.5), $M'_{i,t} = \sum_{u=1}^{s} v_{u^f} M_{i,u^f} M_{u,t} = 0$, a contradiction. □

The rationale behind Lemma 5.3.8 is as follows. If there exists a $\kappa \in K$ represented by a leaf under the current node at depth $m > s$ such that $\nu\kappa \in N_{S_n}(H)$ for some $\nu \in B$, then there exists $d \in D$ such that $d\rho(\kappa) \in \mathrm{MAut}(\gamma(H))$. Let $M' := Md\rho(\kappa)$.

Then we know, up to s unknown scalars from $\mathbb{F}_p^*$, the first s columns of M'. Since rows of M' are linear combinations of the rows of M and the elements of $\langle M\rangle$ are defined by their first s coordinates, we now know the whole of M', up to s unknown scalars. So if we can deduce that some entries of M' must be zero, then we may prune the search tree.

5.4 Algorithm

Given a generating set X of $H \leq S_n$ such that $H \in \mathfrak{InP}(C_p)$, we compute $N_{S_n}(H)$ using Algorithm 6. The algorithm roughly follows that described in Theorem 5.2.22, with extra steps for the pruning of the search tree.

The overview of the algorithm is as follows. We first represent H and $H^\perp$ by generator matrices M and $M^\perp$ respectively over $\mathbb{F}_p$. Then we compute $C_{S_n}(H)$, which may allow us to compute the normaliser of a group with a smaller degree, as in Proposition 4.1.6. We then compute $N_B(H)$, where B is as in Proposition 5.1.8. That is, we compute all normalising elements fixing the H-orbits. Lastly, we perform backtrack search in $K \cong S_k$ to find the remaining normalising elements.

For this section, we shall assume that $\gamma(H)$ has dimension $s \leq k/2$. To simplify notation, we shall also assume that H has no equivalent orbits. See Proposition 4.1.6 for the reduction to this case.

Algorithm 6 Computing the normaliser of $H \in \mathfrak{InP}(C_p)$

Input: A generating set X of $H \leq S_n$, where $H \in \mathfrak{InP}(C_p)$ and H has no equivalent orbits.
Output: $N_{S_n}(H)$

```
1: Check that $H \in \mathfrak{InP}(C_p)$                          ▷ polynomial by Lemma 5.1.1
2: Compute orbits $\mathcal{O} = \{\Omega_1, \Omega_2, \ldots, \Omega_k\}$ of $H$      ▷ polynomial by Proposition 2.1.2
3: Compute the enveloping group $G$ of $H$                ▷ polynomial by Lemma 5.1.3
4: Let $\gamma : G \to \mathbb{F}_p^k$ be as in Definition 5.1.4
5: Initialise $N \leftarrow G$                                      ▷ $C_{S_n}(H) = G$
6: Let $M$ be a generator matrix of $\gamma(H)$ in standard form
7: Let $M^\perp$ be a generator matrix of $\gamma(H^\perp)$     ▷ where $H^\perp$ is as in Definition 5.2.1
8: $N \leftarrow \langle N, N_B(H)\rangle$                               ▷ using Proposition 5.2.21
9: $domains \leftarrow domainsInit(M, M^\perp)$                      ▷ see Algorithm 7
10: $LDScols \leftarrow \{\{i\} \cup \{1 \leq j \leq s \mid M_{j,i} \neq 0\} \mid s+1 \leq i \leq k\}$
11: $dualLDScols \leftarrow \{\{i\} \cup \{k-s \leq j \leq k \mid (M^\perp)_{j,i} \neq 0\} \mid 1 \leq i \leq k-s\}$
12: RECURSESEARCH([ ])                                  ▷ see Algorithm 8
13: return $N$
```

All algorithms are implemented in GAP, apart from MAKEINVARIANTSETPARTN in Algorithm 7, which is implemented in C++, as it is time critical[1].

[1] This implementation of MAKEINVARIANTSETPARTN in C++ is done by Dr Christopher Jefferson.

Pre-search

We shall describe the computation before the backtrack search. By the end of step 11, N contains $N_B(H) \geq C_{S_n}(H)$. So as in Theorem 5.2.22, this allows us to only search for non-trivial elements $\kappa \in K$ which give rise to elements of $N_{S_n}(H)$. We will also be computing various structures we shall use later in the search.

We start by describing lines 1–11 of Algorithm 6 in more detail:

Lines 1–4 We set up the map γ which enable us to compute $N_{S_n}(H)$ by computing $\mathrm{MAut}(\gamma(H))$, as in Theorem 5.1.15.

Line 5 We gather the normalising elements we find as N. We start by initialising $N = G$. Since we assume that H has no equivalent orbits, N contains $C_{S_n}(H)$ by Proposition 2.1.13.

Lines 6–7 We represent H as a code over $\mathbb{F}_p$. We compute the a generator matrix M of $\gamma(H)$ using X. Since H has no equivalent orbits, the row-reduced matrix M is in standard form. We shall be using $H^\perp$ as in Definition 5.2.1. We can compute the generator matrix $M^\perp$ of $\gamma(H)^\perp$ as follows. If $M = (I_s \mid M_0)$ for some $M_0 \in M(k-s, s, \mathbb{F}_p)$, then $M^\perp := (M_0^T \mid I_{k-1})$ is a generator matrix of $\gamma(H)^\perp$ [vL99]. We will use $\gamma(H)^\perp$ to refine our search via Lemma 5.2.2.

Line 8 We first compute the finest disjoint direct product decomposition of H, as in Proposition 5.2.12. Then as in Proposition 5.2.21, we compute $N_B(H)$. We update N as $\langle N, N_B(H)\rangle$.

Line 9 For each H-orbit Ω_i, we compute a set of possible images of Ω_i under $N_{S_n}(H)$, called the *domain* of Ω_i. This gives an approximation of $N_{S_n}(H)$ and will be used to guide our search in Algorithm 8. We use various methods to restrict the initial domains. More details will be given later in Algorithm 7.

lines 10–11 We compute certain sets of linearly dependent columns of M and $M^\perp$. We shall explain how we use these sets for pruning later. Since M is assumed to be in standard form, the standard basis of $\mathbb{F}_p^s$ forms the first s columns of M. Therefore we can write each later column of M as a linear combination of these s columns. More specifically, each element of *LDScols* is the column positions of a linearly dependent set formed by column $M_{*,i}$ for $s+1 \leq i \leq k$ and a subset of the first s columns of M.

As noted in lines 6–7, the standard basis of $\mathbb{F}_p^{k-s}$ forms the last $k-s$ columns of $M^\perp$.

Algorithm 7: Initialising the domains

We describe how we initialise the domains using Algorithm 7. We will be computing several partitions of $\mathcal{O}$, each preserved by $N_{S_n}(H)$. Hence the meet $\mathcal{P}^*$ of these partitions is also preserved by $N_{S_n}(H)$. We update the domains by setting the domain of Ω_i to be the cell of $\mathcal{P}^*$ containing it.

makeDualEquivPartn We first find the equivalent orbits of $H^\perp$ using Lemma 5.3.2. In line 12, we construct certain elements g centralising $H^\perp$, as in Proposition 2.1.13. Using Lemmas 4.3.11 and 5.2.2, for each g we construct a corresponding element of $N_{S_n}(H)$. Next, we compute a partition $\mathcal{P}_e^\perp$ of $\mathcal{O}$ based on sizes of the equivalence classes for $H^\perp$. By Lemma 4.1.1, $N_{S_n}(H^\perp)$ preserves $\mathcal{P}_e^\perp$. By Lemma 5.2.2, $N_{S_n}(H)$ also preserves $\mathcal{P}_e^\perp$. Note that in the algorithm, we store $\mathcal{P}_e^\perp$ by a partition of the indices $\{1, 2, \ldots, k\}$.

makeStabilisersPartn This uses Lemma 2.4.9 to further refine the initial domains. That is, if there exists $g \in N_{S_n}(H)$ such that $\Omega_i^g = \Omega_j$, then by Part 1 of Lemma 5.3.7, there exists a bijection from the equivalence classes of $\equiv_o$ over the $H_{(\Omega_i)}$-orbits to the equivalence classes of $\equiv_o$ over the $H_{(\Omega_j)}$-orbits which preserves the class sizes. By Lemma 5.3.1, the matrices representing the stabilisers can be computed efficiently. Using Lemma 5.2.2, this algorithm also works using the dual $H^\perp$.

makeInvariantSetPartn Here we compute the set *invSet* of minimal weight vectors of $\gamma(H)$. By Part 3 of Lemma 5.3.7, $N_{S_n}(H)$ stabilises *invSet*. The algorithm runs in exponential time by systematically considering linear combinations with increasing numbers of non-zero coefficients of rows of M. The variable m gives the minimal weight of all the vectors we have encountered so far. Since $M_{\overline{s},\overline{s}}$ is the identity matrix, if v is a linear combination with non-zero coefficients of i rows of M, then $wt(v) \geq i$. Since we are only interested in vectors with weight at most m, we only need to consider all such linear combinations v of up to m rows of M. Lastly, we obtain a partition of $\Omega_1, \Omega_2, \ldots, \Omega_k$ stabilised by $N_{S_n}(H)$ by considering the number of vectors in *invSet* with non-zero entries in each Ω_i.

Therefore, the normaliser $N_{S_n}(H)$ preserves the partitions $\mathcal{P}_e^\perp$, $\mathcal{P}_s$, $\mathcal{P}_s^\perp$, $\mathcal{P}_I$ and hence $\mathcal{P}^*$. The function domainsInit returns a list of sets *domains*, where each entry *domains*$[i]$ is a subset of $\{1, 2 \ldots, k\}$ such that if there exists $g \in N_{S_n}(H)$ with $\Omega_i^g = \Omega_j$ then $j \in$ *domains*$[i]$.

Search

Now we shall describe the recursive search, Algorithm 8. We shall traverse the search tree depth first, using the domains to guide our search. Note that N is the global variable which stores the group generated by all elements of $N_{S_n}(H)$ we have found so far.

Line 2 An orbit Ω_j will be removed from the domain of Ω_i if we show that there are no normalising elements under the current node that map Ω_i to Ω_j. When we

Algorithm 7 Initialise domains

```
 1: procedure DOMAINSINIT($M, M^{\perp}$)
 2:     $\mathcal{P}_e^{\perp} \leftarrow$ MAKEDUALEQUIVPARTN($M^{\perp}$)
 3:     $\mathcal{P}_s \leftarrow$ MAKESTABILISERSPARTN($M$)
 4:     $\mathcal{P}_s^{\perp} \leftarrow$ MAKESTABILISERSPARTN($M^{\perp}$)
 5:     $\mathcal{P}_I \leftarrow$ MAKEINVARIANTSETPARTN($M$)
 6:     $\mathcal{P}^* \leftarrow Meet(\mathcal{P}_e^{\perp}, \mathcal{P}_s, \mathcal{P}_s^{\perp}, \mathcal{P}_I)$
 7:     return [cell of $\mathcal{P}^*$ containing $i \mid 1 \leq i \leq k$]
 8: end procedure

 9: procedure MAKEDUALEQUIVPARTN($M^{\perp}$)
10:     Compute the $\equiv_o$-classes of the $H^{\perp}$-orbits        ▷ using Lemma 5.3.2
11:     for each pair of equivalent orbits $\Omega_i$ and $\Omega_j$ do
12:         Let $g \in C_{S_n}(H^{\perp})$ conjugate $\Omega_i$ to $\Omega_j$        ▷ as in Proposition 2.1.13
13:         Find $\nu \in B$ and $\kappa \in K$ such that $g = \nu\kappa$        ▷ using Lemma 4.3.11
14:         $N \leftarrow \langle N, \nu^{-1}\kappa \rangle$        ▷ $\nu^{-1}\kappa \in N_{S_n}(H)$ by Lemma 5.2.2
15:     end for
16:     $\mathcal{P}_e^{\perp} \leftarrow$ partition of $\{1, 2, \ldots, k\}$ such that $i, j$ are in the same cell if and only if
                $|[\Omega_i]_{\equiv_o}| = |[\Omega_j]_{\equiv_o}|$
17:     return $\mathcal{P}_e^{\perp}$
18: end procedure

19: procedure MAKESTABILISERSPARTN($mat$)        ▷ $mat$ is $M$ or $M^{\perp}$
20:     for $i \in [1, 2, \ldots, k]$ do
21:         $stab_i \leftarrow \gamma^{-1}(\langle mat \rangle)_{(\Omega_i)}$        ▷ using Lemma 5.3.1
22:         Compute the $\equiv_o$-classes of the $stab_i$-orbits        ▷ using Lemma 5.3.2
23:     end for
24:     $\mathcal{P}_s \leftarrow$ partition of $\{1, 2, \ldots, k\}$ such that $i, j$ are in the same cell if and only if
                $stab_i$ and $stab_j$ have same multiset of the sizes of the $\equiv_o$-classes
25:     return $\mathcal{P}_s$
26: end procedure

27: procedure MAKEINVARIANTSETPARTN($M$)
28:     $m \leftarrow min_{1 \leq i \leq s}(wt(M_{i,*}))$ ▷ minimum weight of codewords we have found so far
29:     $invSet \leftarrow \{i \mid wt(M_{i,*}) = m\}$        ▷ minimum weight codewords found so far
30:     for $i \in [2, 3, \ldots, m]$ do
31:         for all linear combinations $v$ with non-zero coefficients of $i$ rows of $M$ do
32:             if $wt(v) < m$ then        ▷ finds a vector with smaller weight
33:                 $m \leftarrow wt(v)$, $invSet \leftarrow \{v\}$
34:             else if $wt(v) = m$ then        ▷ finds a minimum weight vector
35:                 Add $v$ to $invSet$
36:             end if
37:         end for
38:     end for
39:     $\mathcal{P}_I \leftarrow$ partition of $\{1, 2, \ldots, k\}$ such that $i, j$ are in the same cell if and only if
                $|\{v \in invSet \mid v[i] \neq 0\}| = |\{v \in invSet \mid v[j] \neq 0\}|$
40:     return $\mathcal{P}_I$
41: end procedure
```

Algorithm 8 RECURSESEARCH

```
 1: procedure RECURSESEARCH($\underline{\alpha} = [\alpha_1, \alpha_2, \dots, \alpha_d], indomains$)
 2:     $domains \leftarrow copy(indomains)$
 3:     if d=k then
 4:         $\kappa \leftarrow$ permutation in $K$ such that $\Omega_i^\kappa = \Omega_{\alpha_i}$ for all $1 \le i \le k$
 5:         if there exists $\nu \in B$ s.t. $\nu\kappa \in N_{S_n}(H)$ then        ▷ using Proposition 5.2.18
 6:             $N \leftarrow \langle N, \nu\kappa \rangle$
 7:             Backtrack to node $[\alpha_1, \alpha_2, \dots, \alpha_j]$, where $j$ is the largest integer such that
                  $\alpha_i = i$ for $1 \le i \le j$, if it exists        ▷ as in Algorithm 2
 8:         end if
 9:     else
10:         if $[\alpha_1, \alpha_2, \dots, \alpha_{d-1}] = [1, 2 \dots, d-1]$ and $\alpha_d \neq d$ and $\alpha_d$ is not the minimum
                  value in orbit $\alpha_d^{N_{(\alpha_1,\alpha_2,\dots,\alpha_{d-1})}}$ then return end if
11:         $passedTests, domains \leftarrow$ CHECKLDS($M, LDScols, \underline{\alpha}, domains$)
                                                                  ▷ see Algorithm 9
12:         if $\neg passedTests$ then return end if
13:         $passedTests, domains \leftarrow$ CHECKLDS($M^\perp, dualLDScols, \underline{\alpha}, domains$)
14:         if $\neg passedTests$ then return end if
15:         $Mstab \leftarrow \gamma(H_{(\Omega_1,\dots,\Omega_d)})$        ▷ using Lemma 5.3.1
16:         $MstabIm \leftarrow \gamma(H_{(\Omega_{\alpha_1},\dots,\Omega_{\alpha_d})})$
17:         $passedTests, domains \leftarrow$ COMPARESTABS($Mstab, MstabIm, \underline{\alpha}, domains$)
                                                                  ▷ see Algorithm 10
18:         if $\neg passedTests$ then return end if
19:         $Mstab^\perp \leftarrow$ generator matrix of $\gamma((H^\perp)_{(\Omega_1,\dots,\Omega_d)})$        ▷ using Lemma 5.3.1
20:         $MstabIm^\perp \leftarrow$ generator matrix of $\gamma((H^\perp)_{(\Omega_{\alpha_1},\dots,\Omega_{\alpha_d})})$
21:         $passedTests, domains \leftarrow$ COMPARESTABS($Mstab^\perp, MstabIm^\perp, \underline{\alpha}, domains$)
22:         if $\neg passedTests$ then return end if
23:         if $d > s$ then
24:             for $1 \le i \le s$ and $t \in [s+1 \le t \le k \mid M_{i,\alpha_u} M_{u,t} = 0]$ do
25:                 $domains[t] \leftarrow [j \in domains[t] \mid M_{i,j} = 0]$        ▷ using Lemma 5.3.8
26:             end for
27:         end if
28:         $domains \leftarrow allDiffRefiner(domains)$
29:         if $\exists i : domains[i] = \emptyset$ then return end if
30:         for $\alpha_{d+1} \in domains[d+1]$ do
31:             RECURSESEARCH($[\alpha_1, \alpha_2, \dots, \alpha_{d+1}], domains$)
32:         end for
33:     end if
34: end procedure
```

backtrack, we shall revert to *indomains*.

Lines 3–9 If $d = k$, then we have arrived at a leaf node of the search tree. Using Proposition 5.2.18, we determine if there exists $\nu \in B$ such that $\nu\kappa \in N_{S_n}(H)$. If such a ν exists, we update N as $\langle N, \nu\kappa\rangle$. To ensure that we only add a generating set of $N_{S_n}(H)$ to N, we find the largest j such that $i = \alpha_i$ for all $1 \leq i \leq j$, and backtrack to node $[\alpha_1, \alpha_2, \ldots, \alpha_{j-1}]$, as in Lemma 2.3.5.

Line 10 This uses Lemma 2.3.6, where for each node $[1, 2, \ldots, d-1, \alpha_d]$ such that $d \neq \alpha_d$, we ensure that α_d is the minimum value of the orbit of α_d in $N_{(\Omega_1,\Omega_2,\ldots,\Omega_{d-1})}$.

Lines 11–22 Here, we perform four additional pruning tests to detect if no normalising elements are arising from a leaf under the current node. We only do further tests if we pass all the tests so far. When appropriate, we also refine our domains. More details on the pruning tests will be given later.

Lines 23–27 This uses Lemma 5.3.8 to prune at depth $m > s$. If there exists $j \in$ *domains*$[t]$ such that $M_{i,j} \neq 0$, then by Lemma 5.3.8, there are no normalising element under the current node which maps Ω_t to to Ω_j. Hence we may remove j from *domains*$[t]$.

Lines 28&29 Since the images of $\mathcal{O}$ must be pairwise different, whenever the domains are changed, we invoke a simple procedure to further refine the domains. For $1 \leq i \leq k$, let $J_i := \{1 \leq j \leq k \mid \mathit{domains}[j] = \mathit{domains}[i]\}$. If $|J_i| = |\mathit{domains}[i]|$, then any normalising element arising under the current node must map $\{\Omega_i \mid i \in J_i\}$ to $\{\Omega_i \mid i \in \mathit{domains}[i]\}$. We remove elements of *domains*$[i]$ from all other *domains*$[l]$ where $l \notin J_i$. That is, for all $l \notin J_i$, we set $\mathit{domains}[l] = \mathit{domains}[l] \backslash \mathit{domains}[i]$. More advanced refinements exist but are not considered in this thesis. For more details, see, for example, [GMN08].

Lines 30–32 If the pruning tests result in empty domains, we backtrack. Otherwise, we continue the depth first search, branching using *domains*. Note that we may also use Lemma 2.3.6 to reduce the partial base images we consider.

Now we describe the pruning tests used in Algorithm 8. These methods use Algorithms 9 and 10.

CHECKLDS See Algorithm 9. This uses Lemma 2.4.16 to prune the search tree. For each linearly dependent set in `LDScols`, if all but one column in the set have been assigned with an image, we use Lemma 5.3.6 to prune the domain of remaining column. To do this, we consider each column in the domain and check if it is in the span of the other columns using the GAP function SolutionMat. If not, by Lemma 5.3.6, we may remove the column from the domain.

Algorithm 9 CHECKLDS

```
procedure CHECKLDS($M, LDScols, [\alpha_1, \alpha_2, \ldots, \alpha_d], domains$)
    for $lds \in LDScols$ do                          ▷ see Algorithm 6, line 10
        $I \leftarrow lds \backslash \{1, 2, \ldots, d\}$        ▷ unassigned column images
        if $|I| = |lds| - 1$ then    ▷ image of $M_{*,I[1]}$ must be in the span of other columns
            for $i \in domains[I[1]]$ do
                if $M_{*,i} \notin \langle M_{*,\alpha_j} \mid j \in lds \backslash I[1] \rangle$ then
                    Remove $i$ from $domains[I[1]]$
                        ▷ no normalising element under current node that sends $\Omega_{I[1]}$ to $\Omega_i$ by Lemma 5.3.6
                end if
            end for
        end if
    end for
    return TRUE, domains
end procedure
```

COMPARESTABS See Algorithm 10. This uses Lemma 2.4.9 to prune our search tree. As in Lemma 5.3.2, we find the orbit equivalence classes E and E' over the stabilisers. As in Part 1 of Lemma 5.3.7, we test whether there is a bijection between the sizes of E and E'. If not, we backtrack. Otherwise, we refine the domain using Lemma 4.1.1 by ensuring that orbits in a sized e equivalence class in E can only map to orbits in a sized e equivalence class in E'. As in Part 2 of Lemma 5.3.7, we can also compare the weight enumerator of the subspaces representing the stabilisers. Since weight enumerator computation takes exponential time, we only do this if the stabiliser is sufficiently small, using a simple heuristics.

5.5 Extension: Groups in class $\mathfrak{InP}(D_{2p})$

In this section, we shall consider $H \in \mathfrak{InP}(D_{2p})$, where D_{2p} is the dihedral group of order $2p$ with odd prime degree p. We shall show that $N_{S_n}(H)$ can be found by computing the normalisers of two of its Sylow subgroups, which can be identified with groups in classes $\mathfrak{InP}(C_p)$ and $\mathfrak{InP}(C_2)$ respectively. This yields a faster algorithm to compute the normalisers of such groups.

Notation 5.5.1. Let $n = pk$. Let $\Omega_1, \Omega_2, \ldots, \Omega_k$ be disjoint sets such that $\Omega = \cup_{i=1}^{k} \Omega_i = \{1, 2, \ldots, n\}$. For $1 \leq i \leq k$, let $D_i \leq \mathrm{Sym}(\Omega_i)$ be a permutation representation of D_{2p} and let G_i be the unique subgroup of D_i isomorphic to C_p. Let $D = D_1 \times D_2 \times \ldots \times D_k \leq S_n$ and $G = G_1 \times G_2 \times \ldots \times G_k \leq S_n$. Let H be a subdirect product of D.

The next two results describe some properties of the Sylow subgroups of H.

Lemma 5.5.2. *Let H_p be a Sylow p-subgroup of H. Then*

Algorithm 10 COMPARESTABS

```
procedure COMPARESTABS($Mstab, MstabIm, [\alpha_1, \alpha_2, \ldots, \alpha_d], domains$)
    Compute the $\equiv_o$-classes $E_1, E_2, \ldots, E_c$ of the columns of $Mstab$
                                                          ▷ using Lemma 5.3.2
    Compute the $\equiv_o$-classes $E'_1, E'_2, \ldots, E'_{c'}$ of the columns of $MstabIm$
                                                          ▷ using Lemma 5.3.2
    if multisets of class sizes are different then
                          ▷ no normalising elements under the current node by Lemma 5.3.7.1
        return FALSE, $domains$
    end if
    for $i \in \{1, 2, \ldots, k\} \backslash \{\alpha_1, \alpha_2, \ldots, \alpha_d\}$ do
        Get $E_u$ such that $\Omega_i \in E_u$
        for $j \in domains[i]$ do
            Get $E'_{u'}$ such that $\Omega_j \in E'_{u'}$
            if $|E_u| \neq |E'_{u'}|$ then   ▷ no normalising elements mapping $\Omega_i$ to $\Omega_j$ by Lemma 4.1.1
                Remove $j$ from $domains[i]$
            end if
        end for
    end for
    if $(s-d) * p \leq 45$ then                                ▷ 45 is a heuristic
        if $WeightEnumerator(Mstab) \neq WeightEnumerator(MstabIm)$ then
                          ▷ $\langle\gamma^{-1}(Mstab)\rangle, \langle\gamma^{-1}(MstabIm)\rangle$ are not conjugate by Lemma 5.3.7.2
            return FALSE, $domains$
        end if
    end if
    return TRUE, $domains$                   ▷ no obstruction to conjugacy found
end procedure
```

1. *$H_p = H \cap G \trianglelefteq H$, and so $N_{S_n}(H) \leq N_{S_n}(H_p)$.*

2. *H_p is a subdirect product of G, and so $H_p \in \mathfrak{InP}(C_p)$.*

Proof. Part 1: Since $H \cap G$ is a p-group, $H \cap G \leq H_p$. As G is the unique Sylow p-subgroup of D, it follows that $H_p \leq G$ and so $H_p \leq H \cap G$. From $G \trianglelefteq D$, we deduce that $H_p \trianglelefteq H$. As H has a unique Sylow p-subgroup, H_p is characteristic in H. Hence $H_p \lhd N_{S_n}(H)$, and so $N_{S_n}(H) \leq N_{S_n}(H_p)$.
Part 2: Certainly $H_p \leq G$ and so $H_p|_{\Omega_i} \leq G_i$ for all $1 \leq i \leq k$. Since G_i is cyclic, to show that $H_p|_{\Omega_i} = G_i$, it suffices to show that there exists $t \in H_p$ such that $t|_{\Omega_i} \neq 1$.
Since H is a subdirect product of D and $G_i \leq D_i$, there exists $h \in H$ such that $1 \neq h|_{\Omega_i} \in G_i$. We show that $h^2 \in H_p$.
Since $h|_{\Omega_i}$ is a p-cycle, $h^2|_{\Omega_i} \neq 1$. For all j, the element $h|_{\Omega_j}$ of D_j is of order $1, 2$ or p. So $h^2|_{\Omega_j}$ is of order 1 or p. Since G_j is the unique p-subgroup of D_i, $h^2|_{\Omega_j} \in G_j$ for all j, so $h^2 \in G \cap H = H_p$. □

Lemma 5.5.3. *Fix a Sylow 2-subgroup H_2 of H. Then*

1. *there exist $\alpha_i \in \Omega_i$ for all $1 \leq i \leq k$ such that $H_2 = H_{(\alpha_1,\alpha_2,\ldots,\alpha_k)}$.*

2. *$H = H_p \rtimes H_2$.*

3. *H_2 is a subdirect product of $(D_1)_{(\alpha_1)} \times (D_2)_{(\alpha_2)} \times \ldots \times (D_k)_{(\alpha_k)}$.*

Proof. Part 1: $\leq$: Suppose that there exist $s_1, s_2 \in H_2$ such that $s_1|_{\Omega_i}$ and $s_2|_{\Omega_i}$ are distinct and non-trivial. Then $\langle s_1|_{\Omega_i}, s_2|_{\Omega_i}\rangle = D_i$ and so $H_2|_{\Omega_i} = D_i$, a contradiction. So there exists an involution $t_i \in D_i$ such that $\langle t_i \rangle \geq H_2|_{\Omega_i}$. Since every involution in D_i fixes a point, there exists $\alpha_i \in \Omega_i$ such that t_i fixes α_i. Then $H_2|_{\Omega_i} \leq (D_i)_{(\alpha_i)}$. Hence, $H_2 \leq (D_1)_{(\alpha_1)} \times (D_2)_{(\alpha_2)} \times \ldots \times (D_k)_{(\alpha_k)}$. As $H_2 \leq H$, the result follows.
$\geq$: For $1 \leq i \leq k$, the group $(H_{(\alpha_1,\alpha_2,\ldots,\alpha_k)})|_{\Omega_i} \leq (D_i)_{(\alpha_i)}$, which is a 2-group. So $H_{(\alpha_1,\alpha_2,\ldots,\alpha_k)}$ is a 2-group and the result follows.
Part 2: By Part 1 of Lemma 5.5.2, $H_p \trianglelefteq H$, so $H_pH_2 \leq H$. From $H_p \cap H_2 = 1$, we see that

$$|H_pH_2| = \frac{|H_p||H_2|}{|H_p \cap H_2|} = |H|.$$

So $H = H_pH_2 = H_p \rtimes H_2$. Part 3: Let $1 \leq i \leq k$. Then by Part 1, $H_2|_{\Omega_i} = (D_i)_{(\alpha_i)}$ or $H_2|_{\Omega_i} = 1$. If H_2 fixes Ω_i, then by Part 2,

$$H|_{\Omega_i} = (H_pH_2)|_{\Omega_i} = H_p|_{\Omega_i} H_2|_{\Omega_i} = H_p|_{\Omega_i} = G_i,$$

which contradicts the fact that H is a subdirect product of D. □

We will show, in Proposition 5.5.6, how we can compute $N_{S_n}(H)$ using normalisers of Sylow subgroups of H. We will be using the following lemma, which is also known as Frattini argument.

Lemma 5.5.4. *Let S be a normal subgroup of a group T and let U be a Sylow p-subgroup of S. Then $T = N_T(U)S$.*

For the rest of the section, assume the following.

Notation 5.5.5. Let H_p and H_2 be the Sylow p-subgroup and a Sylow 2-subgroup of H respectively. For $2 \leq i \leq k$, let $\phi_i : \Omega_1 \to \Omega_i$ be a bijection witnessing the permutation isomorphism from D_1 to D_i, and satisfying $\phi_i(\alpha_1) = \alpha_i$, where α_i is stabilised by H_2. Using Notation 2.1.11, let $K = \langle \overline{\phi_i} \mid 2 \leq i \leq k\rangle$. Let $L = \langle N_{\mathrm{Sym}(\Omega_1)}(G_1), N_{\mathrm{Sym}(\Omega_2)}(G_2), \ldots, N_{\mathrm{Sym}(\Omega_k)}(G_k), K\rangle$.
Fix an orbit Ω_{11} of $H_2|_{\Omega_1}$. For each $2 \leq i \leq k$, let $\Omega_{i1} := \Omega_{11}^{\overline{\phi_i}}$ be a $(H_2|_{\Omega_i})$-orbit. Let $\Gamma = \bigcup_{i=1}^{k} \Omega_{i1}$.

Proposition 5.5.6. *Let H, H_p, H_2 and L be as in Notation 5.5.5. Then $N_{S_n}(H) = (N_L(H_p) \cap N_L(H_2))H \leq L$.*

Proof. By Part 1 of Lemma 5.5.2, $N_{S_n}(H) \leq N_{S_n}(H_p)$, then by Proposition 4.3.10, $N_{S_n}(H) \leq L$, so it suffices to show that $N_{S_n}(H) = (N_L(H_p) \cap N_L(H_2))H$.

We first show that $N_{S_n}(H) \leq (N_L(H_p) \cap N_L(H_2))H$. Taking S, T and U in Lemma 5.5.4 as H, $N_L(H)$ and H_2 respectively,

$$N_L(H) = N_{N_L(H)}(H_2)H = (N_L(H) \cap N_L(H_2))H.$$

It follows from Part 1 of Lemma 5.5.2 that $N_L(H) \leq N_L(H_p)$, which gives the result.
We now show that $N_{S_n}(H) \geq (N_L(H_p) \cap N_L(H_2))H$. Certainly, $H \leq N_{S_n}(H)$. Let $\nu \in N_L(H_p) \cap N_L(H_2)$ and let $h \in H$. By Part 2 of Lemma 5.5.3, there exist $a \in H_p$ and $b \in H_2$ such that $h = ab$. Since $\nu \in N_L(H_p)$, we have $a^\nu \in H_p$. Similarly, $b^\nu \in H_2$ and so $h^\nu = a^\nu b^\nu \in H_p H_2 = H$. Hence $\nu \in N_{S_n}(H)$. □

Therefore, to compute $N_{S_n}(H)$, we compute $N_L(H_p) \cap N_L(H_2)$. We will show how we compute $N_L(H_p) \cap N_L(H_2)$ in Lemma 5.5.9. Before that, we first give a group T containing $N_L(H_p) \cap N_L(H_2)$.

Lemma 5.5.7. *For $1 \leq i \leq k$, let $N_i = N_{\mathrm{Sym}(\Omega_i)}(H_2|_{\Omega_i}) \cap N_{\mathrm{Sym}(\Omega_i)}(H_p|_{\Omega_i})$. Let K and Γ be as in Notation 5.5.5, and let $T := \langle N_1, N_2, \ldots, N_k, K\rangle$.*

1. *$N_L(H_p) \cap N_L(H_2) \leq T$ and so $N_L(H_p) \cap N_L(H_2) = N_T(H_2) \cap N_T(H_p)$.*
2. *Let $r \in \mathbb{F}_p^*$ be primitive. For $1 \leq i \leq k$, let $g_i \in \mathrm{Sym}(\Omega_i)$ be a generator of G_i, and let $c_i \in \mathrm{Sym}(\Omega_i)$ with support $\Omega_i \backslash \{\alpha_i\}$ such that $g_i^{c_i} = g_i^r$. Then $N_i = \langle c_i\rangle$ and so $T \cong \langle c_1\rangle \wr S_k$.*

Proof. Part 1: Let $l \in N_L(H_p) \cap N_L(H_2)$. Since $L \cong N_{\mathrm{Sym}(\Omega_1)}(G_1) \wr S_k$ by Lemma 4.3.8, there exists $\kappa \in K$ such that $l\kappa^{-1}$ fixes each Ω_i setwise. Then $(l\kappa^{-1})|_{\Omega_i} \in N_i$ and so $l \in T$.
Part 2: We first show that $N_i \langle c_i\rangle$. By Proposition 5.1.8, we have $N_{\mathrm{Sym}(\Omega_i)}(G_i) = \langle g_i, c_i\rangle$, so $N_i \leq \langle g_i\rangle \rtimes \langle c_i\rangle$. Assume that there exists $gc \in \langle g_i, c_i\rangle$ such that $gc \in N_i$ with $1 \neq g \in \langle g_i\rangle$ and $c \in \langle c_i\rangle$. Since g moves α_i and c only fixes α_i, we have $\alpha_i^{gc} \neq \alpha_i$. Let r_i be the element generating $H_2|_{\Omega_i} \cong C_2$. Then $r_i^{gc} = r_i$. So

$$(\alpha_i^{gc})^{r_i} = (\alpha_i^{gc})^{(r_i^{gc})} = \alpha_i^{r_i gc} = \alpha_i^{gc}.$$

This yields a contradiction since r_i only fixes one point α_i. Therefore $g = 1$.
Next we show that $N_i \geq \langle c_i\rangle$. Certainly $c_i \in N_{\mathrm{Sym}(\Omega_i)}(G_i)$. Let r_i be the element generating $H_2|_{\Omega_i}$. Since $r_i \in N_{\mathrm{Sym}(\Omega_i)}(G_i)$ and r_i fixes α_i, we have $r_i \in \langle c_i\rangle$. So $r_i^{c_i} \in H_2|_{\Omega_i}$ and hence c_i normalises $H_2|_{\Omega_i}$.
Lastly, the isomorphism follows from Lemma 4.3.8. □

We have seen that H_2 is a subdirect product of C_2^k, where for each $1 \leq i \leq k$, the projection $H_2|_{\Omega_i} \cong C_2$ is intransitive and has support $\Omega_i \backslash \alpha_i$. Recall Γ from Notation 5.5.5. Then $H_2|_\Gamma$ is in class $\mathfrak{InP}(C_2)$. So $N_{\mathrm{Sym}(\Gamma)}(H_2|_\Gamma)$ can be computed using Algorithm 6. Next, we show how $N_T(H_2)$ can be computed from $N_{\mathrm{Sym}(\Gamma)}(H_2|_\Gamma)$. Recall that we take natural inclusion from $\mathrm{Sym}(\Omega_i)$ to S_n.

Lemma 5.5.8. *Let $\theta : N_{\mathrm{Sym}(\Gamma)}(H_2|_\Gamma) \to K$ be defined by $\Omega_i^{\theta(g)} = \Omega_j$ if $\Omega_{i1}^g = \Omega_{j1}$. Then $N_T(H_2) = \langle N_1, N_2, \dots, N_k, \mathrm{Im}(\theta)\rangle$.*

Proof. $\geq$: Observe that as $|(H_2|_{\Omega_i})| = 2$, we have $N_{\mathrm{Sym}(\Omega_i)}(H_2|_{\Omega_i}) = C_{\mathrm{Sym}(\Omega_i)}(H_2|_{\Omega_i})$, so N_i centralises H_2. Therefore $N_i \leq N_T(H_2)$. To show that $\mathrm{Im}(\theta) \subseteq N_T(H_2)$, let $g \in N_{\mathrm{Sym}(\Gamma)}(H_2|_\Gamma)$ and let $h \in H_2$. Then there exists $h' \in H$ such that $(h|_\Gamma)^g = h'|_\Gamma$. We will show that $h^{\theta(g)} = h'$.
Let $1 \leq i \leq k$ and let $\Omega_{j1} = \Omega_{i1}^g$, so $\Omega_i^{\theta(g)} = \Omega_j$. If $h|_{\Omega_i} = 1$ then $(h^{\theta(g)})|_{\Omega_j} = (h|_{\Omega_i})^{\theta(g)} = 1$. As $h'|_{\Omega_j}$ is an element of $H_2|_{\Omega_j} \cong C_2$ with $h'|_{\Omega_{j1}} = (h|_{\Omega_{i1}})^g = 1$, it follows that $h'|_{\Omega_j} = 1$. So $(h^{\theta(g)})|_{\Omega_j} = h'|_{\Omega_j}$.
Suppose instead that $h|_{\Omega_i} \neq 1$, so $h'|_{\Omega_j}$ is the non-trivial element of $H_2|_{\Omega_j}$. Since $\theta(g) \in K$, we can get a bijection witnessing the permutation isomorphism from $H|_{\Omega_i}$ to $H|_{\Omega_j}$ using the restriction of $\theta(g)$ onto $\Omega_i \cup \Omega_j$. As in the proof of the forward implication of Proposition 1.1.7, it follows that $(H|_{\Omega_i})^{\theta(g)} = H|_{\Omega_j}$. Since K permutes the fixed points of H_2, we have $(H_2|_{\Omega_i})^{\theta(g)} = H_2|_{\Omega_j}$, so $(h|_{\Omega_i})^{\theta(g)}$ is also the non-trivial element of $H_2|_{\Omega_j}$. Hence $(h^{\theta(g)})|_{\Omega_j} = (h|_{\Omega_i})^{\theta(g)} = h'|_{\Omega_j}$.
Therefore $h^{\theta(g)} = h' \in H_2$ and hence $\theta(g) \in N_{S_n}(H_2)$. Since $\theta(g) \in K \leq T$, we have $\theta(g) \in N_T(H_2)$.
$\leq$: Let $t \in N_T(H_2)$ and let $\iota := \theta(t|_\Gamma) \in K \leq T$. Then as both $N_T(H_2)$ and $\mathrm{Im}(\theta)$ permute the sets Ω_i, it follows that $\Omega_i^t = \Omega_j$ if and only if $\Omega_{i1}^{t|_\Gamma} = \Omega_{j1}$ if and only if $\Omega_i^t = \Omega_j$. Therefore $t\iota^{-1}$ is an element of T which fixes each Ω_i setwise. Then by Part 2 of Lemma 5.5.7 , $T \cong \langle c_1 \rangle \wr S_k$, and so $t\iota^{-1} \in \langle c_1, c_2, \dots, c_k\rangle$, which is $\langle N_1, N_2, \dots, N_k\rangle$. □

Hence by letting $R := \langle N_{\mathrm{Sym}(\Omega_1)}(G_1), N_{\mathrm{Sym}(\Omega_2)}(G_2), \dots, N_{\mathrm{Sym}(\Omega_k)}(G_k), \mathrm{Im}(\theta)\rangle$, we have $N_T(H_2) \leq R$. As in Theorem 5.2.22, we may compute $N_R(H_p)$ by considering all $\kappa \in \mathrm{Im}(\theta) \leq K$. Lastly, we show that $N_L(H_p) \cap N_L(H_2)$ can be computed from $N_R(H_p)$.

Lemma 5.5.9. *Let $W \leq GL_k(p)$ be as in Definition 5.1.9. Let $\Xi : L \to W$ be as in Lemma 5.1.13. Let T be as in Lemma 5.5.7 and let $\Xi|_T : T \to W$ be the restriction of Ξ to $T \leq L$. Then $\Xi|_T$ is an isomorphism and $N_L(H_p) \cap N_L(H_2) = \Xi|_T^{-1}(\Xi(N_R(H_p)))$.*

Proof. It follows from Part 1 of Lemma 5.1.13 that $\Xi|_T$ is a homomorphism. Since $L = \langle T, G\rangle$ and Ξ has kernel G by Part 3 of Lemma 5.1.13, it follows from Part 2 of Lemma 5.5.7 that $T \cap G = 1$, so $\Xi|_T$ is an isomorphism.
$\leq$: By Part 1 of Lemma 5.5.7 and Lemma 5.5.8, we have

$$N_L(H_p) \cap N_L(H_2) = N_T(H_2) \cap N_T(H_p) \leq R \cap N_T(H_p) \leq R \cap N_L(H_p) = N_R(H_p),$$

since $T, R \leq L$. So $\Xi|_T(N_L(H_p) \cap N_L(H_2)) = \Xi(N_L(H_p) \cap N_L(H_2)) \leq \Xi(N_R(H_p))$.
$\geq$: Let $l \in N_R(H_p)$. Then $\Xi|_T^{-1}(\Xi(l))$ is an element of T normalising H_p. So it remains to show that $\Xi|_T^{-1}(\Xi(l))$ normalises H_2.
By Lemma 5.5.8, $R = \langle N_T(H_2), G\rangle$. As $\mathrm{Ker}(\Xi) = G$ by Part 3 of Lemma 5.1.13, we

have $\Xi|_T(N_T(H_2)) = \Xi(N_T(H_2)) = \Xi(R)$. So

$$\Xi|_T^{-1}(\Xi(l)) \in \Xi|_T^{-1}(\Xi(R)) = N_T(H_2).$$

$\square$

Therefore, to compute the normaliser $N_{S_n}(H)$ of $H \leq S_n$ in class $\mathfrak{InP}(D_{2p})$, we do as follows.

1. Compute the Sylow p-subgroup H_p of H and a Sylow 2-subgroup H_2 of H.
2. Let Γ be as in Notation 5.5.5. Compute $N_{\mathrm{Sym}(\Gamma)}(H_2|_\Gamma)$ using Algorithm 6.
3. Let θ be as in Lemma 5.5.8 and let

   $$R := \langle N_{\mathrm{Sym}(\Omega_1)}(H_p|_{\Omega_1}), N_{\mathrm{Sym}(\Omega_2)}(H_p|_{\Omega_2}), \ldots, N_{\mathrm{Sym}(\Omega_k)}(H_p|_{\Omega_k}), \mathrm{Im}(\theta)\rangle.$$

   Compute $N_R(H_p)$ using backtrack search (Algorithm 6).
4. Compute $N_L(H_p) \cap N_L(H_2)$ as in Lemma 5.5.9. Then by Proposition 5.5.6, $N_{S_n}(H) = \langle N_L(H_p) \cap N_L(H_2), H\rangle$.

Note that by Proposition 2.1.18, Step 1 can be computed in polynomial time. Note also that by Parts 8 and 9 of Theorem 2.1.9, Step 4 can be computed in polynomial time.

5.6 Results

5.6.1 Normalisers of $H \in \mathfrak{InP}(C_p)$

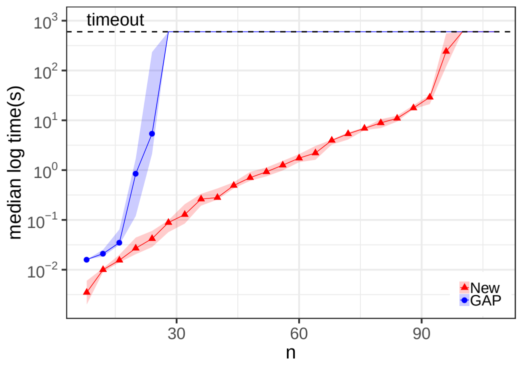
We compare the performance of computing $N_{S_n}(H)$ for $H \leq S_n$ in class $\mathfrak{InP}(C_p)$, using Algorithm 6 and using the GAP function Normalizer.

In our experiments we considered groups $H \leq S_{pk}$ in class $\mathfrak{InP}(C_p)$ as in Definition 5.1.2 that are isomorphic to $C_p^{k/2}$, for $p = 2, 3, 5, 7, 11$. We generate these instances by populating the entries of a $k/2 \times k$ matrix with random elements of $\mathbb{F}_p$. If the rank of M is not $k/2$, we rerun the generation.

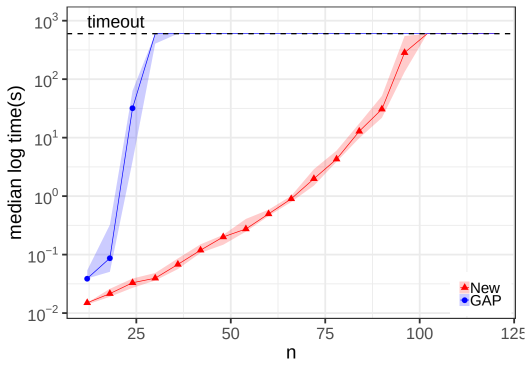
For each value of p and for each value of k, we create 10 instances of H and compute $N_{S_n}(H)$ using both the function in GAP[2] and our new algorithm. Each computation is run with a 10 minute time limit. We report the median, lower quartile and upper quartile time (in seconds) to compute $N_{S_n}(H)$ in Figure 5.1.

The results (Figure 5.1) show that the new algorithm performs faster than the one implemented in GAP. We can also compute the normaliser of groups with a much higher

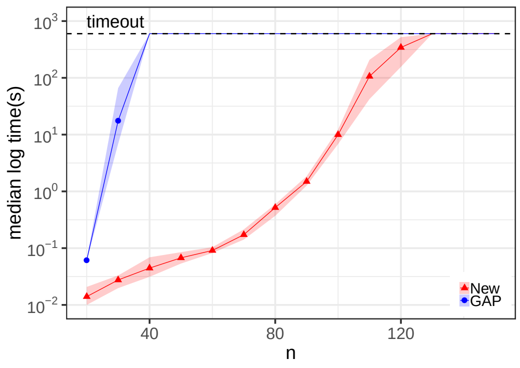
[2]There are some runtime gain by instead computing $N_L(H)$, where $L \cong (C_p \wr C_{p-1}) \wr S_k$ is as in Proposition 5.1.8, or even by computing $N_J(H)$ where $J \cong S_p \wr S_k$ is a subgroup of S_n which permutes the H-orbits. For simplicity, we compute $N_{S_n}(H)$ in our experiments.



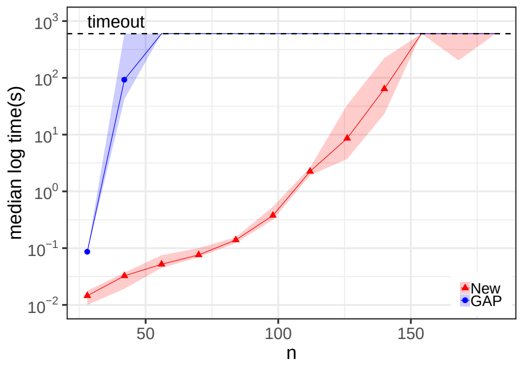
(a) $p = 2$



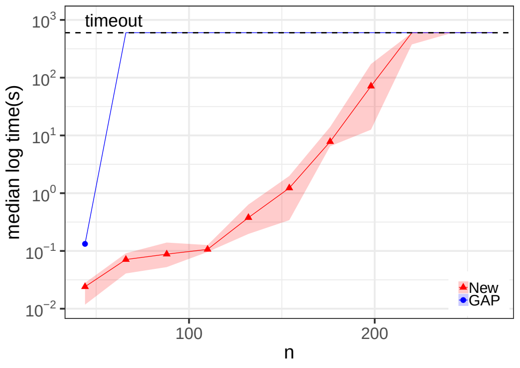
(b) $p = 3$



(c) $p = 5$



(d) $p = 7$



(e) $p = 11$

Figure 5.1: Median log time (s) for computing normalisers of 10 random subdirect product of C_p^k, generated by $k/2$ random generators with 10 minutes timeout. The lower and upper boundaries of the shaded area give the lower and upper quartiles respectively.

degree within the 10 minute limit. Many groups whose normalisers previously could not be computed in 10 minutes can now be computed in less than 0.1s.

Next we compare the performance of computing $N_{S_n}(H)$ for $H \leq S_n$ in class $\mathfrak{InP}(C_p)$ using the methods described in Proposition 5.2.3 and Theorem 5.2.22 respectively.

The results are shown in Figure 5.2, where FULLSEARCH refers to the algorithm described in Section 3.3. To obtain complexity $2^{O(\frac{n}{p}\log(\frac{n}{p}))}$, LIMITDEPTH is a combination of methods of Proposition 5.2.3 and Theorem 5.2.22. The algorithm is as follows. Let $H \in \mathfrak{InP}(C_p)$ and let $m = \dim \gamma(H)$. As in Algorithm 6, we perform backtrack search in K. At a node at depth m, we iterate over all $(p-1)^m$ combination of $(\mathbb{F}_p^*)^m$, as in Proposition 5.2.3. If we succeed in finding a normalising element $g \in N_{S_n}(H)$, we update N as $\langle N, g\rangle$, else we backtrack. Results (Figure 5.2) show that even though FULLSEARCH has a worse worst case complexity, it performs much better than LIMITDEPTH in practice, especially when p or m are large.

p	m	FULLSEARCH	LIMITDEPTH
5	4	0.11	0.125
5	6	0.4765	0.625
5	8	3.0625	2.953
5	10	8.2415	65.75
2	6	0.125	0.125
3	6	0.2655	0.297
7	6	0.9605	12.0235
11	6	5.453	»600

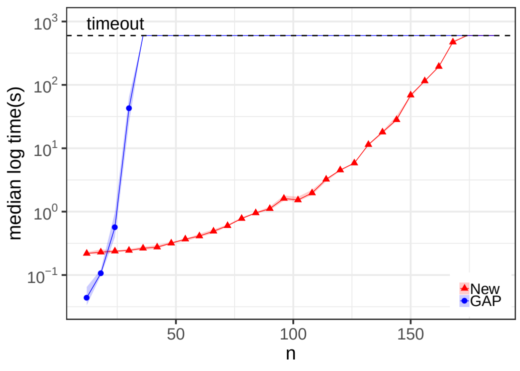
Figure 5.2: Median times (in seconds) of computing $N_{S_n}(H)$ of 10 random $H \leq S_{20p}$ in class $\mathfrak{InP}(C_p)$ with 20 orbits and $\dim \gamma(H) = m$, using LIMITDEPTH and FULLSEARCH.

5.6.2 Normalisers of $H \in \mathfrak{InP}(D_{2p})$

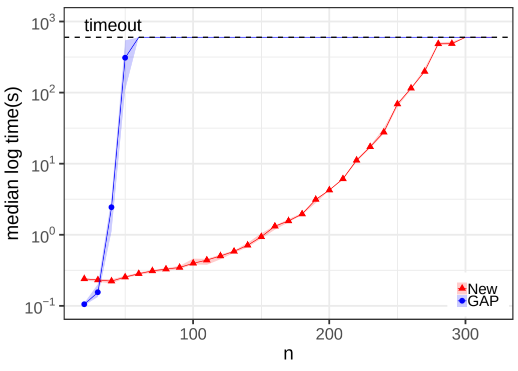
Lastly we compare the performance of computing $N_{S_n}(H)$ for $H \leq S_n$ in class $\mathfrak{InP}(D_{2p})$, using the methods described in Section 5.5 and using the GAP function NORMALIZER.

We consider $H \leq S_{pk}$ in class $\mathfrak{InP}(D_{2p})$ as in Notation 5.5.1, for $p = 3, 5, 7, 11$. By Part 2 of Lemma 5.5.3, H is a product of a subdirect product of C_2^k and a subdirect product of C_p^k. We generate H by generating these subdirect products that are isomorphic to $C_2^{k/2}$ and $C_p^{k/2}$ respectively, using the method described in Section 5.6.1. The rest of the experiment works the same as that in Section 5.6.1.

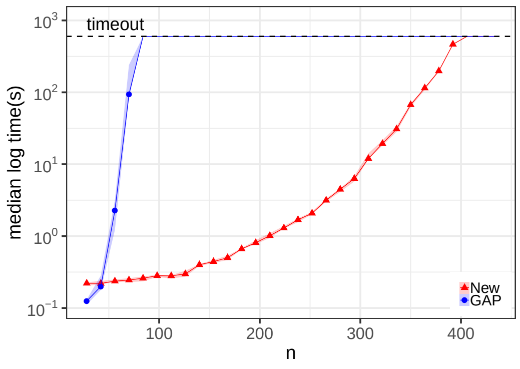
The results (Figure 5.3) show that the new algorithm generally performs better than the one implemented in GAP, only slightly losing in small cases. As in Section 5.6.2, we can now compute the normalisers of much bigger groups than we previously could. Many groups whose normalisers could not previously be computed in 10 minutes can now be computed in less than 0.1s. The algorithm is also very consistent, as the upper and lower quartiles are very close to the median times.



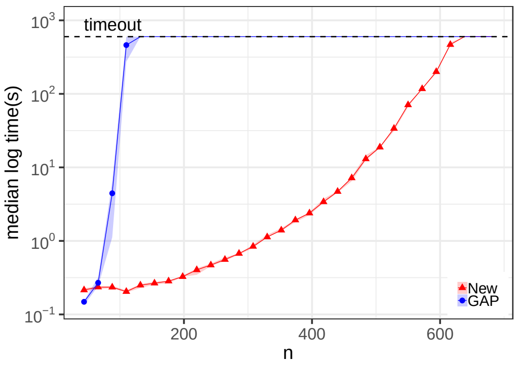
(a) $p = 3$



(b) $p = 5$



(c) $p = 7$



(d) $p = 11$

Figure 5.3: Median log time (s) for computing normalisers of 10 random subdirect product of D_{2p}^k, generated by $k/2$ random generators from C_2^k and $k/2$ random generators of C_p^k with 10 minutes timeout. The lower and upper boundaries of the shaded area give the lower and upper quartiles respectively.

Chapter 6

Normalisers of Groups In Class $\mathfrak{InP}(T)$ for T Non-abelian Simple

Let $T \leq S_m$ be a transitive non-abelian simple group. In this chapter, we consider $H \leq S_n$ for $H \in \mathfrak{InP}(T)$. The main result of this chapter is to prove that the normaliser $N_{S_n}(H)$ can be computed in polynomial time. Recall from Chapter 2 that we measure complexity in terms of n. So, the main theorem is:

Theorem 6.0.1. *Let $T \leq S_m$ be transitive and non-abelian simple. Let $H = \langle X \rangle \leq S_n$ be in the class $\mathfrak{InP}(T)$. Then $N_{S_n}(H)$ can be computed in time polynomial in $|X|n$.*

Since we are also interested in a practical algorithm, we will present an efficient polynomial time algorithm to compute $N_{S_n}(H)$ for $H \in \mathfrak{InP}(T)$. Additionally, we extend our result to give a fast algorithm to compute $N_{S_n}(H)$ for $H \in \mathfrak{InP}(S_m)$ where $5 \leq m \neq 6$.

The structure of the chapter is as follows. In Section 6.1, we present some polynomial time results we shall use in later sections. In Section 6.2, we analyse the structure of H. In Section 6.3, we prove Theorem 6.0.1. In Section 6.4, we further analyse the normaliser $N_{S_n}(H)$, which will give a better implemented algorithm for computing $N_{S_n}(H)$. In Section 6.5, we present an algorithm to compute $N_{S_n}(H)$ using results in Section 6.4. Let m be an integer such that $5 \leq m \neq 6$. In Section 6.6, we show that the normaliser $N_{S_n}(H)$ for $H \leq S_n$ in class $\mathfrak{InP}(S_m)$ can be computed using the normalisers of two subgroups in classes $\mathfrak{InP}(A_m)$ and $\mathfrak{InP}(C_2)$ respectively. In Section 6.7, we present the runtimes of our algorithms against the existing implementation in GAP.

6.1 Polynomial time library

In this section, we present some polynomial-time results we will be using later in the chapter.

We shall assume that all groups are given by their generating sets. By Theorem 2.1.1, we further assume that the generating set of a group $G \leq S_m$ has size at most m. By

Parts 4 to 6 of Theorem 2.1.9, we can check if a map defined on generators extends to a homomorphism, isomorphism or automorphism in polynomial time. Hence, for the rest of this chapter, all homomorphisms (including isomorphisms and automorphisms) will be given by the images of a generating set of the domain group under the homomorphism.

We start with polynomial time simplicity test of permutation groups. Since the abelian simple groups are regular cyclic groups and calculating the size of a permutation group can be done in polynomial time (Part 1 of Theorem 2.1.9), we can also detect non-abelian simple groups in polynomial time.

Lemma 6.1.1 ([BKL83, Theorem 5.1])**.** *Let $T = \langle X \rangle \leq S_m$. Then in polynomial time, we can decide if T is simple. Hence, in polynomial time, we can decide if T is non-abelian simple.*

Next, we give a polynomial bound on the size of the outer automorphism group of a non-abelian simple group. Guralnick et al. give a stronger result in [GMP17], but the following simplified version is sufficient for polynomial-time arguments.

Lemma 6.1.2 ([GMP17])**.** *Let T be a non-abelian simple group with a non-trivial permutation representation of degree m. Then $|Out(T)| \leq 2\sqrt{m}$.*

We will frequently assume that we have a library of standard copies of permutation representations of non-abelian simple groups. The library stores one permutation representation for each simple group S.

Remark 6.1.3. For each group S in a library of standard copies of permutation representations of non-abelian simple groups, we assume that the following information is known:

1. a pair of elements $g, h \in S$ such that $S = \langle g, h \rangle$,
2. the size $|S|$,
3. a right transversal R of $Inn(S)$ in $\mathrm{Aut}(S)$, where each $\omega \in R$ is given by the images of g and h under ω.

The following paraphrased theorem gives polynomial-time isomorphisms between sufficiently large simple groups.

Lemma 6.1.4 ([Kan85, Theorem 9.1])**.** *There exists a polynomial time algorithm which, given a simple subgroup $T \leq S_m$ such that $|T| > m^8$, computes a 'natural' representation of T. Hence, assuming that we have a library of standard non-abelian simple groups as described in Remark 6.1.3, we can construct an isomorphism between T and a standard copy S in time polynomial in m.*

For smaller simple groups T, we can construct an isomorphism between T and its standard copy via the following lemma.

Lemma 6.1.5. *Let $T \leq S_m$ be a non-abelian simple group such that $|T| \leq m^8$. Assume that we have a library of standard copies of non-abelian simple groups, as described in Remark 6.1.3. Then we can construct an isomorphism between T and a standard copy S in time polynomial in m.*

Proof. We first compute the order of T, which can be done in polynomial time (see Part 1 of Theorem 2.1.9). By [Art55], all finite simple groups have distinct orders apart from:

(a) $A_8 \cong PSL_4(2)$ and $PSL_3(4)$, which have order 20160, and

(b) $P\Omega_{2n+1}(q)$ and $PSp_{2n}(q)$ for each odd prime power q and all $n \geq 3$.

So we can identify at most 2 candidates for S. It is well known that all simple groups are 2-generated. For each candidate S, let s and t be the generators of the standard copy of S.
For each pair $g, h \in T$, let $\phi_{g,h} : T \to S$ be a map defined by $g \mapsto s$ and $h \mapsto t$. Then $T \cong S$ if and only if there exists $g, h \in T$ such that $\phi_{g,h}$ is an isomorphism. That is, there exists $g, h \in T$ such that $\langle g, h \rangle = T$ and the map $\phi_{g,h}$ extends to a homomorphism. We shall consider all pairs $g, h \in T$. We check the first assertion by checking if $|\langle g, h \rangle| = |T|$, which can be done in polynomial time, by Part 1 of Theorem 2.1.9. Since $\langle g, h \rangle \leq T$, this implies that $\langle g, h \rangle = T$. By Part 4 of Theorem 2.1.9, deciding if $\phi_{g,h}$ extends to a homomorphism can be decided in polynomial time.
Since we consider at most $2|T|^2$ maps and $|T| \leq m^8$, the algorithm runs in polynomial time. □

Hence, we can obtain an isomorphism between a given non-abelian simple group T and a standard copy S in polynomial time. Therefore, given two non-abelian simple groups T_1 and T_2, we can decide if $T_1 \cong T_2$ in polynomial time by checking if their respective standard copies S_1 and S_2 are the same. Furthermore, if T_1 and T_2 are isomorphic non-abelian simple groups, we can obtain an isomorphism $\phi : T_1 \to T_2$ in polynomial time by first obtaining isomorphisms $\phi_1 : T_1 \to S_1$ and $\phi_2 : T_2 \to S_2$, and then $\phi = \phi_1 \phi_2^{-1}$.

Corollary 6.1.6. *Let $T_1 = \langle X \rangle$ and $T_2 = \langle Y \rangle$ be non-abelian simple groups in S_n. Then in polynomial time, we can decide if $T_1 \cong T_2$, and if they are, give an isomorphism between them.*

As discussed in Remark 6.1.3, we assume that the outer automorphisms $Out(S)$ of each group S in the library of standard non-abelian simple groups can be listed as a set of coset representatives. For $m \geq 5$ and $m \neq 6$, we have that $Out(A_m) = C_2$, and $Out(A_6) = V_4$, the Klein four group. For the outer automorphisms of simple groups of Lie type, see [Car72, Chapter 12]. For the outer automorphisms of the sporadic simple groups, see [Lyo11].

Lemma 6.1.7. *Let $T = \langle X \rangle \leq S_m$ be given. Suppose that we have constructed an isomorphism between T and a standard copy S. Then a transversal R of $Inn(T)$ in $\mathrm{Aut}(T)$ can be obtained in polynomial time. Since $R \subseteq \mathrm{Aut}(T)$, each element $\alpha \in R$ of the transversal is given by the images of the generating set X of T under α.*

Recall from Lemma 2.1.19 that we can decide if a given isomorphism is induced by conjugation in polynomial time. This is an important lemma for this chapter and will be used repeatedly in different contexts throughout the chapter.

6.2 Structure of subdirect products of T^k

Let $T \leq S_m$ be a transitive non-abelian simple group. In this section, we analyse the structure of $H \in \mathfrak{InP}(T)$. In particular, we show in Proposition 6.2.4 that H is a disjoint direct product of subgroups isomorphic to T.

We start by fixing some notation.

Notation 6.2.1. Let $\Omega_1, \Omega_2, \ldots, \Omega_k$ be the orbits of H. Let $\Omega = \dot\cup_{i=1}^{k} \Omega_i = \{1, 2, \ldots, n\}$, where $n = mk$.

Recall the notation from Corollary 1.4.4. Since each $H|_{\Omega_i} \cong T$ is simple, the N_i must be either 1 or $H|_{\Omega_i}$. We now prove a result we shall use in the proof of Proposition 6.2.4. Recall the definition of projection maps in Notation 1.2.4, and recall that we denote $\{1, 2, \ldots, i\}$ by $\bar{i}$.

Lemma 6.2.2. *Let $\Gamma_1, \Gamma_2, \ldots, \Gamma_r$ be pairwise disjoint subsets of Ω such that $\Omega = \cup_{i=1}^{r} \Gamma_i$. For each $1 \leq i \leq r$, let T_i be a subgroup of $\mathrm{Sym}(\Omega)$ with support Γ_i such that $T_i \cong T$. Let $A \leq \mathrm{Sym}(\Omega)$ be the group $A = T_1 T_2 \ldots T_r$. Let N be a normal subgroup of A with index $|T|$. Then there exists a unique $1 \leq i \leq k$ such that N is the kernel of the projection of A onto Γ_i, and so $N = T_1 T_2 \ldots T_{i-1} T_{i+1} T_{i+2} \ldots T_r$.*

Proof. Let $1 \leq j \leq k$. Then $T_j \cong T$ is a minimal normal subgroup of A. By Lemma 1.3.2, either $T_j \leq N$ or $\langle T_j, N \rangle = T_j \times N$. If the latter is true then $\langle T_j, N \rangle$ is a subgroup of A with size $|T|^r$ and so is the group A. Since T_j and N have trivial intersection, they commute. So $N = T_1 T_2 \ldots T_{j-1} T_{j+1} T_{j+2} \ldots T_r$. We show that there exists a unique $1 \leq i \leq k$ such that $\langle T_i, N \rangle = A$.
For the proof of existence, aiming for a contradiction, suppose that there does not exist $1 \leq i \leq k$ such that $\langle T_i, N \rangle = A$. Then for all $1 \leq j \leq k$, we have $T_j \leq N$. Then since $A = T_1 T_2 \ldots T_r$, we have $A \leq N$, which is a contradiction.
For the proof of uniqueness, suppose that that there exist distinct i and j such that $\langle T_i, N \rangle = A$ and $\langle T_j, N \rangle = A$. By the observation in the opening paragraph, $N = T_1 T_2 \ldots T_{i-1} T_{i+1} T_{i+2} \ldots T_r$ and so $T_j \leq N$. Then $\langle T_j, N \rangle = N \neq A$, which is a contradiction. ☐

Next, we shall prove the main result of this section. To avoid confusing notation, we will denote the direct product of groups by $\bigtimes$. For disjoint sets Γ_1 and Γ_2, we shall consider subgroups of $\mathrm{Sym}(\Gamma_1) \times \mathrm{Sym}(\Gamma_2)$ as subgroups of $\mathrm{Sym}(\Gamma_1 \cup \Gamma_2)$.

Notation 6.2.3. Let $\Delta \subset \Omega$ be the union of some H-orbits. We denote by $H|_\Delta \times 1_{\Omega\setminus\Delta}$ the subgroup A of $\mathrm{Sym}(\Omega)$ with support Δ such that $A|_\Delta = H|_\Delta$.

Proposition 6.2.4. *Let $T \leq S_m$ be a transitive non-abelian simple group and let $H \in \mathfrak{InP}(T)$. Then there exists a partition $\mathcal{P} := \langle C_1 \mid C_2 \mid \dots \mid C_r\rangle$ of $\{1, 2, \dots, k\}$ such that by letting $\Gamma_i = \cup_{j\in C_i}\Omega_j$ for all $1 \leq i \leq r$, we have*

1. *$H|_{\Gamma_i} \cong T$ for all $1 \leq i \leq r$, and*
2. *$H = H|_{\Gamma_1} \times H|_{\Gamma_2} \times \dots \times H|_{\Gamma_r}$, where the action of the direct product is on the disjoint union of the supports of the direct factors, and so $H \cong T^r$.*

Proof. We proceed by induction on k. If $k = 1$ then $\mathcal{P} = \langle 1\rangle$ and $H = H|_{\Gamma_1} \cong T \cong T^k$. Let Δ be the set $\cup_{i=1}^{k-1}\Omega_i$. For the inductive step, let $\mathcal{P}_{k-1} := \langle C_1 \mid C_2 \mid \dots \mid C_r\rangle$ be the partition of $\{1, 2, \dots, k-1\}$ and let $\Gamma_i = \cup_{j\in C_i}\Omega_j$ for all $1 \leq i \leq r$ be such that

$$H|_{\Gamma_i} \cong T \text{ for all } 1 \leq i \leq r \text{ and } H|_\Delta = H|_{\Gamma_1} \times H|_{\Gamma_2} \times \dots \times H|_{\Gamma_r},$$

where we identify the direct product as the corresponding subgroup of $\mathrm{Sym}(\Delta)$. Observe that for each $1 \leq i \leq r$, the group $H|_{\Gamma_i} \times 1_{\Delta\setminus\Gamma_i}$ is a subgroup of $H|_\Delta$. Using the notation of Corollary 1.4.4, since $N_k \trianglelefteq H|_{\Omega_k} \cong T$, either $N_k = 1$ or $N_k = H|_{\Omega_k}$.
Suppose first that $N_k = H|_{\Omega_k}$. Then by Part 1 of Corollary 1.4.6, $H = H|_\Delta \times N_k$. Then by letting $C_{r+1} = \{k\}$ and $\mathcal{P} := \langle C_1 \mid C_2 \mid \dots \mid C_{r+1}\rangle$, conditions (1) and (2) are satisfied.
Suppose now that $N_k = 1$ and take $R_k = H|_{\Omega_k}$. Then by the first isomorphism theorem,

$$|\mathrm{Ker}(\theta_{k-1})| = |(H|_\Delta)|/|\mathrm{Im}(\theta_{k-1})| = |T|^{r-1}.$$

So $\mathrm{Ker}(\theta_{k-1})$ is a normal subgroup of $H|_\Delta \cong T^r$ of index $|T|$. By Lemma 6.2.2, there exists a unique $1 \leq s \leq r$ such that

$$\mathrm{Ker}(\theta_{k-1}) = \bigtimes_{1\leq j\leq r,\, j\neq s} H|_{\Gamma_j} \times 1_{\Gamma_s} = H|_{\Delta\setminus\Gamma_s} \times 1_{\Gamma_s},$$

and so $\theta_{k-1}\left(H|_{\Delta\setminus\Gamma_s} \times 1_{\Gamma_s}\right) = 1$. Then by the definition of φ_{k-1}, we have

$$\varphi_{k-1}\left(H|_{\Delta\setminus\Gamma_s} \times 1_{\Gamma_s}\right) = H|_{\Delta\setminus\Gamma_s} \times 1_{\Gamma_s\cup\Omega_k}.$$

Let $S := 1_{\Delta\setminus\Gamma_s} \times H|_{\Gamma_s}$. Since $\theta_{k-1}(H|_\Delta) \neq 1$ and $H|_\Delta = H|_{\Delta\setminus\Gamma_s} \times H|_{\Gamma_s}$, we have

$\theta_{k-1}(S) \neq 1$. By the definition of φ_{k-1}, we have

$$\begin{aligned}\varphi_{k-1}(S) &= \{h|_\Delta \theta_i(h|_\Delta) \mid h|_\Delta \in S\} \\ &= \{(h|_\Delta h|_{\Omega_k}) \mid h|_\Delta \in S\} \\ &= H|_{\Gamma_s \cup \Omega_k} \times 1_{\Delta \setminus \Gamma_s}.\end{aligned}$$

So, by Corollary 1.4.4, since $N_k = 1$, we have

$$\begin{aligned}H &= \varphi_{k-1}(H|_\Delta) = \varphi_{k-1}\left(\left(H|_{\Delta\setminus\Gamma_s} \times 1_{\Gamma_s}\right)(S)\right) \\ &= \langle \varphi_{k-1}\left(H|_{\Delta\setminus\Gamma_s} \times 1_{\Gamma_s}\right), \varphi_{k-1}(S)\rangle.\end{aligned}$$

Since $H|_{\Delta\setminus\Gamma_s} \times 1_{\Gamma_s}$ and S commute, their images under φ_{k-1} also commute, so

$$H = (H|_{\Delta\setminus\Gamma_s} \times 1_{\Gamma_s\cup\Omega_k})(1_{\Delta\setminus\Gamma_s} \times H|_{\Gamma_s\cup\Omega_k}) = \left(\bigtimes_{1\leq j\leq r,\, j\neq s} H|_{\Gamma_j}\right) \times H|_{\Gamma_s\cup\Omega_k},$$

where we identify the direct product as a subgroup of $\mathrm{Sym}(\Omega)$. Let $\mathcal{P}$ be the partition of $\{1, 2, \ldots, k\}$ consisting of cell $C_s \cup \{k\}$ and all other cells C_j for all $1 \leq j \leq r$ such that $j \neq s$. Then Condition (2) is satisfied. Clearly, $H|_{\Gamma_j} \cong T$ for all $1 \leq j \leq r$ such that $j \neq s$. By Corollary 1.4.4, the restriction of θ_{k-1} to S is a homomorphism. Since $S \cong T \cong \theta_{k-1}(S)$, it is an isomorphism. So, the restriction of φ_{k-1} to S is also an isomorphism. Then $H|_{\Gamma_s\cup\Omega_k} = \varphi_{k-1}(S) \cong T$, and so condition (1) is also satisfied. □

Before we prove the polynomial time result in Lemma 6.2.8, we highlight a few corollaries of Proposition 6.2.4. We start by showing that each $H|_{\Gamma_i}$ is a diagonal subgroup of $T^{|C_i|}$.

Lemma 6.2.5. *Suppose that $H \cong T$. Then for each $2 \leq i \leq k$, there exists an isomorphism $\psi_i : H|_{\Omega_1} \to H|_{\Omega_i}$ such that $h|_{\Omega_i} = \psi_i(h|_{\Omega_1})$ for all $h \in H$.*
Therefore, for all $h \in H$, by considering the $\mathrm{Sym}(\Omega_i)$ as subgroups of $\mathrm{Sym}(\Omega)$ via the natural inclusion maps, we have $h = h|_{\Omega_1} \prod_{i=2}^{k} \psi_i(h|_{\Omega_1})$.

Proof. Let $1 \leq i \leq k$. Then since $H \cong T \cong H|_{\Omega_i}$, the projection map $\pi_i : H \to H|_{\Omega_i}$ defined by $h \mapsto h|_{\Omega_i}$ is an isomorphism. Let $\psi_i : H|_{\Omega_1} \to H|_{\Omega_i}$ be defined by $\psi_i(h|_{\Omega_1}) = \pi_i(\pi_1^{-1}(h|_{\Omega_1}))$ for all $h \in H$. Then $\psi_i : H|_{\Omega_1} \to H|_{\Omega_i}$ is an isomorphism and $\psi_i(h|_{\Omega_1}) = \pi_i(h) = h|_{\Omega_i}$. □

Notation 6.2.6. Let T be as in Notation 6.4.1. Denote by $\overline{N_{S_m}(T)}$ the subgroup of $\mathrm{Aut}(T)$ induced by $N_{S_m}(T)$.

Recall $\equiv_o$ from Definition 2.1.10. The order of $\overline{N_{S_m}(T)}$ can be used to bound the number of equivalence classes of $\equiv_o$.

Lemma 6.2.7. *Let $T \leq S_m$ be a transitive and non-abelian simple group. Suppose that $|\mathrm{Aut}(T)|/|\overline{N_{S_m}(T)}| = t$. Let $U \in \mathfrak{InP}(T)$ such that $U \cong T$. Then the U-orbits are partitioned into at most t equivalence classes by $\equiv_o$.*

Proof. For this proof, we will be writing maps on the right. Let $\Omega_1, \Omega_2, \ldots, \Omega_k$ be the orbits of U and let $\Gamma = Supp(T) = \{1, 2, \ldots, m\}$. We identify T and $U|_{\Omega_i}$ for all $1 \leq i \leq k$ as a subgroups $\mathrm{Sym}(\Gamma \cup \Omega_i)$. By Proposition 1.1.7, for all $1 \leq i \leq k$, there exists $d_i \in \mathrm{Sym}(\Gamma \cup \Omega_i)$ such that $T^{d_i} = U|_{\Omega_i}$. By Lemma 6.2.5, there exist $\psi_i : U|_{\Omega_1} \to U|_{\Omega_k}$ for all $2 \leq i \leq k$ such that for all $u \in U$, we have $u|_{\Omega_i} = (u|_{\Omega_1})\psi_i$. Then for all $1 \leq i \leq k$, the map β_i defined by $\tau \mapsto ((\tau^{d_i})\psi_i^{-1})^{d_1^{-1}}$ is an automorphism of T. Let $1 \leq i, j \leq k$. We show that if $\beta_i \overline{N_{S_m}(T)} = \beta_j \overline{N_{S_m}(T)}$, then $\Omega_i \equiv_o \Omega_j$, from which the result shall follow.
We have $\beta_i \beta_j^{-1} \in \overline{N_{S_m}(T)}$ and so $\tau \mapsto \tau^{\beta_i \beta_j^{-1}} = ((\tau^{d_i})\psi_i^{-1}\psi_j)^{d_j^{-1}}$ is induced by conjugation in S_m. Then $\psi_i^{-1}\psi_j$ is an isomorphism induced by conjugation in $\mathrm{Sym}(\Omega_i \cup \Omega_j)$ such that $(u|_{\Omega_i})\psi_i^{-1}\psi_j = u|_{\Omega_j}$ for all $u \in U$. By Lemma 2.1.12, $\Omega_i \equiv_o \Omega_j$. □

Lastly, we show that the decomposition in Proposition 6.2.4 can be computed in polynomial time.

Lemma 6.2.8. *Let $H = \langle X \rangle \leq S_n$. Assume that H is known to be in the class $\mathfrak{InP}(T)$ for some non-abelian simple group T. Then in polynomial time, we can compute $\mathcal{P}$ satisfying the conditions in Proposition 6.2.4.*

Proof. In this proof, we will identify the direct product of groups with disjoint supports to be a subgroup in the symmetric group over the disjoint unions of the supports of the direct factors. Let $\mathcal{P} = \langle C_1 \mid C_2 \mid \ldots \mid C_r \rangle$. As in Proposition 6.2.4, for all $1 \leq i \leq r$, let $\Gamma_i = \cup_{j \in C_i} \Omega_j$. We shall show that $H = H|_{\Gamma_1} \times H|_{\Gamma_2} \times \ldots \times H|_{\Gamma_r}$ is the finest disjoint direct product decomposition of H, from which the result shall follow from Theorem 3.0.2.
Since the $H|_{\Gamma_i}$ have disjoint supports, $H = H|_{\Gamma_1} \times H|_{\Gamma_2} \times \ldots \times H|_{\Gamma_r}$ is a disjoint direct product of H. To show that it is the finest decomposition, aiming for a contradiction, suppose that there exists $1 \leq i \leq r$ such that $H|_{\Gamma_i}$ is d.d.p. decomposable. Then there exists a non-trivial subset $\Delta \subset \Gamma_i$ such that $H|_{\Gamma_i} = H|_\Delta \times H|_{\Gamma_i \setminus \Delta}$. Then $H|_\Delta \times 1$ is a non-trivial normal subgroup of $H|_{\Gamma_i}$. Since $H|_{\Gamma_i} \cong T$ is simple, this is a contradiction. □

6.3 Normalisers in polynomial time

The main objective of this section is to show that the normaliser $N_{S_n}(H)$ of $H \leq S_n$ for $H \in \mathfrak{InP}(T)$ can be computed in polynomial time. We shall also show in Proposition 6.3.6 that in polynomial time, we can decide if a given permutation group is in the class $\mathfrak{InP}(T)$.

Let $U \leq S_t$ be non-abelian simple, that is not necessarily transitive. We start by showing that $N_{S_t}(U)$ can be computed in time polynomial in t [LM11]. Since the proof of the result is short, we will include it here.

Proposition 6.3.1 ([LM11])**.** *Let $U = \langle X \rangle \leq S_t$ be a non-abelian simple group. As in Lemma 6.1.7, assume that a transversal R of $Inn(U)$ in $\mathrm{Aut}(U)$ is known, where each element of R is defined by the images of X. Then $N_{S_t}(U)$ can be computed in polynomial time in t.*

Proof. Initialise Y as the empty set. For each $\omega \in R$, by Lemma 2.1.19, in polynomial time, we can decide if there exists $x_\omega \in S_t$ such that $u^\omega = x_\omega^{-1} u x_\omega$ for all $u \in U$, and exhibit one such x_ω if it exists. If such a x_ω exists, put in Y one such x_ω. This procedure is in polynomial time since by Lemma 6.1.2, $|R|$ is polynomially bounded.
Then by Lemma 1.3.9, $N_{S_t}(U) = \langle C_{S_t}(U), U, Y \rangle$. By Theorem 2.1.16, $C_{S_t}(U)$ can be computed in polynomial time, hence the result follows. □

Indeed, for $A \leq S_t$ with a polynomial bound on $|Out(A)|$, assuming that we have a transversal of $Inn(A)$ in $\mathrm{Aut}(A)$, the normaliser $N_{S_t}(A)$ can be computed in polynomial time.

Now, let Λ_1 and Λ_2 be disjoint sets. Let U_1 and U_2 be subgroups of $\mathrm{Sym}(\Lambda_1 \dot\cup \Lambda_2)$ isomorphic to a non-abelian simple group T. Let Λ_1 and Λ_2 be supports of U_1 and U_2 respectively. We shall show that, in polynomial time, we can decide if U_1 and U_2 are conjugate in $\mathrm{Sym}(\Lambda_1 \cup \Lambda_2)$. Recall from Corollary 6.1.6 that we can construct an isomorphism $\phi : U_1 \to U_2$ in polynomial time. We shall show that we can decide if U_1 and U_2 are conjugate in $\mathrm{Sym}(\Lambda_1 \cup \Lambda_2)$ by considering polynomialy many isomorphisms between U_1 and U_2.

Notation 6.3.2. Let A and B be groups and let $\phi : A \to B$ be an isomorphism. For $\alpha \in \mathrm{Aut}(A)$, let $\phi_\alpha : A \to B$ be the isomorphism defined by $\phi_\alpha(a) = \phi(a^\alpha)$ for all $a \in A$.

Lemma 6.3.3. *Let Λ_1 and Λ_2 be disjoint sets. Let $A = \langle X \rangle$ and $B = \langle Y \rangle$ be subgroups of $\mathrm{Sym}(\Lambda_1 \cup \Lambda_2)$ with supports Λ_1 and Λ_2 respectively. Let $\phi : A \to B$ be an isomorphism. Let R be a transversal of $Inn(A)$ in $\mathrm{Aut}(A)$, where each element of R is defined by the images of X. If A and B are conjugate in $\mathrm{Sym}(\Lambda_1 \cup \Lambda_2)$, then there exists $\omega \in R$ such that ϕ_ω is induced by conjugation in $\mathrm{Sym}(\Lambda_1 \cup \Lambda_2)$.*

Proof. Assume that A and B are conjugate in $\mathrm{Sym}(\Lambda_1 \cup \Lambda_2)$ and let $c \in \mathrm{Sym}(\Lambda_1 \cup \Lambda_2)$ be such that $A^c = B$. Let $\alpha \in \mathrm{Aut}(A)$ be defined by $g^\alpha = \phi^{-1}(g^c)$ for all $g \in A$. So, there exist $\iota \in Inn(A)$ and $\omega \in R$ such that $\alpha = \omega\iota$. We shall show that ϕ_ω is induced by conjugation in $\mathrm{Sym}(\Lambda_1 \cup \Lambda_2)$.
Let $g \in A$. Then $\phi_\omega(g) = \phi(g^\omega) = \phi(g^{\alpha\iota^{-1}})$. Since $\iota^{-1} \in Inn(A)$, there exists $g' \in A$ such that $(g^\alpha)^{\iota^{-1}} = (g^\alpha)^{g'}$. Then

$$\phi_\omega(g) = \phi(g^{\alpha\iota^{-1}}) = \phi((g^\alpha)^{g'}) = \phi(g^\alpha)^{\phi(g')} = \phi(\phi^{-1}(g^c))^{\phi(g')} = g^{c\phi(g')}.$$

So ϕ_ω is induced by the conjugation of $c\phi(g') \in \mathrm{Sym}(\Lambda_1 \cup \Lambda_2)$. □

Proposition 6.3.4. *Let Λ_1 and Λ_2 be disjoint sets. Let $U_1 = \langle X \rangle$ and $U_2 = \langle Y \rangle$ be subgroups of $\mathrm{Sym}(\Lambda_1 \cup \Lambda_2)$ with supports Λ_1 and Λ_2 respectively. Let T be a non-abelian simple group. Assume that we have an isomorphism $\psi_1 : U_1 \to T$, defined by the images of each $x \in X$. Assume further that we have a transversal R of $Inn(U_1)$ in $Aut(U_1)$, where each $\omega \in R$ is defined by the images of each $x \in X$. Then in time polynomial in $|\Lambda_1 \cup \Lambda_2|$, we can decide if there exists $c \in \mathrm{Sym}(\Lambda_1 \cup \Lambda_2)$ such that $U_1^c = U_2$, and if such a c exists, output c.*

Proof. By Corollary 6.1.6, in polynomial time, we can construct an isomorphism $\phi : U_1 \to U_2$. For each $\omega \in R$, using Lemma 2.1.19, we decide (and exhibit, if possible) if there exists $c \in \mathrm{Sym}(\Lambda_1 \cup \Lambda_2)$ such that ϕ_ω is induced by the conjugation of c. If none of the ϕ_ω is induced by a conjugation, then by Lemma 6.3.3, U_1 and U_2 are not conjugate in $\mathrm{Sym}(\Lambda_1 \cup \Lambda_2)$. Otherwise, we find $c \in \mathrm{Sym}(\Lambda_1 \cup \Lambda_2)$ such that $U_1^c = \phi_\omega(U_1) = U_2$. Then the polynomial time result follows from Lemmas 2.1.19 and 6.1.2. □

Finally, we shall show that given $H = \langle X \rangle \leq S_n$, in polynomial time, we can

1. decide if $H \in \mathfrak{InP}(T)$ for some transitive non-abelian simple group T, and
2. if it is, compute $N_{S_n}(H)$.

We shall start with the first assertion. A useful corollary of Proposition 6.3.4 is that we can decide if two non-abelian simple groups are permutation isomorphic, and hence deciding if $H \in \mathfrak{InP}(T)$ can be done in polynomial time.

Lemma 6.3.5. *Let $A = \langle X \rangle \leq S_m$ and $B = \langle Y \rangle \leq S_m$ be non-abelian simple. Assume that we have a library of standard copies of non-abelian simple groups, as described in Remark 6.1.3. Then in polynomial time, we can decide if A is permutation isomorphic to B, and if they are, output $c \in S_m$ such that $A^c = B$.*

Proof. By Corollary 6.1.6, in polynomial time, we can construct isomorphisms ϕ_1 and ϕ_2 from A and B respectively to their respective standard copy S and S'. If $S = S'$, then $A \cong B$, and $\phi_1\phi_2^{-1}$ is an isomorphism between A and B. By Proposition 1.1.7, A is permutation isomorphic to B if and only if A and B are conjugate in S_m. Then the result follows from Proposition 6.3.4. □

Proposition 6.3.6. *Let $H = \langle X \rangle \leq S_n$. Then in polynomial time, we can decide if there exists a transitive non-abelian simple group $S \leq \mathrm{Sym}(\Omega_1)$ such that $H \in \mathfrak{InP}(S)$, and if so, output S.*

Proof. First, we compute the orbits $\Omega_1, \Omega_2, \ldots, \Omega_k$ of H, which can be done in polynomial time by Proposition 2.1.2. Then there exists a transitive non-abelian simple group S such that $H \in \mathfrak{InP}(S)$ if and only if $H|_{\Omega_1}$ is non-abelian simple and for all $1 \leq i \leq k$, we have $H|_{\Omega_i}$ is permutation isomorphic to $H|_{\Omega_1}$.

The $H|_{\Omega_i}$ can be computed in polynomial time. By Lemma 6.1.1, we can check if $H|_{\Omega_1}$ is non-abelian simple in polynomial time. Let $2 \leq i \leq k$. Then by embedding $H|_{\Omega_1}$ and $H|_{\Omega_i}$ in $\mathrm{Sym}(\Omega_1 \cup \Omega_i)$ in the most natural way and Lemma 6.3.5, we can decide if $H|_{\Omega_1}$ is permutation isomorphic to $H|_{\Omega_i}$ in polynomial time. Since we consider at most $k - 1 \leq n$ pairs of groups, the result follows. □

Now, we return to $H \in \mathfrak{InP}(T)$ for a transitive non-abelian simple group $T \leq S_m$ and prove the main theorem of this chapter.

Theorem 6.3.7. *Let $H = \langle X \rangle \leq S_n$ be such that $H \in \mathfrak{InP}(T)$ for some transitive non-abelian simple group T. Then $N_{S_n}(H)$ can be computed in polynomial time.*

Proof. By Proposition 6.3.6, in polynomial time, we can find the transitive non-abelian simple group T such that $H \in \mathfrak{InP}(T)$.
To compute $N_{S_n}(H)$, we first compute a partition $\mathcal{P} = \langle C_1 \mid C_2 \mid \ldots \mid C_r \rangle$ as in Proposition 6.2.4. By Lemma 6.2.8, this partition can be computed in polynomial time. For each $1 \leq i \leq r$, let $\Gamma_i = \cup_{j \in C_i} \Omega_j$. So now we have $H = H|_{\Gamma_1} \times H|_{\Gamma_2} \times \ldots \times H|_{\Gamma_r}$, where we identify the direct product with the corresponding subgroup in the symmetric group over the disjoint union of the supports of the direct factors. By Proposition 4.2.6, $N_{S_n}(H)$ can be computed by computing the $N_{\mathrm{Sym}(\Gamma_i)}(H|_{\Gamma_i})$ for all $1 \leq i \leq r$ and the $c_{ij} \in \mathrm{Sym}(\Gamma_i \cup \Gamma_j)$ such that $(H|_{\Gamma_i})^{c_{ij}} = H|_{\Gamma_j}$ for all $1 \leq i, j \leq k$ such that $i \neq j$, if such c_{ij} exists.
Let $1 \leq i \leq r$. By Proposition 6.3.1, since $H|_{\Gamma_i}$ is non-abelian simple, $N_{\mathrm{Sym}(\Gamma_i)}(H|_{\Gamma_i})$ can be computed time polynomial in $|\Gamma_i|$. Since $|\Gamma_i| \leq n$ and $r \leq k < n$, the normalisers $N_{\mathrm{Sym}(\Gamma_i)}(H|_{\Gamma_i})$ for all $1 \leq i \leq r$ can be computed in polynomial time.
Let $1 \leq i, j \leq r$ such that $i \neq j$. By Proposition 6.3.4, in time polynomial in $|\Gamma_i \cup \Gamma_j|$, we can decide if there exists $c_{ij} \in \mathrm{Sym}(\Gamma_i \cup \Gamma_j)$ such that $(H|_{\Gamma_i})^{c_{ij}} = H|_{\Gamma_j}$, and output c_{ij} if it exists. Since $|\Gamma_i \cup \Gamma_j| \leq n$ and we consider r^2 pairs of orbits, all such c_{ij} can be computed in polynomial time. □

Hence, complexity-wise, the normaliser $N_{S_n}(H)$ of $H \leq S_n$ where $H \in \mathfrak{InP}(T)$ for some transitive non-abelian simple group T can be computed in polynomial time. Next, we will focus on obtaining a polynomial time algorithm for computing $N_{S_n}(H)$ which runs efficiently in practice.

6.4 Isomorphisms induced by conjugations

In this section, we give an alternative way of deciding if an isomorphism ϕ between non-abelian simple groups H_i and H_j with supports Δ_i and Δ_j respectively is induced by conjugation in $\mathrm{Sym}(\Delta_i \cup \Delta_j)$. We need not assume that Δ_i and Δ_j are disjoint or that $H_i \neq H_j$. Note that here we are considering a special case of Lemma 2.1.19, where the domain and the image of the isomorphism are non-abelian simple. The focus of this section will not be on the complexity, but instead, we aim to build the mathematical framework for the implemented algorithm which we shall describe in Section 6.5.

For the rest of the section, we shall adopt the following notation and assumptions.

Notation 6.4.1. Let $\Gamma = \{1, 2, \ldots, m\}$, and let $T \leq \mathrm{Sym}(\Gamma)$. Let $H \leq \mathrm{Sym}(\Omega)$ be a group in $\mathfrak{InP}(T)$.
Let $H = H_1 \times H_2 \times \ldots \times H_r$ be the finest disjoint direct product decomposition of H. For all $1 \leq i \leq r$, let $\Delta_i = \mathrm{Supp}(H_i)$, and so $\Omega = \dot\bigcup_{i=1}^{r} \Delta_i$.
Further assume the H_i have the same number of orbits s, and let $\{\Delta_{i1}, \Delta_{i2}, \ldots, \Delta_{is}\}$ be the orbits of H_i for each $1 \leq i \leq r$.
For all $1 \leq a \leq s$, let H_{ia} denote $H|_{\Delta_{ia}}$, and for all $h_i \in H_i$, let h_{ia} denote $h_i|_{\Delta_{ia}}$.

We will be writing maps on the right. We will also be taking the natural inclusion map of $\mathrm{Sym}(I)$ into $\mathrm{Sym}(J)$ for all subsets $I \subset J$. Our goal is to decide if a given isomorphism $\phi : H_i \to H_j$ is induced by conjugation in $\mathrm{Sym}(\Delta_i \cup \Delta_j)$ and exhibit an element giving rise to such conjugation, should it exist.

When computing the normalisers $N_{S_n}(H)$ using Theorem 6.3.7, we find that the slowest part is determining if an isomorphism $\phi : H_i \to H_j$ is induced by conjugation in $\mathrm{Sym}(\Delta_i \cup \Delta_j)$. Determining this using the procedure described in Lemma 2.1.19 gives time polynomial in $2sm$. In this section, we see how we can improve this. We require some preprocessing consisting of polynomially many applications of Lemma 2.1.19 for isomorphism between groups of degree m, which has time polynomial in $2m$. However, as we will see in Table 6.2, this gives significant improvement in terms of computation time, where the speedups are more evident as s gets larger.

The improvement made is by first testing if the given isomorphism is induced by conjugation using Proposition 6.4.3. For each $1 \leq i \leq r$ and $1 \leq a \leq s$, we will associate the H_i-orbit Δ_{ia} with an automorphism ω_{ia} of T. We will also associate the isomorphism ϕ with an automorphism β of T. We will see how we use these ω_{ia} and β to decide if ϕ is induced by conjugation in Proposition 6.4.3. First, we define and give certain properties of the ω_{ia} and β.

Lemma 6.4.2. *Let $1 \leq i, j \leq r$ and $\phi : H_i \to H_j$ be an isomorphism. For all $1 \leq a \leq s$, let $d_{ia} \in \mathrm{Sym}(\Gamma \cup \Delta_{ia})$ such that $T^{d_{ia}} = H_{ia}$, and let $\delta_{ia} : T \to H_{ia}$ be the isomorphism induced by the conjugation of d_{ia}. For all $1 \leq a \leq s$, let $\psi_{ia} : H_{i1} \to H_{ia}$ be the isomorphism defined by $h_{ia} = (h_{i1})\psi_{ia}$ for all $h_i \in H_i$, and let $\phi_a : H_{ia} \to H_{ja}$ be the isomorphism defined by $(h_{ia})\phi_a = ((h_i)\phi)|_{\Delta_{ja}}$ for all $h_i \in H_i$.*

1. *Let $\omega_{i1} = 1 \in \mathrm{Aut}(T)$ and let $\omega_{ia} := \delta_{i1}\psi_{ia}\delta_{ia}^{-1} \in \mathrm{Aut}(T)$ for all $2 \leq a \leq s$. Then for all $h_i \in H_i$, we have $h_{ia} = (h_{i1})\delta_{i1}^{-1}\omega_{ia}\delta_{ia}$.*

2. *Let $\beta := \delta_{i1}\phi_1\delta_{j1}^{-1} \in \mathrm{Aut}(T)$. Then for all $1 \leq a \leq s$ and $h_i \in H_i$, we have*

$(h_{ia})\phi_a = (h_{i1})\delta_{i1}^{-1}\beta\omega_{ja}\delta_{ja}$, and so

$$(h_i)\phi = \prod_{a=1}^{k}(h_{i1})\delta_{i1}^{-1}\beta\omega_{ja}\delta_{ja}.$$

Proof. Part 1: Let $h_i \in H_i$. Then $(h_{i1})\delta_{i1}^{-1}\omega_{ia}\delta_{ia} = (h_{i1})\psi_{ia} = h_{ia}$.
Part 2: Let $h_i \in H_i$. If $a = 1$, then, since $\omega_{j1} = 1$, we have

$$(h_{i1})\delta_{i1}^{-1}\beta\omega_{ja}\delta_{ja} = (h_{i1})\delta_{i1}^{-1}(\delta_{i1}\phi_1\delta_{ja}^{-1})1\delta_{ja} = (h_{i1})\phi_1.$$

Suppose now that $a \neq 1$. Let $h_j = (h_i)\phi \in H_j$, so $(h_{ia})\phi_a = h_{ja}$. Then

$$(h_{i1})\delta_{i1}^{-1}\beta\omega_{ja}\delta_{ja} = (h_{i1})\delta_{i1}^{-1}(\delta_{i1}\phi_1\delta_{j1}^{-1})\omega_{ja}\delta_{ja} = (h_{i1})\phi_1\delta_{j1}^{-1}\omega_{ja}\delta_{ja} = (h_{j1})\delta_{j1}^{-1}\omega_{ja}\delta_{ja}.$$

By Part 1, $(h_{j1})\delta_{j1}^{-1}\omega_{ja}\delta_{ja} = h_{ja}$. The last assertion follows from the observation that $(h_i)\phi = \prod_{a=1}^{k}(h_{ia})\phi_a$. □

Recall $\overline{N_{S_m}(T)}$ from Notation 6.2.6. We will decide if ϕ is induced by conjugation by comparing certain cosets of $\overline{N_{S_m}(T)}$ in $\mathrm{Aut}(T)$, where the coset representatives are constructed from the ω_{la} and β defined in Lemma 6.4.2.

Proposition 6.4.3. *Let $1 \leq i, j \leq r$, and let H_i and H_j be as in Notation 6.4.1. Let $\phi : H_i \to H_j$ be an isomorphism. Let β and ω_{la} for all $1 \leq l \leq r$ and $1 \leq a \leq s$ be as in Lemma 6.4.2. Then ϕ is induced by a conjugation in $\mathrm{Sym}(\Delta_i \cup \Delta_j)$ if and only if, as multisets,*

$$\{\omega_{ia}\overline{N_{S_m}(T)} \mid 1 \leq a \leq s\} = \{\beta\omega_{ja}\overline{N_{S_m}(T)} \mid 1 \leq a \leq s\}.$$

Furthermore, if ϕ is induced by a conjugation of $g \in \mathrm{Sym}(\Delta_i \cup \Delta_j)$, then g maps Ω_{ia} to Ω_{jb} only if $\omega_{ia}\overline{N_{S_m}(T)} = \beta\omega_{jb}\overline{N_{S_m}(T)}$.

Proof. Let Λ be a set of size $|\Delta_j|$ disjoint from Δ_j, and let $d \in \mathrm{Sym}(\Delta_j \cup \Lambda)$ be an involution such that $\Delta_j^d = \Lambda$. Consider $A \coloneqq \{h_i((h_i)\phi)^d \mid h_i \in H_i\}$, which has orbits $\{\Delta_{ia} \mid 1 \leq a \leq s\} \cup \{\Delta_{ja}^d \mid 1 \leq a \leq s\}$. By Lemma 2.1.19, ϕ is induced by a conjugation in $\mathrm{Sym}(\Delta_i \cup \Delta_j)$ if and only if there exists a bijection $\gamma : \{\Delta_{ia} \mid 1 \leq a \leq s\} \to \{\Delta_{ja}^d \mid 1 \leq a \leq s\}$ such that for all $1 \leq a \leq s$, the A-orbits Δ_{ia} and $(\Delta_{ia})\gamma$ are equivalent. We show that A-orbits Δ_{ia} and Δ_{jb}^d are equivalent if and only if $\omega_{ia}\overline{N_{S_m}(T)} = \beta w_{jb}\overline{N_{S_m}(T)}$, from which the result follows.
$\Rightarrow$: Let δ_{la} for all $1 \leq l \leq r$ and $1 \leq a \leq s$ be as in Lemma 6.4.2. Let $t \in T$. Since $(T)\delta_{ia} = H_i|_{\Delta_{i1}}$, there exists $h_i \in H_i$ such that $h_{i1} = (t)\delta_{i1}$. Let $h_j \coloneqq (h_i)\phi \in H_j$, so $h_ih_j^d$ is an element of A. By the forward implication of Lemma 2.1.12, there exists $g \in \mathrm{Sym}(\Delta_{ia} \cup \Delta_{jb}^d)$ such that ${h_{ia}}^g = h_{jb}^d$. By Part 1 of Lemma 6.4.2,

$${h_{ia}}^g = ((h_{i1})\delta_{i1}^{-1}\omega_{ia}\delta_{ia})^g = ((t)\omega_{ia}\delta_{ia})^g,$$

and since $h_{jb} = (h_{ib})\phi_b$, using Part 2 of Lemma 6.4.2,

$$h_{jb}^d = ((h_{i1})\delta_{i1}^{-1}\beta\omega_{jb}\delta_{jb})^d = ((t)\beta\omega_{jb}\delta_{jb})^d.$$

So $(t)\omega_{ia}$ and $(t)\beta\omega_{jb}$ are conjugate in $\mathrm{Sym}(\Gamma)$. Hence $\omega_{ia}\overline{N_{S_m}(T)} = \beta w_{jb}\overline{N_{S_m}(T)}$.
$\Leftarrow$: We shall show that $\Delta_{ia} \equiv_o \Delta_{jb}^d$ using the backward implication of Lemma 2.1.12. Let $\nu \in \overline{N_{S_m}(T)}$. Since $\omega_{ia}\overline{N_{S_m}(T)} = \beta\omega_{jb}\overline{N_{S_m}(T)}$, there exists $\nu' \in \overline{N_{S_m}(T)}$ such that $\omega_{ia}\nu = \beta\omega_{jb}\nu'$. Consider the isomorphism $\delta_{ia}^{-1}\nu\nu'^{-1}\delta_{jb}$ from H_{ia} to H_{jb}. Then there exists $c \in \mathrm{Sym}(\Delta_{ia} \cup \Delta_{jb})$ inducing the isomorphism $\delta_{ia}^{-1}\nu\nu'^{-1}\delta_{jb}$. Let g be the involution in $\mathrm{Sym}(\Delta_{ia} \cup \Delta_{jb})$ such that $\delta^g = \delta^c$ for all $\delta \in \Delta_{ia}$. We will show that for all $h_i \in H_i$ and $h_j := (h_i)\phi \in H_j$, we have $(h_{ia})^{gd} = {h_{jb}}^d$. Since $h_i h_j^d \in A$, by Lemma 2.1.12, $\Delta_{ia} \equiv_o \Delta_{jb}^d$.
Let $h_i \in H_i$ and $h_j := (h_i)\phi \in H_j$. Then

$$\begin{aligned}
(h_{ia})\delta_{ia}^{-1}\nu\nu'^{-1}\delta_{jb} &= (h_{i1})\delta_{i1}^{-1}\omega_{ia}\delta_{ia}\delta_{ia}^{-1}\nu\nu'^{-1}\delta_{jb} \quad \text{by Part 1 of Lemma 6.4.2}\\
&= (h_{i1})\delta_{i1}^{-1}\omega_{ia}\nu\nu'^{-1}\delta_{jb}\\
&= (h_{i1})\delta_{i1}^{-1}\beta\omega_{jb}\delta_{jb} \quad \text{since } \omega_{ia}\nu = \beta\omega_{jb}\nu'\\
&= h_{jb} \quad \text{by Part 2 of Lemma 6.4.2.}
\end{aligned}$$

Therefore ${h_{ia}}^{gd} = {h_{jb}}^d$. □

Lastly, we give corollaries for the cases when $|\mathrm{Aut}(T) : \overline{N_{S_m}(T)}| \leq 2$, which can be used to create faster algorithms, as we do not need to compare of cosets of $\overline{N_{S_m}(T)}$ in $\mathrm{Aut}(T)$, as in Proposition 6.4.3.

Corollary 6.4.4. *Suppose that* $\mathrm{Aut}(T) = \overline{N_{S_m}(T)}$. *Let* $U, U' \in \mathfrak{InP}(T)$ *such that* $U \cong U' \cong T$. *Then* U *and* U' *are conjugate in* $\mathrm{Sym}(Supp(U) \cup Supp(U'))$ *if and only if the number of orbits of* U *is equal to the number of orbits of* U'.

Proof. By Lemma 6.2.7, all orbits of U are equivalent. Similarly, all orbits of U' are equivalent. The result then follows from Proposition 6.4.3. □

Corollary 6.4.5. *Suppose that* $|\mathrm{Aut}(T) : \overline{N_{S_m}(T)}| \leq 2$. *Let* $U, U' \in \mathfrak{InP}(T)$ *such that* $U_1 \cong U_2 \cong T$. *Then all isomorphisms* $\phi : U \to U'$ *are induced by conjugation in* $\mathrm{Sym}(Supp(U_1) \cup Supp(U_2))$ *if and only if one of the following holds:*

1. *U and U' have the same number of orbits, all of which are equivalent.*

2. *Each of U and U' has two equivalence classes under $\equiv_o$, where all the classes have the same size.*

Proof. $\Leftarrow$: If Part 1 holds, then the result follows from Proposition 6.4.3. Suppose now that Part 2 holds. Since $\mathrm{Aut}(T)/\overline{N_{S_m}(T)}$ forms a group, $\beta \in \mathrm{Aut}(T)$ permutes the cosets of $\overline{N_{S_m}(T)}$ in $\mathrm{Aut}(T)$. So the results follows from Proposition 6.4.3.
$\Rightarrow$: By Lemma 6.2.7, each of U and U' has at most two equivalence classes under $\equiv_o$.

Since $\beta \in \mathrm{Aut}(T)$ either fixes or swaps the cosets of $\overline{N_{S_m}(T)}$, the sizes of the equivalence classes under $\equiv_o$ of U and U' must be the same. □

6.5 Algorithm

In this section, we present an algorithm for computing the normaliser $N_{S_n}(H)$ for $H \leq S_n$ where H is in class $\mathfrak{InP}(T)$. We show that Algorithm 11 computes the normaliser $N_{S_n}(H)$ and runs in polynomial time. In Section 6.7, we will show its performance in practice.

Algorithm 11 uses the procedure in Algorithm 12. We first show that the procedure is correct and runs in polynomial time. Note that Algorithm 12 is an alternative algorithm of the procedure described in Lemma 2.1.19 to decide if an isomorphism between non-abelian simple groups is induced by conjugation, and uses results in Section 6.4.

Lemma 6.5.1. *Algorithm 12 is correct and runs in polynomial time.*

Proof. Let $\phi : H_i \to H_j$ be an isomorphism. Let S be a left transversal of $\overline{N_{S_m}(T)}$ in $\mathrm{Aut}(T)$ and let the ω_{la} be as in Lemma 6.4.2. We first prove the correctness of the algorithm. Since conjugation preserves orbits, if the condition in line 2 holds, then ϕ is not induced by conjugation. Suppose otherwise. Then by Proposition 6.4.3, ϕ is induced by conjugation in $\mathrm{Sym}(Supp(H_i) \cup Supp(H_j))$ if and only if $S_i = S_j$. So the algorithm is correct.
For the complexity result, by Proposition 2.1.2, the orbit structure of a group can be computed in polynomial time. By Lemma 2.1.19, line 11 can be computed in polynomial time. It remains to show that S_i and S_j can be computed in polynomial time.
We shall show that given $\omega \in \mathrm{Aut}(T)$, we can find $\sigma \in S$ such that $\sigma\overline{N_{S_m}(T)} = \omega\overline{N_{S_m}(T)}$ in polynomial time. We do so by considering all $\sigma \in S$ and checking if $\sigma^{-1}\omega \in \overline{N_{S_m}(T)}$. That is, we check if the automorphism $\sigma^{-1}\omega$ is induced by conjugation in S_m. By Lemma 2.1.19, this can be decided in polynomial time. Since $|\mathrm{Aut}(T)|/|\overline{N_{S_m}(T)}| = |Out(T)| \leq |S|$, the size of S is polynomial in m. So we consider at most polynomially many $\sigma \in S$ and hence this procedure runs in polynomial time. □

Before we show that Algorithm 11 is correct and runs in polynomial time, we first show that a left transversal of $\overline{N_{S_m}(T)}$ in $\mathrm{Aut}(T)$ can be computed in polynomial time. By Corollary 6.1.6 and Lemma 6.1.7, a transversal of $Inn(T)$ in $\mathrm{Aut}(T)$ can be obtained in polynomial time.

Lemma 6.5.2. *Let $T = \langle X \rangle \leq S_m$. Let R be a right transversal of $Inn(T)$ in* $\mathrm{Aut}(T)$*, where each element of R is defined by the images of X. Given X and R, in polynomial time, we can compute a left transversal S of $\overline{N_{S_m}(T)}$ in* $\mathrm{Aut}(T)$*, where each element of S is defined by the images of X.*

Proof. Note that $R^{-1} := \{\omega^{-1} \mid \omega \in R\}$ forms a left transversal of $Inn(T)$ in $\mathrm{Aut}(T)$. Initialise S as the empty set $\{\}$. We compute S by considering all $\omega \in R^{-1}$, and add ω

Algorithm 11 Computing the normaliser $N_{S_n}(H)$ for $H \in \mathfrak{InP}(T)$

Input: A generating set X of $H \leq S_n$ such that $H \in \mathfrak{InP}(T)$.

Output: $N_{S_n}(H)$.

```
 1: Find a non-abelian simple group $T$ such that $H \in \mathfrak{InP}(T)$
                                                   ▷ using Proposition 6.3.6
 2: $R_T \leftarrow$ right transversal of $Inn(T)$ in $\mathrm{Aut}(T)$        ▷ see Lemma 6.1.7
 3: $S \leftarrow$ left transversal of $\overline{N_{S_m}(T)}$ in $\mathrm{Aut}(T)$       ▷ as in Lemma 6.5.2
 4: Compute the $H_i$ such that $H = H_1 \times H_2 \times \ldots \times H_r$ is the finest disjoint direct
    product decomposition of $H$                                ▷ so each $H_i \cong T$
 5: For $1 \leq i \leq r$, let $\{\Delta_{i1}, \Delta_{i2}, \ldots, \Delta_{is_i}\}$ be the orbits of $H_i$
 6: For all $1 \leq i \leq r$ and $1 \leq a \leq s_i$, find $d_{ia}$ s.t. $T^{d_{ia}} = H|_{\Delta_{ia}}$   ▷ as in Lemma 6.3.5
 7: For all $1 \leq i \leq r$ and $1 \leq a \leq s_i$, let $\delta_{ia}$ and $\omega_{ia}$ be as in Lemma 6.4.2
 8: Compute $C_{S_n}(H)$ as in Theorem 2.1.16
 9: Initialise $N = \langle H, C_{S_n}(H) \rangle$
10: for $1 \leq i \leq r$ do
11:     for $i \leq j \leq r$ do
12:         if $i \neq j$ then
13:             Find an isomorphism $\phi : H_i \to H_j$              ▷ using Corollary 6.1.6
14:         else
15:             Let $\phi : H_i \to H_j$ be the identity map
16:         end if
17:         $R \leftarrow$ transversal of $Inn(H_i)$ in $\mathrm{Aut}(H_i)$              ▷ see Lemma 6.1.7
18:         for $\omega \in R$ do
19:             Define isomorphism $\phi_\omega : H_i \to H_j$ by $h_i \mapsto \phi(h_i^\omega)$
20:             if IsConjugatorIsom($\phi_\omega, S, \{\omega_{ia}\}_{1 \leq a \leq s_i}, \{\omega_{ja}\}_{1 \leq a \leq s_j}$) $\neq$ FAIL then
21:                 if $i \neq j$ then
22:                     $g \leftarrow$ IsConjugatorIsom($\phi_\omega, S, \{\omega_{ia}\}_{1 \leq a \leq s_i}, \{\omega_{ja}\}_{1 \leq a \leq s_j}$)
23:                     $c \leftarrow$ involution in $S_n$ with support $Supp(H_i) \cup Supp(H_j)$ such that
                              $\delta^c = \delta^g$ for all $\delta \in \Delta_i$
24:                     $N \leftarrow \langle N, c \rangle$
25:                     break                  ▷ Only one conjugating element needed
26:                 else
27:                     $N \leftarrow \langle N,$ IsConjugatorIsom($\phi_\omega$)$\rangle$
28:                 end if
29:             end if
30:         end for
31:     end for
32: end for
33: return $N$
```

Algorithm 12 Determine if a given isomorphism is induced by a conjugation

Input: Isomorphism $\phi : H_i \to H_j$, defined by the images of the generators of H_i; left transversal S of $\overline{N_{S_m}(T)}$ in $\mathrm{Aut}(T)$; the ω_{la} as in Lemma 6.4.2.
Output: If ϕ is not induced by conjugation in $\mathrm{Sym}(\mathit{Supp}(H_i) \cup \mathit{Supp}(H_j))$, outputs FAIL; Otherwise, outputs $g \in \mathrm{Sym}(\mathit{Supp}(H_i) \cup \mathit{Supp}(H_j))$ such that $\phi(h_i) = h_i^g$ for all $h_i \in H_i$.

```
 1: procedure IsConjugatorIsom($\phi, S, \{\omega_{ia}\}_{1\le a\le s_i}, \{\omega_{ja}\}_{1\le a\le s_j}$)
 2:     if number of orbits of $H_i \neq$ number of orbits of $H_j$ then
 3:         return Fail
 4:     else
 5:         Let $\beta \in \mathrm{Aut}(T)$ be as in Lemma 6.4.2
 6:         $S_i \leftarrow$ multiset $\{\sigma_a \in S$ s.t. $\sigma_a\overline{N_{S_m}(T)} = \omega_{ia}\overline{N_{S_m}(T)} \mid 1 \le a \le s_i\}$
 7:         $S_j \leftarrow$ multiset $\{\sigma_a \in S$ s.t. $\sigma_a\overline{N_{S_m}(T)} = \beta\omega_{ja}\overline{N_{S_m}(T)} \mid 1 \le a \le s_j\}$
 8:         if multisets $S_i \neq S_j$ then
 9:             return Fail        ▷ $\phi$ not induced by conjugation by Proposition 6.4.3
10:         else
11:             Find $g \in S_n$ with support $\mathit{Supp}(H_i) \cup \mathit{Supp}(H_j)$ inducing $\phi$
                                                            ▷ using Lemma 2.1.19
12:             return $g$
13:         end if
14:     end if
15: end procedure
```

to S if there does not exists $\sigma \in S$ such that $\sigma\overline{N_{S_m}}(T) = \omega\overline{N_{S_m}}(T)$. To check that, we check if $\omega^{-1}\sigma \in \overline{N_{S_m}(T)}$. That is, we check if $\omega^{-1}\sigma$ is induced by conjugation in S_m. By Lemma 2.1.19, this can be done in polynomial time.
Since we consider at most $|R|^2$ pairs of σ and ω and $|R|^2 \le 4m$ by Lemma 6.1.2, the algorithm runs in polynomial time. □

Finally, we show that Algorithm 11 is correct and runs in polynomial time.

Theorem 6.5.3. *Algorithm 11 computes $N_{S_n}(H)$ in polynomial time.*

Proof. Let N be the output of Algorithm 11. We first show that $N = N_{S_n}(H)$ using Proposition 4.2.6.
$\le$: Since $\langle H, C_{S_n}(H)\rangle \le N_{S_n}(H)$, it remains to show that lines 24 and 27 only add normalising elements to N. Since ϕ is an isomorphism, by Lemma 6.5.1, the function IsConjugatingIsom outputs $g \in \mathrm{Sym}(\mathit{Supp}(H_i) \cup \mathit{Supp}(H_j))$ such that $(H_i|_{\mathit{Supp}(H_i)})^g = H_j|_{\mathit{Supp}(H_j)}$. If $i = j$, then $g \in N_{\mathrm{Sym}(\Delta_i)}(H_i)$ and so by considering the natural inclusion of $\mathrm{Sym}(\Delta_i)$ in S_n, we have $g \in N_{S_n}(H)$.
Suppose now that $i \neq j$. Then c in line 23 conjugates H_i to H_j and H_j to H_i, and fixes all other H_a where $a \neq i, j$. Since $H = H_1 \times H_2 \times \ldots \times H_r$, we have $c \in N_{S_n}(H)$.
$\ge$: By Proposition 4.2.6, to show $N \ge N_{S_n}(H)$, it suffices to show that the following conditions hold:

1. $N_{\mathrm{Sym}(\mathit{Supp}(H_i))}(H_i) \subseteq N$ for all $1 \le i \le r$.

2. If H_i and H_j are conjugate in $\mathrm{Sym}(Supp(H_i) \cup Supp(H_j))$, then there exists $g \in N$ such that $H_i^g = H_j$.

For the first assertion, let $1 \leq i = j \leq r$. Then ϕ is the identity map, so the ϕ_ω are outer automorphisms of H_i. Then by Lemma 6.5.1, N contains a set

$$\{x_\omega \in \mathrm{Sym}(Supp(H_i)) \mid \forall g \in G \text{ such that } g^{x_\omega} = g^\omega \text{ for some } \omega \in R\},$$

where R is a transversal of $Inn(H_i)$ in $\mathrm{Aut}(H_i)$. Since $H_i \leq H$ and $C_{\mathrm{Sym}(Supp(H_i))}(H_i) \leq C_{S_n}(H)$, by Lemma 1.3.9, $N_{\mathrm{Sym}(Supp(H_i))}(H_i) \subseteq N$.
For the latter assertion, let $1 \leq i, j \leq r$ where $i \neq j$ and H_i and H_j are conjugate in $\mathrm{Sym}(Supp(H_i) \cup Supp(H_j))$. By Lemma 6.3.3, there exists $\omega \in R$ such that ϕ_ω is induced by conjugation in $\mathrm{Sym}(Supp(H_i) \cup Supp(H_j))$. The result then follows from Lemma 6.5.1.

For the complexity claim, line 1 is polynomial by Proposition 6.3.6. By Lemma 6.1.7, we get R_T in line 2 in polynomial time. By Lemma 6.5.2, S in line 3 can be computed in polynomial time. We compute the finest disjoint direct product decomposition of H using Algorithm 3, which runs in polynomial time by Theorem 3.0.2. By Proposition 2.1.2, computing the orbits of a group can be done in polynomial time, so line 5 runs in polynomial time. Since T and H_{ia} are permutation isomorphic for all $1 \leq i \leq r$ and $1 \leq a \leq s_i$, and $rs_i \leq n^2$, by Lemma 6.3.5, the d_{ia} in line 6 can be found in polynomial time. By Theorem 2.1.16, $C_{S_n}(H)$ can be computed in polynomial time, so line 8 runs in polynomial time. Lines 10 and 11 give polynomially many iterations since $r \leq k \leq n$ and $s_i \leq k \leq n$. By Corollary 6.1.6, line 13 is in polynomial time. By Lemma 6.1.7, we can obtain the R in line 17 in polynomial time. Lastly, line 20 is in polynomial time by Lemma 6.5.1. □

6.6 Extension: Groups in class $\mathfrak{InP}(S_m)$

In this section, we present a new algorithm for computing the normaliser $N_{S_n}(H)$ of $H \leq S_n$ for $H \in \mathfrak{InP}(S_m)$, where $m \geq 5$ and $m \neq 6$. First, we set up the notation we will use throughout the section.

Notation 6.6.1. Let $m \geq 5$ such that $m \neq 6$. Denote by S_m the symmetric group acting naturally on m points. Let H be a subgroup of S_n in class $\mathfrak{InP}(S_m)$ with orbits $\Omega_1, \Omega_2, \ldots, \Omega_k$. Let $\Omega = \cup_{i=1}^k \Omega_i$ and assume that $\Omega = \{1, 2, \ldots, n\}$.
Let $G = H|_{\Omega_1} \times H|_{\Omega_2} \times \ldots \times H|_{\Omega_k}$, where we identify the direct product as a subgroup of $\mathrm{Sym}(\Omega)$. So $H \leq G$ and $G \cong S_m^k$.
Let L be as in Lemma 4.3.8, so $L \cong S_m \wr S_k$ and $N_{S_n}(H) \leq L$.
Denote by A_m the alternating group acting naturally on m points. Let C be the subgroup of G in class $\mathfrak{InP}(A_m)$ such that $C \cong A_m^k$. So $C|_{\Omega_i} = \mathrm{Alt}(\Omega_i)$ for all $1 \leq i \leq k$.
Let $A := H \cap C$.

Next, we show that the subgroup A of H is in class $\mathfrak{InP}(A_m)$. Note that we will be taking natural inclusion maps, so $\mathrm{Sym}(\Omega_i) \leq \mathrm{Sym}(\Omega)$ for all $1 \leq i \leq k$.

Lemma 6.6.2. *Let H, A and Ω_i be as in Notation 6.6.1. Then*

1. *A is a subdirect product of C, and so $A \in \mathfrak{InP}(A_m)$.*
2. *$A = soc(H)$. Hence $A \trianglelefteq N_{S_n}(H)$, and $N_{S_n}(H) \leq N_{S_n}(A)$.*

Proof. Part 1: By the definition of A, we have $A \leq C$. It remains to show that $A|_{\Omega_i} = \mathrm{Alt}(\Omega_i)$ for all $1 \leq i \leq k$.
Let $1 \leq i \leq k$. Since $H \leq G$ and $C \trianglelefteq G$, we have $A = C \cap H$ is normal in H. So $A|_{\Omega_i} \trianglelefteq H|_{\Omega_i}$, therefore $A|_{\Omega_i} = 1, \mathrm{Alt}(\Omega_i)$ or $\mathrm{Sym}(\Omega_i)$. Since $A \leq C$, the projection $A|_{\Omega_i}$ is a subgroup of $\mathrm{Alt}(\Omega_i)$, so $A|_{\Omega_i} \neq \mathrm{Sym}(\Omega_i)$. We shall show that $A|_{\Omega_i} \neq 1$.
Since $H|_{\Omega_i} = \mathrm{Sym}(\Omega_i)$, there exists $h \in H$ such that $h|_{\Omega_i} \in \mathrm{Alt}(\Omega_i)$ and $h|_{\Omega_i}$ is non-trivial and not an involution. Then $h^2 \in A$ and $(h^2)|_{\Omega_i} \neq 1$. So $A|_{\Omega_i} \neq 1$.
Part 2: Since $A \in \mathfrak{InP}(A_m)$, by Proposition 6.2.4, there exist subgroups $N_1, N_2, \ldots, N_r$ of A such that each N_i is isomorphic to A_m and A is the (internal) direct product of the N_i. Observe that each N_i is normal in G. So the N_i are normal subgroups of H. Since the N_i are simple, they are minimal normal subgroups of H, and hence are contained in $soc(H)$. So $A = N_1 \times N_2 \times \ldots \times N_r$ is also contained in $soc(H)$.
To show that $soc(H) \leq A$, we show that all minimal normal subgroups of H are contained in A. Let N be a minimal normal subgroup of H. Then for all $1 \leq i \leq k$, we have $N|_{\Omega_i} \trianglelefteq H|_{\Omega_i}$, so $N|_{\Omega_i}$ can be either $1, \mathrm{Alt}(\Omega_i)$ or $\mathrm{Sym}(\Omega_i)$. Since a minimal normal subgroup is a direct product of isomorphic simple groups, $N|_{\Omega_i} \neq \mathrm{Sym}(\Omega_i)$ for all $1 \leq i \leq k$. Therefore $N \subseteq A$.
Lastly, since $soc(H)$ is a characteristic subgroup of H, the last two assertions follow. □

In Lemma 6.6.5, we will construct a complement of A in H. Before that, we give some lemmas we will be using in the proof. We start with a well-known fact on the automorphism groups of symmetric groups. For more information, see, for example, [McC14].

Lemma 6.6.3. *Let $t \neq 2, 6$. Then all automorphisms of S_t are inner.*

Recall the definition of equivalent orbits from Definition 2.1.10. Recall also that we identify $\mathrm{Sym}(\Omega_i)$ and $\mathrm{Sym}(\Omega_j)$ as subgroups of $\mathrm{Sym}(\Omega_i \cup \Omega_j)$.

Lemma 6.6.4. *Let H, A be as in Notation 6.6.1 and let Ω_i and Ω_j be H-orbits. If Ω_i and Ω_j are equivalent orbits of A, then Ω_i and Ω_j are equivalent orbits of H.*

Proof. Without loss of generality, suppose that $i = 1$ and $j = 2$. Let $N := H_{(\Omega_1)}|_{\Omega_2}$. Then by Corollary 1.4.4, we have $N \trianglelefteq H|_{\Omega_2}$. So $N = 1, \mathrm{Alt}(\Omega_2)$ or $\mathrm{Sym}(\Omega_2)$.
We will first show that $N = 1$. Aiming for a contradiction, suppose that N contains $\mathrm{Alt}(\Omega_2)$. Then there exists $h \in H_{(\Omega_1)}$ such that $h|_{\Omega_2}$ is non-trivial and not an involution.

It follows that $h' := h^2$ is an element of A where $h'|_{\Omega_1} = 1$ and $h'|_{\Omega_2} \neq 1$. Then Lemma 2.1.12 gives a contradiction. Hence $N = 1$.
Let $\theta_1 : H|_{\Omega_1} \to (H|_{\Omega_2})/N$ be the surjective homomorphism as in Corollary 1.4.4. Since $N = 1$, the map $\tilde{\theta_1} : H|_{\Omega_1} \to H|_{\Omega_2}$ defined by $h|_{\Omega_1} \mapsto h|_{\Omega_2}$ is also a surjective homomorphism. Since $H|_{\Omega_1} \cong H|_{\Omega_2}$ and θ_1 is a surjective homomorphism, the map $\tilde{\theta_1}$ is an isomorphism. Since $H|_{\Omega_1} \cong S_m$, by Lemma 6.6.3, $\tilde{\theta_1}$ is induced by conjugation in $\mathrm{Sym}(\Omega_1 \cup \Omega_2)$. That is, there exists $c \in \mathrm{Sym}(\Omega_1 \cup \Omega_2)$ such that for all $h \in H$, we have $h|_{\Omega_2} = \tilde{\theta_1}(h|_{\Omega_1}) = (h|_{\Omega_1})^c$. Finally, by Lemma 2.1.12, Ω_1 and Ω_2 are equivalent orbits of H. □

Now we show that we can compute a complement of A in H by taking some point stabilisers of H. We will identify the direct product of permutation groups with disjoint supports as a subgroup of the symmetric group over the disjoint union of the supports of the direct factors. Recall that we will always be taking the inclusion maps $\mathrm{Sym}(\Gamma) \leq \mathrm{Sym}(\Delta)$ for all $\Gamma \subseteq \Delta$.

Lemma 6.6.5. *Let H and A be as in Notation 6.6.1. Fix a 2-subset Γ_1 of Ω_1. Let $F_1 := H_{(\Omega_1 \setminus \Gamma_1)}$. For $2 \leq i \leq k$, let*

$$F_i = \begin{cases} F_{i-1} & \text{if } |Supp(F_{i-1}|_{\Omega_i})| = 2, \\ (F_{i-1})_{(\Omega_i \setminus \Gamma_i)} & \text{for some fixed 2-subset } \Gamma_i \text{ of } \Omega_i, \text{ otherwise.} \end{cases}$$

Let $F := F_k$. Then $H = AF$.

Proof. For all $1 \leq i \leq k$, let Δ_i be as in Corollary 1.4.4. We will inductively show that $H|_{\Delta_i} = A|_{\Delta_i} F_i|_{\Delta_i}$. For the base case, by Part 1 of Lemma 6.6.2, $A|_{\Delta_1} = \mathrm{Alt}(\Omega_1)$. Since $H|_{\Delta_1} = \mathrm{Sym}(\Omega_1)$, by letting $g_1 \in \mathrm{Sym}(\Omega_1)$ be the transposition with support Γ_1, we have $F_1|_{\Delta_1} = \langle g_1 \rangle$. Hence

$$H|_{\Delta_1} = \mathrm{Sym}(\Omega_1) = \mathrm{Alt}(\Omega_1)\langle g_1 \rangle = A|_{\Delta_1} F_1|_{\Delta_1}.$$

For the inductive step, for notational convenience, we consider step k. By letting $\Delta = \Delta_{k-1}$ for notational convenience, assume that $H|_\Delta = A|_\Delta F_{k-1}|_\Delta$. Let N_k, R_k, θ_{k-1} and φ_{k-1} be as in Corollary 1.4.4. Then, by considering $\mathrm{Sym}(\Omega_k)$ as a subgroup of $\mathrm{Sym}(\Omega)$, we have $H = \langle \varphi_{k-1}(H|_\Delta), N_k \rangle$ and $N_k \trianglelefteq H|_{\Omega_k}$, so N_k is $\mathrm{Sym}(\Omega_k)$, $\mathrm{Alt}(\Omega_k)$ or 1. We shall consider the three cases separately.
Case 1: Suppose that $N_k = \mathrm{Sym}(\Omega_k)$. Then by Part 1 of Corollary 1.4.6,

$$\begin{aligned} H &= H|_\Delta \times N_k \\ &= (A|_\Delta F_{k-1}|_\Delta) \times \mathrm{Sym}(\Omega_k) \quad \text{by the inductive hypothesis.} \end{aligned} \tag{6.1}$$

By the definition of A, we have $A = A|_\Delta \times \mathrm{Alt}(\Omega_k)$.
Since F_{k-1} is the stabiliser of a subset of Δ, we have $H_{(\Delta)} \leq F_{k-1}$ and so $H_{(\Delta)}|_{\Omega_k} \leq$

$F_{k-1}|_{\Omega_k}$. However, $H_{(\Delta)}|_{\Omega_k} = N_k = \mathrm{Sym}(\Omega_k)$, so $F_{k-1}|_{\Omega_k} = \mathrm{Sym}(\Omega_k)$ and thus $F_{k-1} = F_{k-1}|_\Delta \times \mathrm{Sym}(\Omega_k)$. Hence $F = F_{k-1}|_\Delta \times \mathrm{Sym}(\Gamma_k)$ for some 2-subset Γ_k of Ω_k.
Therefore, from Equation (6.1), we get

$$\begin{aligned} H &= (A|_\Delta F_{k-1}|_\Delta) \times (\mathrm{Alt}(\Omega_k)\mathrm{Sym}(\Gamma_k)) \\ &= (A|_\Delta \times \mathrm{Alt}(\Omega_k))(F_{k-1}|_\Delta \times \mathrm{Sym}(\Gamma_k)) = AF. \end{aligned}$$

Case 2: Suppose that $N_k = \mathrm{Alt}(\Omega_k)$. Then, by Corollary 1.4.4 and the inductive hypothesis,

$$H = \langle \varphi_{k-1}(A|_\Delta F_{k-1}|_\Delta), N_k\rangle = \langle \varphi_{k-1}(A|_\Delta), \varphi_{k-1}(F_{k-1}|_\Delta), \mathrm{Alt}(\Omega_k)\rangle. \tag{6.2}$$

We will first show that $A = \langle \varphi_{k-1}(A|_\Delta), \mathrm{Alt}(\Omega_k)\rangle$ and $F = \varphi_{k-1}(F_{k-1}|_\Delta)$.
If $\theta_{k-1}(A|_\Delta) \neq 1$, then the kernel of the restriction of θ_{k-1} to $A|_\Delta$ is a normal subgroup of $A|_\Delta$ of index 2. Since $A|_\Delta$ is isomorphic to A_m^r for some r, and normal subgroups of A_m^r is isomorphic to A_m^u for some $u \leq r$, this gives a contradiction. Hence $\theta_{k-1}(A|_\Delta) = 1$.Thus $\varphi_{k-1}(A|_\Delta) \leq A$, and so $\langle \varphi_{k-1}(A|_\Delta), \mathrm{Alt}(\Omega_k)\rangle \leq A$. Observe that as a consequence of Part 1 of Lemma 6.6.2, $A \leq A|_\Delta \times \mathrm{Alt}(\Omega_k)$. Since $\langle \varphi_{k-1}(A|_\Delta), \mathrm{Alt}(\Omega_k)\rangle$ contains $A|_\Delta \times \mathrm{Alt}(\Omega_k)$, we have $A = \langle \varphi_{k-1}(A|_\Delta), \mathrm{Alt}(\Omega_k)\rangle$.
To show that $F = \varphi_{k-1}(F_{k-1}|_\Delta)$, by the definition of F_{k-1}, it follows from Equation (6.2) that $F_{k-1} = \langle \varphi_{k-1}(F_{k-1}|_\Delta), \mathrm{Alt}(\Omega_k)\rangle$. Since the support of $F_{k-1}|_{\Omega_k}$ is not sized 2, we have $F = (F_{k-1})_{(\Omega_k \setminus \Gamma_k)}$ for any fixed 2-subset $\Gamma_k \subset \Omega_k$. Let g be the transposition in $\mathrm{Sym}(\Gamma_k)$, then $\theta_{k-1}(H|_\Delta) = C_2$, so R_k has size 2 and we may let $R_k = \{1, g\}$. Then $F = \varphi_{k-1}(F_{k-1}|_\Delta)$.
Hence, from Equation (6.2), $H = \langle A, F\rangle$. Since $A \trianglelefteq H$, we have $H = AF$.
Case 3: Suppose that $N_k = 1$. Then $H = \varphi_{k-1}(H|_\Delta) = \varphi_{k-1}(A|_\Delta F_{k-1}|_\Delta)$. We first show that there exists $1 \leq j \leq k-1$ such that Ω_j and Ω_k are equivalent A-orbits. By Part 1 of Lemma 6.6.2, $A \in \mathfrak{InP}(A_m)$. Then by Proposition 6.2.4, there exists a partition $P = \langle C_1 \mid C_2 \mid \ldots \mid C_r\rangle$ of $\{1, 2, \ldots, k\}$ such that, by letting $\Lambda_i = \cup_{j \in C_i} \Omega_j$ for all $1 \leq i \leq r$, we have $A|_{\Lambda_i} \cong A_m$ and $A = A|_{\Lambda_1} \times A|_{\Lambda_2} \times \ldots \times A|_{\Lambda_r}$, where we identify the direct product as a subgroup of $\mathrm{Sym}(\Omega)$. Without loss of generality, suppose that $k \in C_1$. If $C_1 = \{k\}$, then, by taking the natural inclusion map of the symmetric groups, $A|_{\Omega_k}$ is a direct factor of A, and so $A|_{\Omega_k} \leq A \leq H$. Since $A|_{\Omega_k}$ fixes Δ, it is a subgroup of $H_{(\Delta)}$, which contradicts the fact that $N_k = 1$. So there exists $1 \leq j \leq k-1$ such that $j \in C_1$. Without loss of generality, suppose that $j = 1$.
Since $\mathrm{Aut}(A_m) \cong S_m = N_{S_m}(A_m)$, by Lemma 6.2.7, all orbits of $A|_{\Lambda_1}$ are equivalent to each other. Then by Lemma 6.6.4, Ω_1 and Ω_k are equivalent H-orbits. By Lemma 2.1.12, there exists $c \in \mathrm{Sym}(\Omega_1 \cup \Omega_k)$ such that for all $h \in H$, we have $h|_{\Omega_k} = (h|_{\Omega_1})^c$. As a consequence, $F_{k-1}|_{\Omega_k}$ has support of size 2, and so $F = F_{k-1}$.

Thus,

$$\begin{aligned} H &= \{(h|_\Delta h|_{\Omega_k}) \mid h \in H\} \\ &= \{h|_\Delta (h|_{\Omega_1})^c \mid h \in H\} \quad \text{since } h|_{\Omega_k} = (h|_{\Omega_1})^c \\ &= \{af((af)|_{\Omega_1})^c \mid a \in A|_\Delta, f \in F_{k-1}|_\Delta\} \quad \text{by the inductive hypothesis} \\ &= \{af(a|_{\Omega_1})^c (f|_{\Omega_1})^c \mid a \in A|_\Delta, f \in F_{k-1}|_\Delta\} \\ &= \{(a(a|_{\Omega_1})^c)(f(f|_{\Omega_1})^c) \mid a \in A|_\Delta, f \in F_{k-1}|_\Delta\} \text{ as } Supp((a|_{\Omega_1})^c) \cap Supp(f) = \emptyset \\ &= AF_{k-1} = AF \quad \text{since } F = F_{k-1}. \end{aligned}$$

Therefore, in all three cases of N_k, we have $H = AF$, as required. □

Since $A \cap F = 1$, the subgroup F is a complement of A in H. Next, we will show that the normaliser $N_{S_n}(H)$ conjugates F to a conjugate in H. We will be using the fact that transpositions in S_m are conjugate in A_m.

Lemma 6.6.6. *Let $m \geq 5$. Then all transpositions in S_m are conjugate in A_m.*

Proof. Let σ_1 and σ_2 be transpositions on S_m. Then there exists $\tau \in S_m$ such that $\sigma_1^\tau = \sigma_2$. If τ is an even permutation then we are done. Suppose otherwise. Since $m \geq 4$, there exists distinct points $1 \leq i, j \leq m$ such that $i, j \notin Supp(\sigma_2)$. Then $\tau(i,j) \in A_m$ and $\sigma_1^{\tau(i,j)} = \sigma_2$. □

Lemma 6.6.7. *Let H and A be as in Notation 6.6.1 and let $F \leq H$ be constructed as in Lemma 6.6.5 such that $H = AF$. Let $\nu \in N_{S_n}(H)$. Then there exists $h \in H$ such that $F^\nu = F^h$.*

Proof. Let $F' = F^\nu$. Firstly, observe that since g permutes the H-orbits, for each $1 \leq i \leq k$, the support of $F'|_{\Omega_i}$ has size 2.
We proceed by induction on k. For the base case, let $g, g' \in \mathrm{Sym}(\Omega_1)$ be transpositions such that $F = \langle g \rangle$ and $F' = \langle g' \rangle$. Then there exists $h \in H = \mathrm{Sym}(\Omega_1)$ such that $g^h = g'$ and so $F^h = F'$.
Let $\Delta := \cup_{i=1}^{k-1} \Omega_i$. Then, by the inductive hypothesis, there exists $\hat{h} \in H|_\Delta$ such that $F'|_\Delta = (F|_\Delta)^{\hat{h}}$. Let $N = H_{(\Delta)}|_{\Omega_k}$. Then, by Corollary 1.4.4, $N = \mathrm{Sym}(\Omega_k), \mathrm{Alt}(\Omega_k)$ or 1.
Case 1: Suppose first that $N = \mathrm{Sym}(\Omega_k)$. Then by Part 1 of Corollary 1.4.6, , $H = H|_\Delta \times N$. Since $H|_\Delta$ and N have disjoint supports Δ and Ω_k respectively, and $F, F' \leq H$, we have that $F = F|_\Delta \times F|_{\Omega_k}$ and $F' = F'|_\Delta \times F'|_{\Omega_k}$. Let $g, g' \in \mathrm{Sym}(\Omega_k)$ be transpositions such that $F|_{\Omega_k} = \langle g \rangle$ and $F'|_{\Omega_k} = \langle g' \rangle$. Then, there exists $\bar{h} \in H|_{\Omega_k} = \mathrm{Sym}(\Omega_k)$ such that $F'|_{\Omega_k} = (F|_{\Omega_k})^{\bar{h}}$. So $h := \hat{h}\bar{h}$ is an element of H where

$$F^h = (F|_\Delta)^{\hat{h}} \times (F|_{\Omega_k})^{\bar{h}} = F'|_\Delta \times F'|_{\Omega_k} = F'.$$

Case 2: Suppose now that $N = \mathrm{Alt}(\Omega_k)$. By the observation in the opening paragraph, there exist transpositions $g, g' \in \mathrm{Sym}(\Omega_k)$ such that $F|_{\Omega_k} = \langle g \rangle$ and $F'|_{\Omega_k} = \langle g' \rangle$.

By Lemma 6.6.6, there exists $\bar{h} \in N$ which conjugates g to g'. Let φ_{k-1} be as in Corollary 1.4.4, then $h := \varphi_{k-1}(\hat{h})\bar{h}$ is an element of H.
Let $f \in F$ and consider $f^h \in H$. Since $h|_\Delta = \hat{h}$, we deduce that $(f^h)|_\Delta = (f|_\Delta)^{\hat{h}} \in F'|_\Delta$. Let f' be an element of F' such that $f'|_\Delta = (f^h)|_\Delta$. Since $F'|_{\Omega_k} = \langle g' \rangle$, such an f' is unique as otherwise $f'|_\Delta$ and $f'|_\Delta g'$ are elements of H and so $g' \in H_{(\Delta)} = \mathrm{Alt}(\Omega_k)$. We will show that $f^h = f'$.
By taking R_k in Corollary 1.4.4 as $\{1, g\}$, the projection $(\varphi_{k-1}(\hat{h}))|_{\Omega_k}$ is in $\langle g \rangle$. Since $f|_{\Omega_k} \in \langle g \rangle$ and $\bar{h}$ conjugates g to g', we have $(f^h)|_{\Omega_k} = (f^{\varphi_{k-1}(\hat{h})\bar{h}})|_{\Omega_k} \in \langle g' \rangle$. Since $F'|_{\Omega_k} = \langle g' \rangle$, we have that $(f^h f'^{-1})|_{\Omega_k}$ is also contained in $\langle g' \rangle$. Now, since f^h and f' have the same projection on Δ, the permutation $f^h f'^{-1}$ pointwise stabilises Δ, and so $f^h f'^{-1} \in \mathrm{Alt}(\Omega_k)$. Therefore $f^h f'^{-1} \in \langle g' \rangle \cap \mathrm{Alt}(\Omega_k) = 1$.
Case 3: Lastly, suppose that $N = 1$. As in Lemma 6.6.5, there exists $1 \le j \le k-1$ such that Ω_j and Ω_k are equivalent orbits of H. Without loss of generality, suppose that $j = 1$. Then, by Lemma 2.1.12, there exists $c \in \mathrm{Sym}(\Omega_1 \cup \Omega_k)$ such that for all $h \in H$, we have $h|_{\Omega_k} = (h|_{\Omega_1})^c$. Let $h = \hat{h}(\hat{h}|_{\Omega_1})^c \in H$. Let $f \in F$ and $f' \in F'$ such that $(f|_\Delta)^{\hat{h}} = f'|_\Delta$. Then

$$(f|_{\Omega^k})^{(\hat{h}|_{\Omega_1})^c} = ((f|_{\Omega_1})^c)^{(c^{-1}\hat{h}|_{\Omega_1}c)} = (f|_{\Omega_1})^{\hat{h}|_{\Omega_1}c} = (f'|_{\Omega_1})^c = f'|_{\Omega_k}.$$

Therefore,

$$f^h = (f|_\Delta f|_{\Omega_k})^h = (f|_\Delta)^{\hat{h}}(f|_{\Omega_k})^{(\hat{h}|_{\Omega_1})^c} = (f'|_\Delta)(f'|_{\Omega_k}) = f' \in F'.$$

So $F^h \le F'$. The reverse inclusion follows from symmetry. □

Let $L \cong S_m \wr S_k$ be as in Notation 6.6.1. We now show that we can compute $N_{S_n}(H)$ by computing the normalisers $N_L(A)$ and $N_L(F)$.

Theorem 6.6.8. *Let H, A and L be as in Notation 6.6.1. Let F be as constructed in Lemma 6.6.5. Then $N_{S_n}(H) = (N_L(F) \cap N_L(A))H$.*

Proof. $\ge$: Clearly $H \le N_{S_n}(H)$. To show that $N_L(F) \cap N_L(A) \le H$, let $h \in H$ and $g \in N_L(F) \cap N_L(A)$. Then by Lemma 6.6.5, there exists $a \in A$ and $f \in F$ such that $h = af$. Thus $h^g = a^g f^g \in AF = H$.
$\le$: Let $g \in N_{S_n}(H)$. Then, by Lemma 6.6.7, there exists $h \in H \le N_{S_n}(H)$ such that $F^g = F^h$ and so gh^{-1} is an element of $N_{S_n}(H)$ which normalises F. That is, $gh^{-1} \in N_{N_{S_n}(H)}(F)$. By Lemma 4.3.8, $N_{S_n}(H) \le L$ and so $N_{S_n}(H) = N_L(H)$. Therefore

$$gh^{-1} \in N_{N_{S_n}(H)}(F) = N_{N_L(H)}(F) = N_L(F) \cap N_L(H).$$

By Part 2 of Lemma 6.6.2, $N_L(H) = L \cap N_{S_n}(H) \le L \cap N_{S_n}(A) = N_L(A)$, hence $gh^{-1} \in N_L(F) \cap N_L(A)$. □

Observe that A and $F|_{Supp(F)}$ are in classes $\mathfrak{InP}(A_m)$ and $\mathfrak{InP}(C_2)$ respectively.

Therefore, we would like to use the algorithms in Section 6.5 and Section 5.4. The following results show how we use Algorithms 11 and 6 to compute normalisers of groups H in $\mathfrak{InP}(S_m)$.

Recall from Notation 6.6.1 that $L \cong S_m \wr S_k$, where $\mathrm{Sym}(\Omega_1) \times \mathrm{Sym}(\Omega_2) \times \ldots \times \mathrm{Sym}(\Omega_k)$ is the subgroup of L which embeds into the base group of the wreath product, and the top group of the wreath product permutes the Ω_i. Note that we identify the direct product above as a subgroup of $\mathrm{Sym}(\Omega)$.

Lemma 6.6.9. *Let H, A and L be as in Notation 6.6.1. Then $N_L(A) = N_{S_n}(A)$.*

Proof. Since $N_{\mathrm{Sym}(\Omega_1)}(A|_{\Omega_1}) = N_{S_m}(A_m) = S_m$, by Lemma 4.3.8, $N_{S_n}(A) \leq L$. □

Lemma 6.6.10. *Let H and L be as in Notation 6.6.1. Let F and the Γ_i be as in Lemma 6.6.5. Let $\Gamma := Supp(F)$. Let $\phi : N_{\mathrm{Sym}(\Gamma)}(F|_\Gamma) \to \mathrm{Sym}(\Omega)$ be an isomorphism where for all $g \in N_{\mathrm{Sym}(\Gamma)}(F|_\Gamma)$,*

1. *$\phi(g)|_\Gamma = g$, and*
2. *for all $1 \leq i, j \leq k$ such that $\Gamma_i^g = \Gamma_j$, the image $\phi(g)$ maps $\Omega_i \backslash \Gamma_i$ to $\Omega_j \backslash \Gamma_j$.*

Then $N_L(F) = \langle \mathrm{Im}\,\phi, \mathrm{Sym}(\Omega_1\backslash\Gamma_1) \times \mathrm{Sym}(\Omega_2\backslash\Gamma_2) \times \ldots \times \mathrm{Sym}(\Omega_k\backslash\Gamma_k)\rangle$, where we identify the direct product as a subgroup of $\mathrm{Sym}(\Omega)$ with support $\mathrm{Sym}(\Omega\backslash\Gamma)$.

Proof. $\geq$: Firstly observe that $\mathrm{Sym}(\Omega_1\backslash\Gamma_1) \times \mathrm{Sym}(\Omega_2\backslash\Gamma_2) \times \ldots \times \mathrm{Sym}(\Omega_k\backslash\Gamma_k)$ is a subgroup of $\mathrm{Sym}(\Omega_1) \times \mathrm{Sym}(\Omega_2) \times \ldots \times \mathrm{Sym}(\Omega_k)$, which is contained in L. As the direct product has support disjoint to the support of F, it normalises F. To show $\mathrm{Im}\,\phi \subseteq N_L(F)$, let $g \in N_{\mathrm{Sym}(\Gamma)}(F|_\Gamma)$. Clearly $\phi(g)$ normalises F, so it remains to show that $\phi(g) \in L$. Observe that if g induces a permutation σ on the Γ_i, then $\phi(g)$ induces the same permutation σ on the Ω_i. So $\phi(g)$ is contained in a subgroup of $\mathrm{Sym}(\Omega)$ isomorphic to $S_m \wr S_k$, where $\mathrm{Sym}(\Omega_1) \times \mathrm{Sym}(\Omega_2) \times \ldots \times \mathrm{Sym}(\Omega_k)$ embeds into the base group of the wreath product and the top group permutes the Ω_i, which is exactly L.
$\leq$: Let $\nu \in N_L(F)$. Since $N_L(F)$ permutes the F-orbits, Γ is a union of orbits of $N_L(F)$. Then $g := \nu|_\Gamma \in N_{\mathrm{Sym}(\Gamma)}(F|_\Gamma)$. Since $\nu \in L$, there exists a permutation $\sigma \in S_k$ induced by the action of ν on the Ω_i. Then g induces the same permutation σ on the Γ_i. It then follows that $\phi(g)$ induces the same permutation σ on the Ω_i. So $\nu\phi(g)^{-1}$ setwise stabilises each of the Ω_i.
Since $\phi(g)|_\Gamma = g = \nu|_\Gamma$, the permutation $\nu\phi(g)^{-1}$ fixes Γ pointwise. So

$$\nu\phi(g)^{-1} \in \mathrm{Sym}(\Omega_1\backslash\Gamma_1) \times \mathrm{Sym}(\Omega_2\backslash\Gamma_2) \times \ldots \times \mathrm{Sym}(\Omega_k\backslash\Gamma_k),$$

where we identify the direct product as a subgroup of $\mathrm{Sym}(\Omega)$ with support $\mathrm{Sym}(\Omega\backslash\Gamma)$. □

Therefore, we compute $N_{S_n}(H)$ for $H \in \mathfrak{InP}(S_m)$ as follows:

1. Compute $A := soc(H)$.

2. Compute a complement F of A in H as in Lemma 6.6.5. Let $L \cong S_m \wr S_k$ be as in Notation 6.6.1.

3. Compute $N_{S_n}(A)$ using Algorithm 11. By Lemma 6.6.9, $N_L(A) = N_{S_n}(A)$.

4. Let $\Gamma = Supp(F)$. Compute $N_{\mathrm{Sym}(\Gamma)}(F|_\Gamma)$ using Algorithm 6. Then compute $N_L(F)$ as in Lemma 6.6.10.

5. Set $N := \langle N_L(F) \cap N_L(A), H\rangle$. By Theorem 6.6.8, $N = N_{S_n}(H)$.

Lastly, note that by Proposition 2.1.17, A can be computed in polynomial time. Also, since $|\mathrm{Aut}(A_m)|/|\overline{N_{S_m}(A_m)}| = 1$, we may use Corollary 6.4.4 for faster computation of $N_{S_n}(A)$.

6.7 Results

In this section, we compare the runtimes of computing $N_{S_n}(H)$ of $H \leq S_n$ in classes $\mathfrak{InP}(T)$ or $\mathfrak{InP}(S_m)$, for transitive non-abelian simple group T and $5 \leq m \neq 6$,using the GAP function NORMALIZER and our new algorithms.

6.7.1 Normaliser of $H \in \mathfrak{InP}(T)$

We shall consider $H \in \mathfrak{InP}(T)$ for the different cases of T in Table 6.1.

T	m	Name	$\|Out(T)\|$	$\|Aut(T)\|/\|\overline{N_{S_m}(T)}\|$
PRIMITIVEGROUP(6,1)	6	$PSL(2,5)$	2	1
PRIMITIVEGROUP(6,3)	6	A_6	4	2
PRIMITIVEGROUP(12,3)	12	$PSL(2,11)$	2	1
PRIMITIVEGROUP(12,2)	12	M_{12}	2	2
PRIMITIVEGROUP(50,3)	50	$PSL(2,49)$	1	1
PRIMITIVEGROUP(50,1)	50	$PSU(3,5)$	6	3

Table 6.1: Transitive non-abelian simple groups T considered for experiments.

For each $T \leq S_m$ and integer k, we generate a group $H \leq S_{mk}$ in class $\mathfrak{InP}(T)$ by first generating a random partition P of $\{1, 2, \dots, k\}$. This partition shall give the supports of the finest disjoint direct factors of H. That is, for each cell C of P, we have $H|_{\cup_{i \in C}\Omega_i}$ is a finest disjoint direct factor of H isomorphic to T.

We generate such a partition P by iteratively computing random partitions P_i of $\{1, 2, \dots, i\}$ for all $1 \leq i \leq k$. Initialise P_1 as $\langle 1\rangle$. Let P_i be a partition of $\{1, 2, \dots, i\}$ and let s be the number of cells of P_i. We generate P_{i+1} by first generating a random number r from $\{1, 2, \dots, s+1\}$. If $r \leq s$, we construct P_{i+1} by adding $i+1$ to the cell of P_i indexed by r. Else if $r = s+1$, we construct P_{i+1} by adding a new cell consists of only $i+1$ to P_i. Finally, we set $P = P_k$.

For each cell C_i of P, we generate a subdirect product H_i of $T^{|C_i|}$ isomorphic to T with orbits $\{\Omega_j \mid j \in C_i\}$ in the following way. First, we fix a generating set X of T. For each $j \in C_i$, let $c_j \in S_n$ such that $\mathit{Supp}(T)^{c_j} = \Omega_j$, and let α_j be a random element of $\mathrm{Aut}(T)$. Then let $H_i := \langle \prod_{j \in C_i} x^{\alpha_j c_j} \mid x \in X \rangle$. Finally, we get $H \in \mathfrak{InP}(T)$ by letting H be the direct product of the H_i.

For each T and k, we consider 10 groups $H \leq S_{mk}$ in class $\mathfrak{InP}(T)$. For each instance of H, we compute its normalisers using both the GAP function NORMALIZER and Algorithm 11. We report the lower quartile, median and upper quartile computation time for both of these algorithms in Figure 6.1.

Next, we compare the algorithms of computing $N_{S_n}(H)$ of $H \leq S_n$ in classes $\mathfrak{InP}(T)$, using the procedure described in Theorem 6.3.7 (NOTESTS) and Algorithm 11 (WITHTESTS). More specifically, NOTESTS is the same as WITHTESTS, but ISCONJUGATORISOM follows the procedure described in Lemma 2.1.19 instead of that in Algorithm 12. We consider the groups $H \in \mathfrak{InP}(T)$ which have 2 disjoint direct factors with the same number of orbits s, for T = PRIMITIVEGROUP(12,3), PRIMITIVEGROUP(12,2). Each of these $H \leq S_{24s}$ is generated by first generating a partition P of $\{1, 2, \dots, 2s\}$ by first randomly shuffling the list $l := [1, 2, \dots, 2s]$, and taking P as the partition with cells $C_1 := \{l[i] \mid 1 \leq i \leq s\}$ and $C_2 := \{l[i] \mid s + 1 \leq i \leq 2s\}$. Then H is generated from P as before. The result is shown in Table 6.2.

s	NOTESTS	WITHTESTS
1	0.125	0.25
2	0.258	0.3435
3	0.7815	0.5465
4	2.0625	0.735
5	4.4535	1.0545
6	7.6875	0.953
7	13.5315	1.063
8	24.0235	1.297
9	38.0315	1.477
10	58.133	1.5935

(a) T = PRIMITIVEGROUP(12,3)

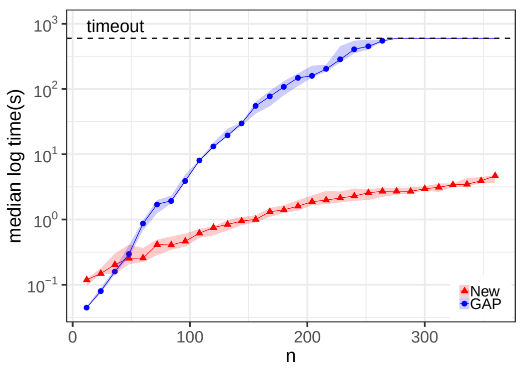
s	NOTESTS	WITHTESTS
1	0.266	0.6405
2	1.7585	0.7815
3	2.664	1.0625
4	8.8435	1.1015
5	15.7735	1.3825
6	22.406	1.563
7	25.7735	1.7815
8	38.9135	2.0475
9	41.633	2.453
10	55.727	2.6485

(b) T = PRIMITIVEGROUP(12,2)

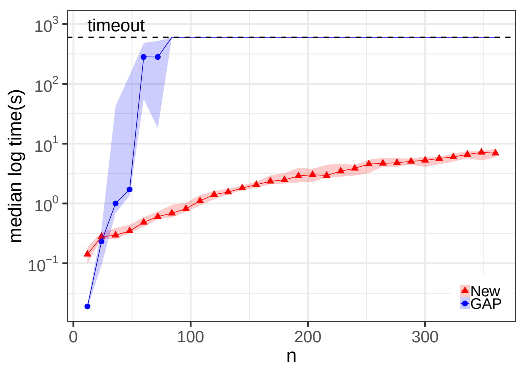
Table 6.2: Median times (in seconds) of computing $N_{S_n}(H)$ of 10 random H in class $\mathfrak{InP}(T)$, where H has 2 disjoint direct factors, each of which has s orbits, using Theorem 6.3.7 (NOTESTS) and Algorithm 11 (WITHTESTS).

6.7.2 Normaliser of $H \in \mathfrak{InP}(S_m)$

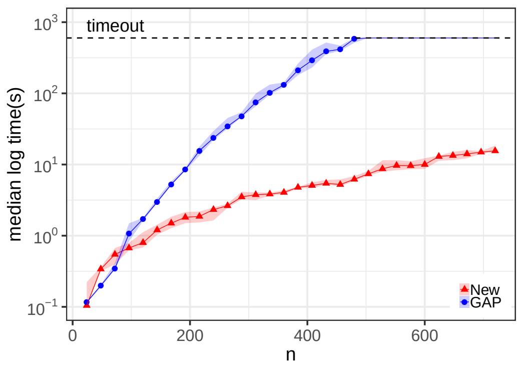
Let $H \in \mathfrak{InP}(S_m)$. Then by Lemma 6.6.5, there exists $A \in \mathfrak{InP}(A_m)$ and $F \in \mathfrak{InP}(C_2)$ such that $H = AF$. Hence we can generate a group H in $\mathfrak{InP}(S_m)$ by generating a



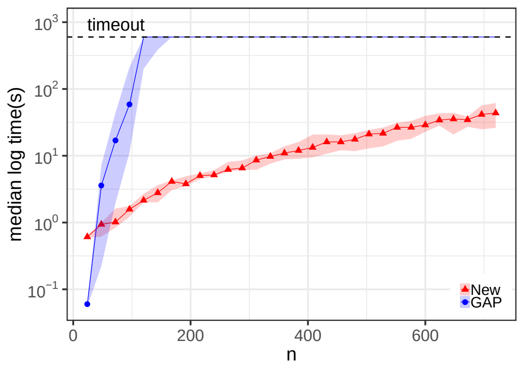
(a) $A = \textsc{PrimitiveGroup}(6, 1)$



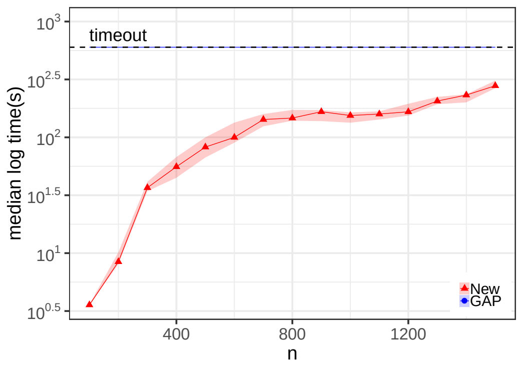
(b) $A = \textsc{PrimitiveGroup}(6, 3)$



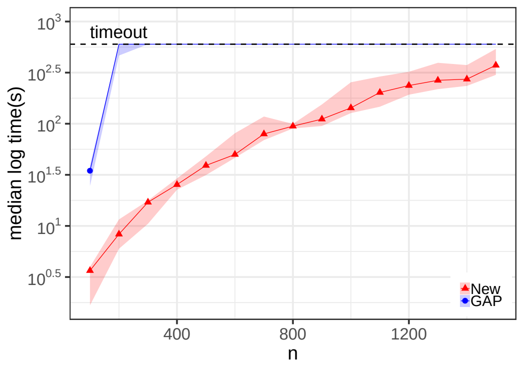
(c) $A = \textsc{PrimitiveGroup}(12, 3)$



(d) $A = \textsc{PrimitiveGroup}(12, 2)$

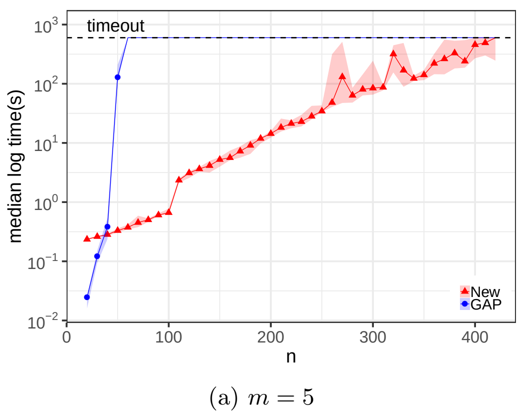


(e) $A = \textsc{PrimitiveGroup}(50, 3)$

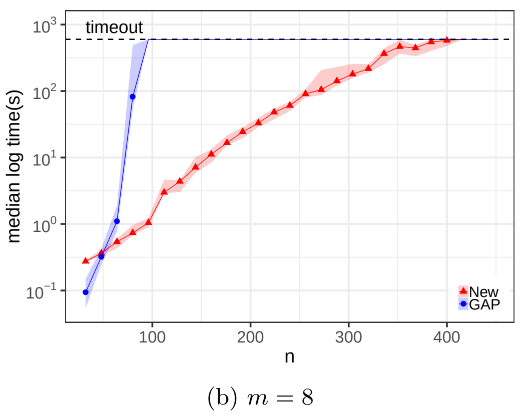


(f) $A = \textsc{PrimitiveGroup}(50, 1)$

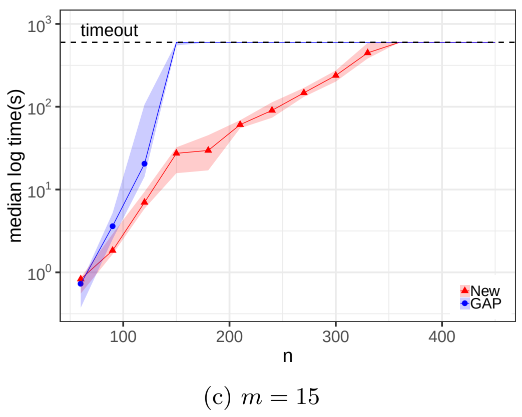
Figure 6.1: Median log time (s) for computing normalisers of 10 random $H \leq S_n$ for $H \in \mathfrak{InP}(A)$ with 10 minutes timeout. The lower and upper boundaries of the shaded area give the lower and upper quartiles respectively.



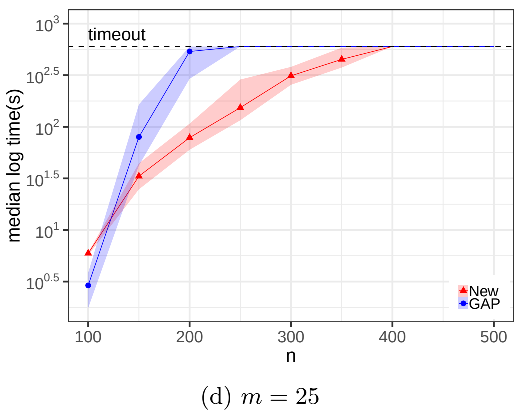
(a) $m = 5$



(b) $m = 8$



(c) $m = 15$



(d) $m = 25$

Figure 6.2: Median log time (s) for computing normalisers of 10 random $H \leq S_n$ for $H \in \mathfrak{InP}(S_m)$ with 10 minutes timeout. The lower and upper boundaries of the shaded area give the lower and upper quartiles respectively.

group A as in Section 6.7.1 and a group F as in Section 5.6.1 and take $H = \langle A, F\rangle$. However, experiments suggest that this will mostly result in $A \cong A_m^k$.

Hence we construct a group $H \in \mathfrak{InP}(S_m)$ the following way. We first construct a random partition $P = \langle C_2 \mid C_2 \mid \dots \mid C_r\rangle$ of $\{1, 2, \dots, k\}$ as in Section 6.7.1. Let s be a random integer between $\lceil r/2\rceil$ and r. Next, we construct a random $s \times r$ matrix M over $\mathbb{F}_2$, as in Section 5.6.1. Then let M' be an $s \times k$ matrix over $\mathbb{F}_2$ such that $M'_{i,j} = 1$ if and only if $j \in C_s$ and $M_{i,s} = 1$. For $1 \leq j \leq k$, let g_j be the transposition permuting $(j-1)m + 1$ and $(j-1)m + 2$. Let $F = \langle \prod_{j=1}^{k} g_j^{M_{i,j}} \mid 1 \leq i \leq s\rangle$.

Next, we generate a finer partition P' from P by constructing partitions of each cell of P, where each random partitions are generated as in Section 6.7.1. Then $A \in \mathfrak{InP}(A_m)$ is constructed from P' as in Section 6.7.1. Finally, take $H = \langle A, F\rangle$.

We consider $H \in \mathfrak{InP}(S_m)$ for $m = 5, 8, 15$ and 25. For each value of m and k, we consider 10 groups $H \leq S_{mk}$ in class $\mathfrak{InP}(S_m)$. For each instance of H, we compute its normalisers using both the GAP function Normalizer and the method described in Section 6.6. We report the lower quartile, median and upper quartile computation time for both of these algorithms in Figure 6.2.

Conclusion

In this thesis, we investigated the Norm-Sym problem of computing the normaliser $N_{S_n}(H)$ of a given intransitive group $H = \langle X \rangle \leq S_n$.

We first introduced disjoint direct product decomposition and showed that the finest such decomposition of a given permutation group can be computed in polynomial time. The disjoint direct product decomposition can be used to convert the Norm-Sym problem into polynomially many smaller Norm-Sym problems and solving the conjugacy problem for polynomially many smaller pairs of groups. We also saw how we can reduce the Norm-Sym problem into a smaller Norm-Sym problem using equivalent orbits.

As the Norm-Sym problem is solved in practice using backtrack search, we presented several pruning methods using the aforementioned equivalent orbits and disjoint direct product decompositions, and also using the permutation isomorphism classes of certain projections. We then shifted our focus to the class $\mathfrak{InP}(A)$ of groups whose transitive constituents are permutation isomorphic to a transitive group $A \leq S_m$, and we reduce the computation of $N_{S_n}(H)$ for $H \in \mathfrak{InP}(A)$ into computing $N_S(H)$ for a proper subgroup group $S \lneqq S_n$, hence reducing the size of the search space. The methods mentioned so far are shown to be able to speed up calculations of $N_{S_n}(H)$ for $H \in \mathfrak{InP}(A)$.

We then considered the case where $H \in \mathfrak{InP}(A)$ and A is simple, which is further divided into two parts based on whether A is abelian. For $H \in \mathfrak{InP}(C_p)$ where p is prime, we show that computing $N_{S_n}(H)$ is polynomial equivalent to computing the monomial automorphism group of codes over $\mathbb{F}_p$. Using this alternative viewpoint, we give an algorithm to compute $N_{S_n}(H)$ with a better worst-case practical complexity that performs efficiently in practice. For $H \in \mathfrak{InP}(T)$ where T is non-abelian simple, we showed that $N_{S_n}(H)$ can be computed in polynomial time, and give an algorithm that also performs well in practice. Finally we used these two algorithms to compute $N_{S_n}(H)$ for $H \in \mathfrak{InP}(D_{2p}) \cup \mathfrak{InP}(S_m)$, where p is an odd prime and $5 \leq m \neq 6$.

For most of the thesis, we have focused on groups with permutation isomorphic transitive constituents. The results from Chapters 5 and 6 may be useful when computing $N_{S_n}(H)$ where there exists Δ such that $H|_\Delta$ is highly intransitive and is in class $\mathfrak{InP}(C_p) \cup \mathfrak{InP}(T)$. However, although we can detect such a set Δ in polynomial time, it is unclear when we should check for it. While we believe that the pruning methods from Chapter 4 may be beneficial in a more general setting, it is also unclear if the benefits outweigh the cost of running such tests and refiners at every node of the search tree. Therefore a future direction is to generalise the pruning methods and integrate them into the state-of-the-art partition backtrack or graph backtrack framework, to directly compare their effectiveness.

Bibliography

[ALSS11] Michael Aschbacher, Richard Lyons, Stephen D. Smith, and Ronald Solomon. *The classification of finite simple groups*, volume 172 of *Mathematical Surveys and Monographs*. American Mathematical Society, Providence, RI, 2011. Groups of characteristic 2 type.

[Art55] Emil Artin. The orders of the classical simple groups. *Comm. Pure Appl. Math.*, 8:455–472, 1955.

[Atk75] Michael D. Atkinson. An algorithm for finding the blocks of a permutation group. *Math. Comput.*, 29:911–913, 1975.

[Bab15] László Babai. Graph isomorphism in quasipolynomial time. *CoRR*, abs/1512.03547, 2015.

[Bab16] László Babai. Graph isomorphism in quasipolynomial time. In *Proc. of the forty-eighth annual ACM symposium on Theory of Computing*, pages 684–697. ACM, 2016.

[BCFS91] László Babai, Gene Cooperman, Larry Finkelstein, and Ákos Seress. Nearly linear time algorithms for permutation groups with a small base. In *Proc. of the 1991 International Symposium on Symbolic and Algebraic Computation*, ISSAC '91, page 200–209, New York, 1991. ACM Press.

[BCGQ11] László Babai, Paolo Codenotti, Joshua A. Grochow, and Youming Qiao. Code equivalence and group isomorphism. In *Proc. of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1395–1408. SIAM, Philadelphia, PA, 2011.

[Bes06] Christian Bessiere. Chapter 3 - constraint propagation. In Francesca Rossi, Peter [van Beek], and Toby Walsh, editors, *Handbook of Constraint Programming*, volume 2 of *Foundations of Artificial Intelligence*, pages 29 – 83. Elsevier, 2006.

[BF73] Raymond A. Beauregard and John B. Fraleigh. *A first course in linear algebra.* Houghton Mifflin Co., Boston, Mass., 1973. With optional introduction to groups, rings, and fields.

[BKL83] László Babai, William M. Kantor, and Eugene M. Luks. Computational complexity and the classification of finite simple groups. *24th Annual Symposium on Foundations of Computer Science (sfcs 1983)*, pages 162–171, 1983.

[BSZ15] Kristine Bauer, Debasis Sen, and Peter Zvengrowski. A generalized Goursat lemma. *Tatra Mt. Math. Publ.*, 64:1–19, 2015.

[But83] Gregory Butler. Computing normalizers in permutation groups. *J. Algorithms*, 4(2):163–175, 1983.

[Cam99] Peter J. Cameron. *Permutation Groups.* London Mathematical Society Student Texts. Cambridge University Press, 1999.

[Car72] Roger W. Carter. *Simple groups of Lie type.* John Wiley & Sons, London-New York-Sydney, 1972. Pure and Applied Mathematics, Vol. 28.

[CF94] Gene Cooperman and Larry Finkelstein. A random base change algorithm for permutation groups. *J. Symbolic Comput.*, 17(6):513–528, 1994.

[CFL89] Gene Cooperman, Larry Finkelstein, and Eugene M. Luks. Reduction of group constructions to point stabilizers. In *Proc. of the ACM-SIGSAM 1989 International Symposium on Symbolic and Algebraic Computation*, ISSAC '89, page 351–356, New York, 1989. ACM Press.

[Cha21] Mun See Chang. Computing normalisers of highly intransitive groups (thesis data). Dataset, University of St Andrews Research Portal, 2021. `https://doi.org/10.17630/710dfd8d-356b-4080-b2ad-c6791b7c21fe`.

[Dem] Sophie Demassey. Global constraint catalog. `https://sofdem.github.io/gccat/`. Accessed: 03 Sep 2020.

[DM96] John D. Dixon and Brian Mortimer. *Permutation groups*, volume 163 of *Graduate Texts in Mathematics.* Springer-Verlag, New York, 1996.

[DM06] Alastair F. Donaldson and Alice Miller. Exact and approximate strategies for symmetry reduction in model checking. *FM 2006: Formal Methods*, pages 541–556, 2006.

[DM09] Alastair F. Donaldson and Alice Miller. On the constructive orbit problem. *Ann. Math. Artif. Intell.*, 57(1):1–35, 2009.

[Feu09] Thomas Feulner. The automorphism groups of linear codes and canonical representatives of their semilinear isometry classes. *Adv. Math. Commun.*, 3(4):363–383, 2009.

[GAP20] The GAP Group. *GAP – Groups, Algorithms, and Programming, Version 4.11.0*, 2020.

[GJMRD09] Andrew Grayland, Chris Jefferson, Ian Miguel, and Colva M. Roney-Dougal. Minimal ordering constraints for some families of variable symmetries. *Ann. Math. Artif. Intell.*, 57(1):75–102, 2009.

[GMN08] Ian P. Gent, Ian Miguel, and Peter Nightingale. Generalised arc consistency for the AllDifferent constraint: an empirical survey. *Artificial Intelligence*, 172(18):1973–2000, 2008.

[GMP17] Robert M. Guralnick, Attila Maróti, and László Pyber. Normalizers of primitive permutation groups. *Adv. Math.*, 310:1017–1063, 2017.

[Gor82] Daniel Gorenstein. *Finite simple groups.* University Series in Mathematics. Plenum Publishing Corp., New York, 1982. An introduction to their classification.

[Gor83] Daniel Gorenstein. *The classification of finite simple groups. Vol. 1.* The University Series in Mathematics. Plenum Press, New York, 1983. Groups of noncharacteristic 2 type.

[Gou89] Édouard Goursat. Sur les substitutions orthogonales et les divisions régulières de l'espace. *Annales scientifiques de l'École Normale Supérieure*, 3e série, 6:9–102, 1889.

[GPP06] Ian P. Gent, Karen E. Petrie, and Jean-François Puget. Chapter 10 - symmetry in constraint programming. In Francesca Rossi, Peter van Beek, and Toby Walsh, editors, *Handbook of Constraint Programming*, volume 2 of *Foundations of Artificial Intelligence*, pages 329 – 376. Elsevier, 2006.

[Gra11] Andrews Grayland. *Automated static symmetry breaking in constraint satisfaction problems.* PhD thesis, University of St Andrews, 2011.

[Hol91] D. F. Holt. The computation of normalizers in permutation groups. *J. Symbolic Comput.*, 12(4-5):499–516, 1991.

[Hul05] Alexander Hulpke. Constructing transitive permutation groups. *J. Symbolic Comput.*, 39(1):1–30, 2005.

[Hul08] Alexander Hulpke. Normalizer calculation using automorphisms. In *Computational group theory and the theory of groups*, volume 470 of *Contemp. Math.*, pages 105–114. Amer. Math. Soc., Providence, RI, 2008.

[Hul17] Alexander Hulpke. TransGrp, transitive groups library, Version 2.0.2. `http://www.math.colostate.edu/~hulpke/transgrp`, Nov 2017. GAP package.

[Hun74] Thomas W. Hungerford. *Algebra.* Holt, Rinehart and Winston, Inc., New York-Montreal, Que.-London, 1974.

[JPW19] Christopher Jefferson, Markus Pfeiffer, and Rebecca Waldecker. New refiners for permutation group search. *J. Symbolic Comput.*, 92:70–92, 2019.

[JPWW19] Christopher Jefferson, Markus Pfeiffer, Rebecca Waldecker, and Wilf A. Wilson. Permutation group algorithms based on directed graphs. *arXiv e-prints*, page arXiv:1911.04783, November 2019.

[Kan85] William M. Kantor. Sylow's theorem in polynomial time. *J. Comput. System Sci.*, 30(3):359–394, 1985.

[Kan90] William M. Kantor. Finding Sylow normalizers in polynomial time. *J. Algorithms*, 11(4):523–563, 1990.

[KL90] William M. Kantor and Eugene M. Luks. Computing in quotient groups. In *Proc. of the Twenty-Second Annual ACM Symposium on Theory of Computing*, STOC '90, page 524–534, New York, 1990. ACM Press.

[KN09] Neeraj Kayal and Timur Nezhmetdinov. Factoring groups efficiently. In *Automata, languages and programming. Part I*, volume 5555 of *Lecture Notes in Comput. Sci.*, pages 585–596. Springer, Berlin, 2009.

[Leo91] Jeffrey S. Leon. Permutation group algorithms based on partitions. I. Theory and algorithms. *J. Symbolic Comput.*, 12(4-5):533–583, 1991.

[LM02] Eugene M. Luks and Takunari Miyazaki. Polynomial-time normalizers for permutation groups with restricted composition factors. In *Proc. of the 2002 International Symposium on Symbolic and Algebraic Computation*, pages 176–183. ACM, New York, 2002.

[LM11] Eugene M. Luks and Takunari Miyazaki. Polynomial-time normalizers. *Discrete Mathematics and Theoretical Computer Science*, Vol. 13 no. 4(4):61–96, December 2011.

[LRW94] Eugene M. Luks, Ferenc Rákóczi, and Charles R. B. Wright. Computing normalizers in permutation p-groups. In *Proc. of the International Symposium on Symbolic and Algebraic Computation*, ISSAC '94, page 139–146, New York, 1994. ACM Press.

[Luk93] Eugene M. Luks. Permutation groups and polynomial-time computation. In *Groups and computation (New Brunswick, NJ, 1991)*, volume 11 of *DIMACS Ser. Discrete Math. Theoret. Comput. Sci.*, pages 139–175. Amer. Math. Soc., Providence, RI, 1993.

[Lyo11] Richard Lyons. Automorphism groups of sporadic groups. *arXiv e-prints*, page arXiv:1106.3760, June 2011.

[McC14] Jon McCammond. The exceptional symmetry. *arXiv e-prints*, page arXiv:1412.1855, December 2014.

[Miy06] Izumi Miyamoto. An improvement of GAP normalizer function for permutation groups. In *ISSAC 2006*, pages 234–238. ACM, New York, 2006.

[MP14] Brendan D. McKay and Adolfo Piperno. Practical graph isomorphism, II. *J. Symbolic Comput.*, 60:94–112, 2014.

[PJ08] Karen E. Petrie and Christopher Jefferson. Efficiently solving problems where the solutions form a group. In *Proc. of the 14th International Conference on Principles and Practice of Constraint Programming*, CP '08, page 529–533, Berlin, Heidelberg, 2008. Springer-Verlag.

[PR97] Erez Petrank and Ron M. Roth. Is code equivalence easy to decide? *IEEE Trans. Inform. Theory*, 43(5):1602–1604, 1997.

[PS18] Cheryl E. Praeger and Csaba Schneider. *Permutation groups and Cartesian decompositions*, volume 449 of *London Mathematical Society Lecture Note Series*. Cambridge University Press, Cambridge, 2018.

[RDS20] Colva M. Roney-Dougal and Sergio Siccha. Normalisers of primitive permutation groups in quasipolynomial time. *Bulletin of the London Mathematical Society*, 52(2):358–366, April 2020.

[Sch94] Roland Schmidt. *Subgroup lattices of groups*, volume 14 of *De Gruyter Expositions in Mathematics*. Walter de Gruyter & Co., Berlin, 1994.

[Ser03] Ákos Seress. *Permutation group algorithms*, volume 152 of *Cambridge Tracts in Mathematics*. Cambridge University Press, Cambridge, 2003.

[Sic20] Sergio Siccha. Towards efficient normalizers of primitive groups. In Anna Maria Bigatti, Jacques Carette, James H. Davenport, Michael Joswig, and Timo de Wolff, editors, *Mathematical Software – ICMS 2020*, pages 105–114, Cham, 2020. Springer International Publishing.

[SS13] Nicolas Sendrier and Dimitris E. Simos. The hardness of code equivalence over $\mathbb{F}_q$ and its application to code-based cryptography. In Philippe Gaborit, editor, *Post-Quantum Cryptography*, pages 203–216. Springer Berlin Heidelberg, 2013.

[The97] Heiko Theißen. *Eine Methode zur Normalisatorberechnung in Permutationsgruppen mit Anwendungen in der Konstruktion primitiver Gruppen.* PhD thesis, RWTH Aachen, 1997.

[TvL84] Robert E. Tarjan and Jan van Leeuwen. Worst-case analysis of set union algorithms. *J. Assoc. Comput. Mach.*, 31(2):245–281, 1984.

[vL99] J. H. van Lint. *Introduction to coding theory*, volume 86 of *Graduate Texts in Mathematics*. Springer-Verlag, Berlin, third edition, 1999.

[Wie19] Daniel Wiebking. Normalizers and permutational isomorphisms in simply-exponential time. *CoRR*, abs/1904.10454, 2019.

[Wil12] James B. Wilson. Existence, algorithms, and asymptotics of direct product decompositions, I. *Groups Complex. Cryptol.*, 4(1):33–72, 2012.