Towards Efficient Normalizers of Primitive Groups

Sergio Siccha(✉)

Department of Mathematics, University of Kaiserslautern, Kaiserslautern, Germany
siccha@mathematik.uni-kl.de

Abstract. We present the ideas behind an algorithm to compute normalizers of primitive groups with non-regular socle in polynomial time. We highlight a concept we developed called permutation morphisms and present timings for a partial implementation of our algorithm. This article is a collection of results from the author's PhD thesis.

Keywords: Normalizers · Primitive groups · Permutation group algorithms

1 Introduction

One of the tools to study the internal structure of groups is the normalizer. For two groups G and H, which are contained in a common overgroup K, we call the *normalizer of* G *in* H, denoted $N_H(G)$, the subgroup of H consisting of those elements that leave G invariant under conjugation.

We only consider finite sets, finite groups, and permutation groups acting on finite sets. We assume permutation groups to always be given by generating sets and say that a problem for permutation groups can be solved in *polynomial time*, if there exists an algorithm which, given permutation groups of degree n, solves it in time bounded polynomially in n and in the sizes of the given generating sets. While many problems for permutation groups can be solved efficiently both in theory and in practice, no polynomial time algorithm to compute normalizers of permutation groups is known.

A transitive permutation group G acting on a set Ω is called *primitive* if there exists no non-trivial G-invariant partition of Ω. Primitive groups have a rich and well-understood structure. Hence many algorithms use the natural recursion from general permutation groups to transitive and in turn to primitive ones. For two permutation groups $G, H \leq \operatorname{Sym}\Omega$ computing the normalizer of G in H in general is done by searching for the normalizer of G in the symmetric group $\operatorname{Sym}\Omega$ and simultaneously computing the intersection with H. We focus on computing the normalizer of a primitive group $G \leq \operatorname{Sym}\Omega$ in $\operatorname{Sym}\Omega$. Being able to compute normalizers for primitive groups efficiently may lead to improved algorithms for more general situations.

Our results build substantially on the O'Nan-Scott classification of primitive groups, see [17], and on the classification of finite simple groups (CFSG).

Recall that the *socle* of a group G, denoted $\operatorname{Soc} G$, is the subgroup generated by all minimal normal subgroups of G. Our theoretical main result is the following theorem.

Theorem 1 **([23, Theorem 9.1]).** *Let a primitive group $G \leq \operatorname{Sym} \Omega$ with non-regular socle*[1] *be given. Then we can compute $N_{\operatorname{Sym} \Omega}(G)$ in polynomial time.*

As is often the case in computational group theory, ideas from theoretical algorithms can be employed in practical algorithms and vice versa. While the algorithms in [23] are primarily theoretical ones, we also provide probabilistic nearly-linear time versions where possible. The author is developing the GAP package `NormalizersOfPrimitiveGroups`, hosted at https://github.com/ssiccha/NormalizersOfPrimitiveGroups[2], with the aim to implement practical versions of the algorithms developed in [23]. Until now, algorithms concerning permutation morphisms and primitive groups of type PA are implemented. First experiments indicate that already for moderate degrees these outperform the GAP built-in algorithm `Normalizer` by several orders of magnitude, see Table 1.

Since no polynomial time solutions are known for the normalizer problem, the generic practical algorithms resolve to backtracking over the involved groups in one way or another. The fundamental framework of modern backtrack algorithms for permutation groups is Leon's partition backtrack algorithm [16], which generalizes previous backtrack approaches [5,6,12,24] and generalizes ideas of *nauty* [19] to the permutation group setting. Partition backtrack is implemented in GAP [9] and Magma [4]. Recently, the partition backtrack approach was generalized to a "graph backtrack" framework [14].

Theißen developed a normalizer algorithm which uses orbital graphs to prune the backtrack search [25]. Chang is currently developing specialized algorithms for highly intransitive permutation groups, her PhD thesis should appear shortly. It is to expect that the work in [14] can also be extended to normalizer problems. Hulpke also implemented normalizer algorithms in [13] using group automorphisms and the GAP function `NormalizerViaRadical` based on [10].

In Sect. 2 we outline the strategy behind our algorithms. In Sect. 3 we recall the O'Nan-Scott Theorem. We present our new concept of permutation morphisms in Sect. 4. In Sect. 5 we sketch how we use our results to obtain Theorem 1. In Sect. 6 we discuss our implementation.

2 Strategy

We describe the strategy of the theoretical algorithm behind Theorem 1. Comments regarding the implementation of its building blocks are given at the end of each following section.

In this section let $G \leq \operatorname{Sym} \Omega$ be a primitive group with non-regular socle H. The normalizer of H in $\operatorname{Sym} \Omega$ plays a central role in our algorithm, in this

[1] This excludes groups of affine and of twisted wreath type.
[2] May move to https://github.com/gap-packages/NormalizersOfPrimitiveGroups.

section we denote it by M. Observe that to compute $N_{\mathrm{Sym}\,\Omega}(G)$ it suffices to compute $N_M(G)$ since the former is contained in M.

The socle H is isomorphic to T^ℓ for some finite non-abelian simple group T and some positive integer ℓ. The group G is isomorphic to a subgroup of the wreath product $\mathrm{Aut}(T) \wr S_\ell$, see Sect. 3 for a definition of wreath products. By the O'Nan-Scott Theorem the respective isomorphism extends to an embedding[3] of the normalizer M into $\mathrm{Aut}(T) \wr S_\ell$. Furthermore ℓ is of the order $O(\log|\Omega|)$. Hence the index of G in M, and thus also the search-space of the normalizer computation $N_M(G)$, is tiny in comparison to the index of G in $\mathrm{Sym}\,\Omega$.

Our approach can be divided into two phases. First we compute M, this is by far the most labor intensive part. To this end we compute a sufficiently well-behaved conjugate of G, such that we can exhibit the wreath structure mentioned above. In [23] we make this more precise and define a *weak canonical form* for primitive groups. Using that conjugate and the O'Nan-Scott Theorem we can write down generators for M. In the second phase, we compute a reduction homomorphism $\rho : M \to S_k$ with $k \leq 6\log|\Omega|$. After this logarithmic reduction, we use Daniel Wiebking's simply exponential time algorithm [26,27], which is based on the canonization framework [22], to compute $N_{S_k}(\rho(G))$. Note that the running time of a simply exponential time algorithm called on a problem of size $\log n$ is $2^{O(\log n)}$ and thus is bounded by $2^{c \log n} = n^c$ for some constant $c > 0$. Then we use Babai's famous quasipolynomial time algorithm for graph-isomorphism [1,2] to compute the group intersection $N_{\rho(M)}(\rho(G)) = \rho(M) \cap N_{S_k}(\rho(G))$. Notice that since we perform these algorithms on at most $6\log n$ points they run in time polynomial in n. The homomorphism ρ is constructed in such a way, that computing the preimage of the above normalizer $N_{\rho(M)}(\rho(G))$ yields $N_M(G)$. Recall that $N_M(G)$ is equal to $N_{\mathrm{Sym}\,\Omega}(G)$.

In our implementation we do not use the algorithms by Wiebking and Babai since these are purely theoretical. Instead we use the partition backtrack implemented in GAP.

3 The O'Nan-Scott Theorem

The goal of this and the next section is to illustrate how we use the O'Nan-Scott Theorem to prove the following theorem. In this article we limit ourselves to groups of type PA, which we define shortly.

Theorem 2 ([23, Theorem 8.1]). *Let a primitive group $G \leq \mathrm{Sym}\,\Omega$ with non-abelian socle be given. Then we can compute $N_{\mathrm{Sym}\,\Omega}(\mathrm{Soc}\,G)$ in polynomial time.*

Proof. For groups of type PA this will follow from Corollary 5 and Lemma 6.

The O'Nan-Scott Theorem classifies how the socles of primitive groups can act, classifies the normalizers of the socles, and determines criteria to decide which subgroups of these normalizers act primitively. We follow the division of

[3] For twisted wreath type the situation is slightly more complicated.

primitive groups into eight O'Nan-Scott types as it was suggested by László G. Kovács and first defined by Cheryl Praeger in [21]. In this section we define the types AS and PA and recall some of their basic properties. In particular we describe the normalizer of the socle for groups of type PA and how to construct the normalizer of the socle, if the group is given in a sufficiently well-behaved form.

The version of the O'Nan-Scott Theorem we use, for a proof see [17], is:

Theorem 3. *Let G be a primitive group on a set Ω. Then G is a group of type HA, AS, PA, HS, HC, SD, CD, or TW.*

The abbreviation AS stands for **A**lmost **S**imple. A group is called *almost simple* if it contains a non-abelian simple group and can be embedded into the automorphism group of said simple group. A primitive group G is of *AS type* if its socle is a non-regular non-abelian simple group.

The abbreviation PA stands for **P**roduct **A**ction. The groups of AS type form the building blocks for the groups of PA type. To define this type, we shortly recall the notion of wreath products and their product action.

The *wreath product* of two permutation groups $H \leq \operatorname{Sym} \Delta$ and $K \leq S_d$ is denoted by $H \wr K$ and defined as the semidirect product $H^d \rtimes K$ where K acts per conjugation on H^d by permuting its components. We identify H^d and K with the corresponding subgroups of $H \wr K$ and call them the *base group* and the *top group*, respectively.

For two permutation groups $H \leq \operatorname{Sym} \Delta$ and $K \leq S_d$ the *product action* of the wreath product $H \wr K$ on the set of tuples Δ^d is given by letting the base group act component-wise on Δ^d and letting the top group act by permuting the components of Δ^d.

Definition 4. *Let $G \leq \operatorname{Sym} \Omega$ be a primitive group. We say that G is of* type PA *if there exist an $\ell \geq 2$ and a primitive group $H \leq \operatorname{Sym} \Delta$ of type AS such that G is permutation isomorphic to a group $\widehat{G} \leq \operatorname{Sym} \Delta^\ell$ with*

$$(\operatorname{Soc} H)^\ell \leq \widehat{G} \leq H \wr S_\ell$$

in product action on Δ^ℓ.

The product action wreath products $A_5 \wr \langle(1,2,3)\rangle$ and $A_5 \wr \langle(1,2)\rangle$ are examples for primitive groups of type PA.

Let $\widehat{G} \leq \operatorname{Sym}(\Delta^\ell)$ and $H \leq \operatorname{Sym} \Delta$ be as in Definition 4. We sketch how to construct the normalizer of the socle of $\widehat{G}$. Let $T := \operatorname{Soc} H \leq \operatorname{Sym} \Delta$. Since $\widehat{G}$ is given acting in product action we can read off H and thus compute T. By [8, Lemma 4.5A] we know that the normalizer of $\operatorname{Soc} \widehat{G}$ in $\operatorname{Sym} \Delta^\ell$ is $N_{\operatorname{Sym} \Delta}(T) \wr S_\ell$. By recent work of Luks and Miyazaki we can compute the normalizer of T, in polynomial time [18, Corollary 3.24]. More precisely this approach yields the following corollary:

Corollary 5. *Let $G \leq \operatorname{Sym}(\Delta^\ell)$ be a primitive group of type PA with socle T^ℓ in component-wise action on Δ^ℓ. Then $N_{\operatorname{Sym}(\Delta^\ell)}(T^\ell)$ can be computed in polynomial time.*

In the practical implementation we use the GAP built-in algorithm to compute the normalizer of T in $\operatorname{Sym}\Delta$. Our long-term goal is to use the constructive recognition provided by the `recog` package [20]. Computing the normalizer of T in $\operatorname{Sym}\Delta$ is then only a matter of iterating through representatives for the outer automorphisms of T.

4 Permutation Morphisms

In general a group of PA type might be given on an arbitrary set and needs only be permutation isomorphic to a group in product action. In this section we discuss how to construct such a permutation isomorphism:

Lemma 6. *Let $G \leq \operatorname{Sym}\Omega$ be a primitive group of type PA. Then we can compute a non-abelian simple group $T \leq \operatorname{Sym}\Delta$, a positive integer ℓ, and a permutation isomorphism from G to a permutation group $\widehat{G} \leq \operatorname{Sym}(\Delta^\ell)$ such that the socle of $\widehat{G}$ is T^ℓ in component-wise action on Δ^ℓ.*

To this end we present the notion of *permutation morphisms* developed in [23]. They arise from permutation isomorphisms by simply dropping the condition that the domain map and the group homomorphism be bijections. We illustrate how to use them to prove Lemma 6.

4.1 Basic Definitions

For two maps $f : A \to B$ and $g : C \to D$ we denote by $f \times g$ the product map $A \times C \to B \times D,\ (a, c) \mapsto (f(a), g(c))$. For a right-action $\rho : \Omega \times G \to \Omega$ of a group G and $g \in G$, $\omega \in \Omega$ we also denote $\rho(\omega, g)$ by ω^g.

Definition 7. *Let G and H be permutation groups on sets Ω and Δ, respectively, let $f : \Omega \to \Delta$ be a map, and let $\varphi : G \to H$ be a group homomorphism. Furthermore let ρ and τ be the natural actions of G and H on Ω and Δ, respectively. We call the pair (f, φ) a* permutation morphism from G to H *if the following diagram commutes:*

$$\begin{tikzcd}
\Omega \times G \arrow[r, "\rho"] \arrow[d, "f\times\varphi"] & \Omega \arrow[d, "f"] \\
\Delta \times H \arrow[r, "\tau"] & \Delta
\end{tikzcd},$$

that is if $f(\omega^g) = f(\omega)^{\varphi(g)}$ holds for all $\omega \in \Omega$, $g \in G$. We call φ the group homomorphism of (f, φ) *and f the* domain map of (f, φ).

It is immediate from the definition, that the component-wise composition of two permutation morphisms again yields a permutation morphism. In particular we define the *category of permutation groups* as the category with all permutation groups as objects, all permutation morphisms as morphisms, and the componentwise composition as the composition of permutation morphisms. We rely on this categorical perspective in many of our proofs.

We denote a permutation morphism F from a permutation group G to a permutation group H by $F : G \to H$. When encountering this notation keep in mind that F itself is not a map but a pair of a domain map and a group homomorphism. We use capital letters for permutation morphisms.

It turns out that a permutation morphism F is a mono-, epi-, or isomorphism in the categorical sense if and only if both its domain map and group homomorphism are injective, surjective, or bijective, respectively.

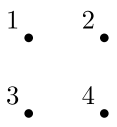
For a permutation group $G \leq \operatorname{Sym} \Omega$ we call a map $f : \Omega \to \Delta$ *compatible with* G if there exists a group homomorphism φ such that $F = (f, \varphi)$ is a permutation morphism. We say that a partition Σ of Ω is G-invariant if for all $A \in \Sigma$ and $g \in G$ we have $A^g \in \Sigma$.

Lemma 8 ([23], Lemma 4.2.10]). *Let $G \leq \operatorname{Sym} \Omega$ be a permutation group and $f : \Omega \to \Delta$ a map. Then f is compatible with G if and only if the partition of Ω into the non-empty fibers $\{f^{-1}(\{\delta\}) \mid \delta \in f(\Omega)\}$ is G-invariant.*

If G is transitive, then the G-invariant partitions of Ω are precisely the block systems of G. Hence for a given blocksystem we can define a compatible map f by sending each point to the block it is contained in.

Let $G \leq \operatorname{Sym} \Omega$ be a permutation group and $f : \Omega \to \Delta$ a surjective map compatible with G. Then there exist a unique group $H \leq \operatorname{Sym} \Delta$ and a unique group homomorphism $\varphi : G \to H$ such that $F := (f, \varphi)$ is a permutation epimorphism, see [23, Corollary 4.2.7]. We call F the *permutation epimorphism* and φ the *group epimorphism of G induced by f*.

Example 9. Let $\Omega = \{1, \dots, 4\}$, $a := (1,2)(3,4)$, $b := (1,3)(2,4)$, and $V := \langle a, b \rangle$. Further consider the set $\Omega_1 := \{1, 2\}$, the map $p_1 : \Omega \to \Omega_1,\ 1,3 \mapsto 1,\ 2,4 \mapsto 2$, and the following geometric arrangement of the points $1, \dots, 4$:



Observe that a acts on Ω by permuting the points horizontally, while b acts on Ω by permuting the points vertically. The map p_1 projects Ω vertically or "to the top". Notice how the fibers of p_1 correspond to a block-system of V. We determine the group epimorphism π_1 of V induced by p_1. By definition $\pi_1(a)$ is the permutation which makes the following square commute:

$$\begin{tikzcd}
\Omega \arrow[r, "a"] \arrow[d, "p_1"] & \Omega \arrow[d, "p_1"] \\
\Omega_1 \arrow[r, "\pi_1(a)"] & \Omega_1
\end{tikzcd}$$

Take $1 \in \Omega_1$. We have $p_1^{-1}(\{1\}) = \{1,3\}$, $a(\{1,3\}) = \{2,4\}$, and $p_1(\{2,4\}) = \{2\}$. Hence $\pi_1(a) = (1,2)$. Correspondingly we get $\pi_1(b) = \mathrm{id}_{\Omega_1}$.

4.2 Products of Permutation Morphisms

For two permutation groups $H \leq \operatorname{Sym}\Delta$ and $K \leq \operatorname{Sym}\Gamma$ we define the *product of the permutation groups H and K* as the permutation group given by $H \times K$ in component-wise action on $\Delta \times \Gamma$. Correspondingly, for an additional permutation group $G \leq \operatorname{Sym}\Omega$ and two permutation morphisms (f, φ) and (g, ψ) from G to H and K, respectively, we define the *product permutation morphism $G \to H \times K$* as $(f \times g, \varphi \times \psi)$. Iteratively, we define the product of several permutation groups or permutation morphisms.

To prove Lemma 6 it suffices to be able to compute the following: given the socle $H \leq \operatorname{Sym}\Omega$ of a PA type group compute a non-abelian simple group $T \leq \operatorname{Sym}\Delta$ and permutation epimorphisms, think projections, $P_1, \ldots, P_\ell : H \to T$ such that the product morphism $P : H \to T^\ell$ is an isomorphism. Since every surjective map compatible with H induces a unique permutation epimorphism, it in turn suffices to compute suitable maps $p_i : \Omega \to \Delta$.

Example 10. Consider the situation in Example 9. Let $P_1 := (p_1, \pi_1)$ and $\Omega_2 := \{1,3\}$. Then the map $p_2 : \Omega \to \Omega_2,\ 1,2 \mapsto 1,\ 3,4 \mapsto 3$ is compatible with V and induces the permutation epimorphism $P_2 : V \to \langle(1,3)\rangle$. The product maps $p_1 \times p_2 : \Omega \to \Omega_1 \times \Omega_2$ and $P_1 \times P_2 : V \to \langle(1,2)\rangle \times \langle(1,3)\rangle$ are isomorphisms of sets and permutation groups, respectively.

We illustrate how to construct one of the needed projections for PA type groups.

Example 11. Let $\Delta = \{1, \ldots, 5\}$ and $H := A_5 \times A_5 \leq \operatorname{Sym}(\Delta^2)$. We denote by $\mathbf{1}_\Delta$ the trivial permutation group on Δ. The subgroup $H_1 := A_5 \times \mathbf{1}_\Delta \leq \operatorname{Sym}(\Delta^2)$ is normal in H. Let us denote sets of the form $\{(\delta, x_2) \mid \delta \in \Delta\}$ by $\{(*, x_2)\}$. Then partitioning Δ^2 into orbits under H_1 yields the block system

$$\Sigma = \{\{(*, \delta_2)\} \mid \delta_2 \in \Delta\}.$$

Note how mapping each $x \in \Delta^2$ to the block of Σ it is contained in is equivalent to mapping each x to x_2. Thus we have essentially constructed the map $p_2 : \Delta^2 \to \Delta,\ x \mapsto x_2$. Observe that we only used the group theoretic property that H_1 is a maximal normal subgroup of H and thus in particular did not use the actual product structure of Δ^2.

Analogously we can construct the map $p_1 : \Delta^2 \to \Delta,\ x \mapsto x_1$. For $i = 1,2$ let $P_i : H \to A_5$ be the permutation epimorphisms of H induced by p_1 and p_2, respectively. Since $p_1 \times p_2$ is an isomorphism, $P_1 \times P_2$ must be a monomorphism. By order arguments $P_1 \times P_2$ is thus an isomorphism.

In general the above construction does not yield permutation epimorphisms with identical images. We can alleviate this by computing elements of the given group which conjugate the minimal normal subgroups of its socle to each other. For the general construction see the *(homogenized) product decomposition by minimal normal subgroups* in [23, Definitions 5.1.3 and 5.1.5]. Lemma 6 then follows from [23, Corollary 5.19].

5 Reduction Homomorphism

Recall from Sect. 2 that a key ingredient of our second phase is a group homomorphism which reduces the original problem on n points to a problem on less or equal than $6 \log n$ points. We illustrate shortly how to construct this homomorphism, for the details refer to [23, Theorem 9.1.6].

Let $G \leq \operatorname{Sym} \Omega$ be a primitive group with non-regular socle and $T \leq \operatorname{Sym} \Delta$ be a *socle-component of* G, confer [23, Chapters 5 and 7] for a definition. Then T is a non-abelian simple group, there exists a positive integer ℓ such that $\operatorname{Soc} G$ is isomorphic to T^ℓ, and by [23, Lemma 2.6.1] we have $|\Omega| = |\Delta|^s$ for some $s \in \{\ell/2, \dots, \ell\}$. Denote by R the permutation group induced by the right-regular action of $\operatorname{Out} T$ on itself. We show that we can evaluate the following two group homomorphisms: first an embedding $N_{\operatorname{Sym} \Omega}(\operatorname{Soc} G) \to \operatorname{Aut} T \wr S_\ell$ and second an epimorphism $\operatorname{Aut} T \wr S_\ell \to R \wr S_\ell$, where $R \wr S_\ell$ is the imprimitive wreath product and thus acts on $|R| \cdot \ell$ points.

We sketch the proof that $|R| \cdot \ell \leq 6 \log n$. Let $m := |\Delta|$ and $r := |R|$. Note that for ℓ we have $\ell \leq 2s = 2 \log_m n$. Since R is regular, we have $r = |\operatorname{Out} T|$. Since T is a socle-component of G, we have $|\operatorname{Out} T| \leq 3 \log m$ by [11, Lemma 7.7]. In total we have $r \cdot \ell \leq 3 \log m \cdot 2 \log_m n = 6 \log n$.

In our implementation we use a modified version of this reduction. For groups of type PA we can directly compute an isomorphism from the product action wreath product into the corresponding imprimitive wreath product.

6 Implementation

A version of our normalizer algorithm for groups of type PA is implemented in the GAP package `NormalizersOfPrimitiveGroups`.

Table 1 shows a comparison of runtimes of our algorithm and the GAP function `Normalizer`. At the time of writing, there are two big bottlenecks in the implementation. First, the GAP built-in algorithm to compute the socle of a group is unnecessarily slow. State-of-the-art algorithms as in [7] are not yet implemented. Secondly, computing a permutation which transforms a given product decomposition into a so-called natural product decomposition currently also is slow. The latter may be alleviated by implementing the corresponding routines in for example C [15] or Julia [3]. Note that the actual normalizer computation inside the normalizer of the socle appears to be no bottleneck: in the example with socle type $(A_5)^7$ it took only 40 ms!

Table 1. Table with runtime comparison.

Socle type	Degree	Our algorithm	GAP built-in alg.
$(A_5)^2$	25	24 ms	200 ms
$(A_5)^3$	125	50 ms	1500 ms
$(A_5)^4$	625	300 ms	29400 ms
$(A_5)^7$	78125	67248 ms	–
$PSL(2,5)^2$	36	40 ms	300 ms
$PSL(2,5)^3$	216	90 ms	1900 ms
$PSL(2,5)^4$	1296	400 ms	64000 ms
$(A_7)^2$	49	38 ms	900 ms
$(A_7)^3$	343	200 ms	16800 ms
$(A_7)^4$	2401	1400 ms	839000 ms

Acknowledgments. Substantial parts of the work presented in this article were written while the author was supported by the German Research Foundation (DFG) research training group "Experimental and constructive algebra" (GRK 1632) and employed by the Lehr- und Forschungsgebiet Algebra, RWTH Aachen University and the Department of Mathematics, University of Siegen.

References

1. Babai, L.: Graph isomorphism in quasipolynomial time. arXiv e-prints arXiv:1512.03547, December 2015
2. Babai, L.: Graph isomorphism in quasipolynomial time [extended abstract]. In: Proceedings of the Forty-Eighth Annual ACM Symposium on Theory of Computing, STOC 2016, pp. 684–697. ACM (2016). https://doi.org/10.1145/2897518.2897542
3. Bezanson, J., Edelman, A., Karpinski, S., Shah, V.B.: Julia: a fresh approach to numerical computing. SIAM Rev. **59**(1), 65–98 (2017). https://doi.org/10.1137/141000671
4. Bosma, W., Cannon, J., Playoust, C.: The magma algebra system. I. The user language. J. Symbolic. Comput. **24**(3–4), 235–265 (1997). https://doi.org/10.1006/jsco.1996.0125. Computational algebra and number theory (London, 1993)
5. Butler, G.: Computing in permutation and matrix groups. ii. Backtrack algorithm. Math. Comput. **39**, 671–680 (1982). https://doi.org/10.1090/S0025-5718-1982-0669659-5
6. Butler, G.: Computing normalizers in permutation groups. J. Algorithms **4**(2), 163–175 (1983). https://doi.org/10.1016/0196-6774(83)90043-3
7. Cannon, J., Holt, D.: Computing chief series, composition series and socles in large permutation groups. J. Symbolic Comput. **24**(3), 285–301 (1997). https://doi.org/10.1006/jsco.1997.0127
8. Dixon, J.D., Mortimer, B.: Permutation Groups, vol. 163. Springer, New York (1996). https://doi.org/10.1007/978-1-4612-0731-3
9. The GAP-Group: GAP - Groups, Algorithms, and Programming, Version 4.11.0 (2020). https://www.gap-system.org

10. Glasby, S.P., Slattery, M.C.: Computing intersections and normalizersin soluble groups. J. Symbolic Comput. **9**(5), 637–651 (1990). https://doi.org/10.1016/S0747-7171(08)80079-X
11. Guralnick, R.M., Maróti, A., Pyber, L.: Normalizers of primitive permutation groups. Adv. Math. **310**, 1017–1063 (2017). https://doi.org/10.1016/j.aim.2017.02.012
12. Holt, D.: The computation of normalizers in permutation groups. J. Symbolic Comput. **12**(4), 499–516 (1991). https://doi.org/10.1016/S0747-7171(08)80100-9
13. Hulpke, A.: Normalizer calculation using automorphisms. In: Computational Group Theory and the Theory of Groups. AMS special session On Computational Group Theory, Davidson, USA, pp. 105–114 (2007). https://doi.org/10.1090/conm/470
14. Jefferson, C., Pfeiffer, M., Waldecker, R., Wilson, W.A.: Permutation group algorithms based on directed graphs. arXiv preprint arXiv:1911.04783 (2019)
15. Kernighan, B.W., Ritchie, D.M.: The C Programming Language, 2nd edn. Prentice Hall, Englewood Cliffs (1988)
16. Leon, J.S.: Permutation group algorithms based on partitions i theory and algorithms. J. Symbolic Comput. **12**, 533–583 (1991)
17. Liebeck, M.W., Praeger, C.E., Saxl, J.: On the o'nan-scott theorem for finite primitive permutation groups. J. Aust. Math. Soc. Ser. A. Pure Math. Stat. **44**(3), 389–396 (1988). https://doi.org/10.1017/S144678870003216X
18. Luks, E., Miyazaki, T.: Polynomial-time normalizers. Discrete Math. Theor. Comput. Sci. **13**(4), 61–96 (2011)
19. McKay, B.D., Piperno, A.: Practical graph isomorphism, ii. J. Symbolic Comput. **60**, 94–112 (2014). https://doi.org/10.1016/j.jsc.2013.09.003
20. Neunhöffer, M., et al.: Recog, a collection of group recognition methods, Version 1.3.2, April 2018. https://gap-packages.github.io/recog. GAP package
21. Praeger, C.E.: The inclusion problem for finite primitive permutation groups. Proc. Lond. Math. Soc. **3**(1), 68–88 (1990). https://doi.org/10.1112/plms/s3-60.1.68
22. Schweitzer, P., Wiebking, D.: A unifying method for the design of algorithms canonizing combinatorial objects. arXiv e-prints arXiv:1806.07466, June 2018
23. Siccha, S.: Normalizers of primitive groups with non-regular socle in polynomial time. Ph.D. thesis, RWTH Aachen University. to appear
24. Sims, C.C.: Determining the conjugacy classes of permutation groups. Comput. Algebra Number Theo. **4**, 191–195 (1971)
25. Theißen, H.: Eine methode zur normalisatorberechnung in permutationsgruppen mit anwendungen in der konstruktion primitiver gruppen. Ph.D. thesis, RWTH Aachen University (1997)
26. Wiebking, D.: Normalizers and permutational isomorphisms in simply-exponential time. arXiv e-prints arXiv:1904.10454, April 2019
27. Wiebking, D.: Normalizers and permutational isomorphisms in simply-exponential time. In: Proceedings of the Thirty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, 5–8 January 2020, Salt Lake City, Utah, USA (2020)