Journal of Algebra 392 (2013) 315–335

Contents lists available at SciVerse ScienceDirect

Journal of Algebra

www.elsevier.com/locate/jalgebra

Fast recognition of alternating groups of unknown degree

Sebastian Jambor [a], Martin Leuner [b,*], Alice C. Niemeyer [c,d], Wilhelm Plesken [b]

[a] Department of Mathematics, The University of Auckland, Private Bag 92019, Auckland, New Zealand
[b] Lehrstuhl B für Mathematik, RWTH Aachen University, Templergraben 64, 52062 Aachen, Germany
[c] Lehrstuhl D für Mathematik, RWTH Aachen University, Templergraben 64, 52062 Aachen, Germany
[d] Centre for the Mathematics of Symmetry and Computation, University of Western Australia, Nedlands 6009, Australia

ARTICLE INFO

Article history:
Received 21 December 2012
Available online 19 June 2013
Communicated by Derek Holt

MSC:
20P05
20B30
20B40
68Q25
20-04

Keywords:
Black-box group
Constructive recognition
Alternating group
Probabilistic methods
Proportions of elements

ABSTRACT

We present a constructive recognition algorithm to decide whether a given black-box group is isomorphic to an alternating or a symmetric group without prior knowledge of the degree. This eliminates the major gap in known algorithms, as they require the degree as additional input.
Our methods are probabilistic and rely on results about proportions of elements with certain properties in alternating and symmetric groups. These results are of independent interest; for instance, we establish a lower bound for the proportion of involutions with small support.

1. Introduction

The computational recognition of finite simple groups is a fundamental task in the finite matrix group recognition project (see [8,9,11]). Generally not much is known about the way in which a group might be given as input and therefore algorithms which take black-box groups (see [1]) as input are the most versatile. For the important infinite family of alternating groups, the present black-box

* Corresponding author.
E-mail addresses: jambor@math.auckland.ac.nz (S. Jambor), leuner@momo.math.rwth-aachen.de (M. Leuner), alice.niemeyer@uwa.edu.au (A.C. Niemeyer), plesken@momo.math.rwth-aachen.de (W. Plesken).

0021-8693/$ – see front matter
http://dx.doi.org/10.1016/j.jalgebra.2013.06.005

algorithms [3,4] can only test whether a given black-box group is isomorphic to an alternating or a symmetric group of a particular degree, provided as additional input to the algorithm. Therefore deciding whether a given black-box group is isomorphic to an alternating group may require to run the algorithm once for each possible degree. The present paper describes a one-sided Monte-Carlo (see e.g. [12, p. 14]) black-box algorithm which avoids this bottleneck. Our algorithm takes as input a black-box group given by a set of generators together with a natural number N and decides whether the given group is isomorphic to an alternating group of any degree at most N. If the algorithm proves this to be the case, it computes the degree of the group and recognises it constructively. Otherwise the algorithm reports failure. Our algorithm runs in time nearly linear in N whereas the older algorithms have a runtime complexity of $\widetilde{\mathcal{O}}(N^2)$ to solve the same task in the worst case.

Given a black-box group G, we let μ denote an upper bound for the cost of multiplying two elements in G and let ρ denote an upper bound for the cost of computing a uniformly distributed, independent random element of G. Throughout this paper, log denotes the natural logarithm.

Theorem 1.1. *Algorithm* 4.29, RecogniseSnAn, *is a one-sided Monte-Carlo algorithm with the following properties. It takes as input a black-box group $G = \langle X \rangle$, a natural number N and a real number ε with $0 < \varepsilon < 1$. If $G \cong \mathrm{A}_n$ or $G \cong \mathrm{S}_n$ for some $9 \leqslant n \leqslant N$, it returns with probability at least $1 - \varepsilon$ the degree n and an isomorphism $\lambda : G \to \mathrm{A}_n$ or $\lambda : G \to \mathrm{S}_n$. Otherwise it reports failure. The algorithm runs in time $\mathcal{O}(N \log(N)^2 \log(\varepsilon^{-1})(|X|\mu + \rho))$ and stores at most $\mathcal{O}(\log(N))$ group elements at any moment.*

The black-box construction of a 3-cycle – one of the key ingredients of the algorithm – is a surprisingly hard problem. The solution lies in the combination of the following theoretical results, which are also of independent interest. The first allows us to find involutions with small support; the second uses these to construct a 3-cycle.

Theorem 1.2. *Let $9 \leqslant n \in \mathbb{N}$ and $G \in \{\mathrm{A}_n, \mathrm{S}_n\}$. The proportion of elements $x \in G$ of even order satisfying $|\operatorname{supp} x^{|x|/2}| \leqslant 4\sqrt{n}/3$ is at least $(13 \log(n))^{-1}$.*

Theorem 1.3. *Let $7 \leqslant n \in \mathbb{N}$, $G \in \{\mathrm{A}_n, \mathrm{S}_n\}$ and $1 \leqslant k \leqslant 2\sqrt{n}/3$. Let $s \in G$ be an involution moving $2k$ points.*

1. *The proportion of elements r in the conjugacy class s^G such that r and s move exactly one common point is at least $10/(3n)$.*
2. *Let M be the set of elements in s^G not commuting with s. The proportion of elements r in M such that $(sr)^2$ is a 3-cycle is at least $1/3$.*

The constructive recognition algorithm for alternating and symmetric groups described in [3] consists of two parts: the construction of standard generators assuming the degree is known, and the algorithmic construction of the inverse of the isomorphism $\lambda : G \to \mathrm{A}_n$. The contribution of this paper is to replace the first part by an algorithm determining the degree and finding the standard generators simultaneously. Together with the second part of [3], this establishes the algorithm for the main theorem above. If one is interested in recognising the symmetric group rather than the alternating group, the remarks of [3] apply and the same complexity is achieved.

Our algorithm has been implemented in the computer algebra system GAP [7]. Comparisons of our implementation with the GAP implementation of the first part of [3] show that our algorithm is a significant improvement. Given as input a black-box group isomorphic to a symmetric or alternating group, the new algorithm establishes this fact and determines the degree of the group in about the same time that the old algorithm requires to decide whether the input group is isomorphic to an alternating or symmetric group of the specific degree given as part of the input. In general, the old algorithm has to be run several times to find the degree of the input group. Therefore, the new algorithm wins out by a factor determined by the number of putative degrees the old algorithm has to test. The scope of our implementation depends on many factors, in particular the way the group is represented. To give a very rough indication, in the natural permutation representation the present implementation can deal with degrees of around 10 000.

In applications in the matrix group recognition project it is imperative that the algorithm reports failure quickly when the input group is not isomorphic to an alternating nor a symmetric group. We tested the performance of our algorithm when handed some examples of almost simple groups which are not alternating or symmetric. In all these examples our algorithm reported failure extremely fast. This is mainly due to finding an element of order not existing in the symmetric group of degree N, thus even proving that the group cannot be of the specified isomorphism types (cf. remark after Algorithm 4.1).

The practical performance of our algorithm exceeds its predicted performance as the constants in our estimates of proportions of elements are too conservative, notably in the proportion proved in Theorem 1.2. Further improvements of the performance could be achieved in situations where an order oracle is available by lowering the a priori upper bound N.

As E. O'Brien pointed out, our algorithm can also be applied to decide whether the input group G is a central extension of some (not necessarily finite) abelian group by A_n or S_n by working with $G/Z(G)$ as black-box group.

Here is a short overview of this paper. We fix some notation in Section 2 and give an outline of the algorithm in Section 3. In Section 4 we describe the setup in detail and prove Theorem 1.1. Finally, in Section 5 we give proofs of Theorems 1.2 and 1.3, along with proofs of some technical results which are used in the proof of Theorem 1.1.

2. Preliminaries

This paper describes a constructive recognition algorithm which decides whether a given black-box group is isomorphic to an alternating or a symmetric group. The notion of when a black-box group is *constructively recognisable* is defined in [3, Definition 1.1]. In particular, we note that if our algorithm concludes that a given black-box group G is indeed isomorphic to an alternating group A_n or a symmetric group S_n of some degree n, then it also determines an isomorphism $\lambda : G \to \mathrm{A}_n$ or $\lambda : G \to \mathrm{S}_n$ and a pair $\{s, t\}$ of generators for G, called the *standard generators of* G. We call λ together with the standard generators $\{s, t\}$ a *constructive isomorphism*.

The standard generators for A_n chosen by the algorithm satisfy the following presentations given by Carmichael [5]:

$$\left\{ s, t \;\middle|\; s^{n-2} = t^3 = (st)^{n-1} = \left(t^{(-1)^k} s^{-k} t s^k\right)^2 = 1 \text{ for } 1 \leqslant k \leqslant \frac{n-2}{2} \right\} \tag{1}$$

for even $n > 3$ and

$$\left\{ s, t \;\middle|\; s^{n-2} = t^3 = (st)^{n} = \left(t s^{-k} t s^k\right)^2 = 1 \text{ for } 1 \leqslant k \leqslant \frac{n-3}{2} \right\} \tag{2}$$

for odd $n > 3$.

Examples of standard generators for A_n are $s = (1,2)(3,4,\ldots,n)$ and $t = (1,2,3)$ for n even, and $s = (3,4,\ldots,n)$ and $t = (1,2,3)$ for n odd.

Our algorithm exploits information gained by considering the cycle types of permutations in symmetric groups. Recall that the *cycle type* of an element $g \in \mathrm{S}_n$ is defined as $1^{a_1} \cdots n^{a_n}$ if g contains a_i cycles of length i for $1 \leqslant i \leqslant n$. Note that for $n \geqslant 7$ we have $\mathrm{Aut}(\mathrm{A}_n) = \mathrm{S}_n$, so the cycle type is preserved by all automorphisms of A_n. Thus, if G is isomorphic to A_n or S_n, the cycle type of $\lambda(g)$ is independent of the choice of isomorphism λ from G to A_n or S_n. This allows us to generalise the notion of cycle type to elements of a black-box group G isomorphic to A_n or S_n.

During the course of the algorithm, we may encounter subgroups A_k of A_n. For $k \geqslant 7$ and k odd, given a 3-cycle $c \in \mathrm{A}_n$ we say that a k-cycle g *matches* c if $\{gc^2, c\}$ are standard generators for A_k. Note that in this case g must be of the form $(u, v, w, \ldots)$, where $c = (u, v, w)$ for $u, v, w \in \{1, \ldots, n\}$.

Let $\pi \in \mathrm{S}_n$. Call a point i with $1 \leqslant i \leqslant n$ a *moved point* of π if $i^\pi \neq i$. Call the set of moved points of π the *support* of π, denoted by $\operatorname{supp}\pi$. Similarly, denote by $\operatorname{fix}\pi$ the set of fixed points of π, that is $\{1, \ldots, n\} - \operatorname{supp}\pi$.

3. Brief outline of the algorithm

We describe a one-sided Monte-Carlo algorithm which takes as input a black-box group G, a real number ε with $0 < \varepsilon < 1$ and a positive integer N. The aim of the algorithm is to determine whether there is an integer n with $9 \leqslant n \leqslant N$ such that G is isomorphic to A_n or S_n. In the following we describe the main steps of our algorithm. We present this description under the assumption that the algorithm is given a black-box group G which is indeed isomorphic via the unknown isomorphism λ to A_n or S_n for some $n \leqslant N$ and describe the types of elements in G we seek to establish this fact. If the algorithm is handed a black-box group not isomorphic to an alternating or symmetric group, then one of the subsequent steps will fail to find the required elements and the algorithm reports failure.

The algorithm consists of three main steps. In the first step we compute a subset $R \subseteq G$ which contains a 3-cycle with high probability. The details are presented in Algorithm ThreeCycleCandidates in Section 4.1. If no such set R was found, then we conclude that G is not isomorphic to A_n or S_n for any n with $9 \leqslant n \leqslant N$ and terminate.

The second step repeats the following basic step for each element $c \in R$. We may assume without loss of generality that $\lambda(c) = (1, 2, 3)$ and we seek a k-cycle g matching c such that $k \geqslant 3n/4$. The construction of g is described in Algorithm ConstructLongCycle in Section 4.2. If no such element g was found, then we discard c as a putative 3-cycle and continue with the next candidate for c in R. Otherwise, without loss of generality, we may assume that $\lambda(g) = (1, 2, \ldots, k)$.

The third step, described in Algorithm StandardGenerators in Section 4.4, determines the degree n. This step repeats a basic step which computes random conjugates $r = g^x$ of g for $x \in G$. Note that by now we have derived some partial information about λ, namely $\lambda(c) = (1, 2, 3)$ and $\lambda(g) = (1, 2, \ldots, k)$. This allows us to decide whether $\operatorname{supp}\lambda(g^x)$ contains hereto unseen points in which case the basic step replaces g by an element g' such that $\lambda(g') = (1, 2, \ldots, \ell)$ for some $\ell > k$. The third step repeats this basic step until it obtains an n- or an $(n-1)$-cycle and constructs the standard generators for G from these.

Finally, we use methods from [3] to check whether we have found standard generators and compute a constructive isomorphism.

4. Details of the algorithm

In this section, the steps of the algorithm are described in detail. Each step in turn is broken down into one or more procedures. Each procedure is designed to accept an arbitrary black-box group as input, which forces the output to be fairly generic. Therefore each procedure has an accompanying lemma which gives an interpretation of the output if the input is in fact a symmetric or alternating group. A second lemma determines the complexity, which is valid for arbitrary black-box groups as input.

4.1. Construction of possible 3-cycles

The following algorithm constructs a set of putative 3-cycles. It is based on the simple observation that the product of two involutions t_1, t_2 with $|\operatorname{supp}(t_1) \cap \operatorname{supp}(t_2)| = 1$ squares to a 3-cycle.

Algorithm 4.1 *(ThreeCycleCandidates).*
Input: A group G, a real number $0 < \varepsilon < 1$ and $N \in \mathbb{N}$.
Output: A set $R \subset G$ or fail.
Algorithm:

1. Let $M := \prod_p p^{\lfloor \log_p(N) \rfloor}$, where the product is over all odd primes p with $p \leqslant N$. Let $B := \lceil 13 \log(N) \log(3/\varepsilon) \rceil$, $T := \lceil 3 \log(3/\varepsilon) \rceil$ and $C := \lceil 3NT/5 \rceil$.
2. Choose B random elements $r_1, \ldots, r_B \in G$ and set $t_i := r_i^M$ for $1 \leqslant i \leqslant B$.
3. For each t_i, if there is a smallest $a \in \mathbb{N}$ such that $t_i^{(2^a)} = 1_G$ and $a - 1 \leqslant \log_2(N)$, then replace t_i by $t_i^{(2^{a-1})}$. Otherwise return fail.

4. For each t_i set $\Gamma_i := \emptyset$. Repeat the following step at most C times: Choose a random conjugate c of t_i. If $t_i c \neq c t_i$ and $|\Gamma_i| < T$, then add c to Γ_i.
5. Return $\bigcup_{i=1}^{B} \{(t_i c)^2 \colon c \in \Gamma_i\}$.

Note that if the algorithm returns FAIL, then Step 3 has found an element $g \in G$ such that $|g|$ cannot be the order of any element in any group S_n for $n \leqslant N$. Hence G is proven not to be isomorphic to A_n or S_n for any $n \leqslant N$.

Lemma 4.2. *Let $9 \leqslant N \in \mathbb{N}$, $0 < \varepsilon < 1$ and $G \in \{S_n, A_n\}$ for some $9 \leqslant n \leqslant N$. A call to Algorithm* ThreeCycleCandidates(G, ε, N) *returns a subset R of G and, with probability at least $1 - \varepsilon$, R contains a 3-cycle in G. Moreover, $|R| \leqslant \lceil 13 \log(N) \log(3/\varepsilon) \rceil \cdot \lceil 3 \log(3/\varepsilon) \rceil$.*

Proof. Note that M is an odd integer and that for every $g \in G$ the element g^M has even order or is trivial. Therefore, by Corollary 5.6, with probability at least $1 - \varepsilon/3$ one of the t_i constructed in Step 2 has even order such that $t := t_i^{|t_i|/2}$ is a product of k disjoint transpositions with $k \leqslant \lfloor \max\{2\sqrt{n}/3, 2\} \rfloor$. Let X be a list of C random conjugates of t. Then, with probability at least $1 - \varepsilon/3$, X contains at least T elements which do not commute with t by Corollary 5.9. Now let Γ be a list of T random conjugates of t not commuting with t. By Corollary 5.10 there is, with probability at least $1 - \varepsilon/3$, an element $c \in \Gamma$ such that $(tc)^2$ is a 3-cycle. Thus, with probability at least $(1 - \varepsilon/3)^3 \geqslant 1 - \varepsilon$, the set R contains a 3-cycle. Since after Step 4 we have $|\Gamma_i| \leqslant T$, clearly $|R| \leqslant T \cdot B$ holds. This implies the claimed bound for $|R|$. □

Lemma 4.3. *Let G be a finite group, $0 < \varepsilon < 1$ and $N \in \mathbb{N}$. Then* ThreeCycleCandidates *with input G, ε, N runs in $\mathcal{O}(N \log(N)^2 \log(\varepsilon^{-1})^2(\mu + \rho))$ time and requires storage of $\mathcal{O}(\log(N) \log(\varepsilon^{-1})^2)$ group elements.*

Proof. Since $M < \prod_{2<p\leqslant N} N \leqslant N^N$, computing the M-th power of a group element with a square-and-multiply algorithm requires $\mathcal{O}(N \log(N))$ group operations. In Step 2 we construct B random elements and compute their M-th power. We compute $t_i^{(2^a)}$ by repeated squaring, ensuring that $a - 1 \leqslant \log_2(N)$, thus Step 3 can be performed in $B \cdot \log_2(N)$ group operations. Step 4 requires $B \cdot C$ random elements and $\mathcal{O}(B \cdot C)$ group operations; likewise, Step 5 requires $\mathcal{O}(B \cdot T)$ group operations. Thus, the total runtime of the algorithm is $\mathcal{O}(N \log(N)^2 \log(\varepsilon^{-1})^2(\mu + \rho))$.

Clearly, we only need to store $\mathcal{O}(\log(N) \log(\varepsilon^{-1})^2)$ elements overall, concluding the proof. □

4.2. Construction of a matching cycle

The aim of this section is, given a 3-cycle c in a black-box group G isomorphic to an alternating or symmetric group of degree n, to construct a k-cycle g matching c with $k \geqslant 3n/4$. The proportion of cycles with this property is too small for our purposes, so we consider other types of elements in G which occur more frequently and allow the construction of a k-cycle g with the desired properties. As a first step, we describe what we call bolstering elements. These allow us to construct the desired cycle g easily. Since bolstering elements are still too rare, we consider pre-bolstering elements from which we obtain bolstering elements in turn.

4.2.1. Bolstering elements

Let c be a 3-cycle with $\operatorname{supp} c = \{u, v, w\}$. Call an element $x \in S_n$ *bolstering* with respect to c if it is of the form $x = (v, a_1, \ldots, a_\alpha)(w, b_1, \ldots, b_\beta)(\ldots)$ or $x = (v, a_1, \ldots, a_\alpha, w, b_1, \ldots, b_\beta)(\ldots)$ with $u \in \operatorname{fix} x$ and $\alpha, \beta \geqslant 2$.

Remark 4.4. Given a bolstering element x with respect to the 3-cycle $c = (u, v, w)$, we can find a cycle g matching c. Let $m := \min\{\alpha, \beta\}$ and $m' := \lfloor |\alpha - \beta|/2 \rfloor$.

1. $c \cdot c^{x} \cdot c^{(x^2)} \cdots c^{(x^m)} =: y$ is a single cycle of length $2m+3$.
2. If $\alpha \leqslant \beta - 2$, we can compute $z = (u, b_{\alpha+2}, b_{\alpha+1})$. Then m' is the least positive integer such that $z^{(x^{2(m'+1)})}c$ does not have order 5 and $y \cdot z \cdot z^{(x^2)} \cdot z^{(x^4)} \cdots z^{(x^{2(m'-1)})} =: g$ is a cycle of length $2m' + 2m + 3$.
3. If $\beta \leqslant \alpha - 2$, we compute $z := (u, a_{\beta+1}, a_{\beta+2})$ to obtain a $(2m' + 2m + 3)$-cycle in similar fashion.

The details of how to compute z will be described in Algorithm BuildCycle.

Since the proportion of bolstering elements with respect to a given 3-cycle in A_n and S_n is too small, we instead try to find pre-bolstering elements and use these to construct bolstering elements.

An element r is called *pre-bolstering* with respect to c if it is of the form

$$r = (w, u, a_1, \ldots, a_\alpha)(v, b_1, \ldots, b_\beta)(\ldots)$$

or

$$r = (w, u, a_1, \ldots, a_\alpha, v, b_1, \ldots, b_\beta)(\ldots)$$

with $\operatorname{supp} c = \{u, v, w\}$ and $\alpha, \beta \geqslant 2$. Note that if r is pre-bolstering, then either $x = cr$ or $x = c^2 r$ is bolstering with respect to c.

The next lemma gives a criterion when an element $r \in S_n$ is pre-bolstering with respect to a 3-cycle c.

Lemma 4.5. *Let $c \in S_n$ be a 3-cycle. Then r is pre-bolstering with respect to c if and only if $[c^r, c] \neq 1_G$, $c^{(r^2)} \notin \{c, c^2\}$ and $[c, c^{(r^2)}] = 1_G$.*

Proof. Clearly, if r is pre-bolstering, then the conditions hold. Conversely, suppose that r is not pre-bolstering. Then either $\operatorname{supp} c^r \cap \operatorname{supp} c = \emptyset$ or $\operatorname{fix} r \cap \operatorname{supp} c \neq \emptyset$ or $\min\{\alpha, \beta\} < 2$. In the first case we find $[c^r, c] = 1_G$. In both the second and the third cases, clearly $\operatorname{supp} c \cap \operatorname{supp} c^{(r^2)} \neq \emptyset$, thus either $[c, c^{(r^2)}] \neq 1_G$ or $\operatorname{supp} c = \operatorname{supp} c^{(r^2)}$ holds. (Note that if the supports of c and $c^{(r^2)}$ coincide, then $c^{(r^2)} = c$ or $c^{(r^2)} = c^2$.) □

For a group G isomorphic to an alternating or a symmetric group and a 3-cycle $c \in G$, the following algorithm constructs a list of bolstering elements with respect to c. It achieves this by selecting a number of random elements from G and using the criteria in Lemma 4.5 to recognise pre-bolstering elements among these. From these it then constructs bolstering elements with respect to c.

Algorithm 4.6 *(BolsteringElements).*
Input: A group G, an element $c \in G$, a real number ε with $0 < \varepsilon < 1$ and $N \in \mathbb{N}$.
Output: A list B with $B \subset G$.
Algorithm:

1. Let $S := 7N\lceil \frac{7}{4} \log \varepsilon^{-1} \rceil$ and $R := \lceil \frac{7}{4} \log \varepsilon^{-1} \rceil$.
2. Set $C := \emptyset$. Repeat the following step at most S times: choose a random element $r \in G$; if $[c^r, c] \neq 1_G$, $c^{(r^2)} \notin \{c, c^2\}$, $[c, c^{(r^2)}] = 1_G$ and $|C| < R$, then add r to C.
3. For each $r \in C$, compute $z_r := c^{rcr} c^{rc^{(r^2)}c}$. If $(z_r)^3 = 1_G$, then add $c^2 r$ to B. Otherwise add cr to B. Return B.

Lemma 4.7. *Let $7 \leqslant n \leqslant N$, $G \in \{S_n, A_n\}$, $c \in G$ be a 3-cycle and $0 < \varepsilon < 1$. Let $B :=$ BolsteringElements(G, c, ε, N). Then B is a list of random bolstering elements and, with probability at least $1 - \varepsilon$, we have $|B| \geqslant \lceil \frac{7}{4} \log \varepsilon^{-1} \rceil$.*

Proof. Let $\operatorname{supp} c = \{u, v, w\}$. Clearly, using Lemma 4.5 the elements r constructed in Step 1 of Algorithm 4.6 are pre-bolstering with respect to c. Step 3 has to decide whether $c = (u, v, w)$ or $c = (u, w, v)$. In the first case z_r is a 3-cycle, while in the second case z_r is a 5-cycle. Thus, $(z_r)^3 = 1_G$ if and only if $c = (u, v, w)$ and B is a list of bolstering elements. By Proposition 5.12, we find less than R elements with probability at most ε, since $S = 7N\lceil \frac{7}{4} \log \varepsilon^{-1} \rceil \geqslant 5N \cdot \max((5/4)^4 \log \varepsilon^{-1}, \frac{25}{18} \lceil \frac{1}{2} \log_{3/4} \varepsilon \rceil)$. □

Lemma 4.8. *Let G be a black-box group, $c \in G$ an arbitrary element, $0 < \varepsilon < 1$ and $N \in \mathbb{N}$. Then Algorithm* BolsteringElements *with input G, c, ε, N runs in $\mathcal{O}(N \log \varepsilon^{-1}(\mu + \rho))$ time and requires storage of $\mathcal{O}(\log \varepsilon^{-1})$ group elements.*

Proof. This is immediate. □

4.2.2. Exploiting bolstering elements

Given a bolstering element x with respect to a 3-cycle c, we can construct a cycle g_x matching c, using Remark 4.4. But depending on the type of the bolstering element, this may require different steps to obtain the longest possible matching cycle. The type of a given bolstering element can be determined using only black-box operations as described in Remark 4.9. We first describe Algorithm BuildCycle which applies this remark to obtain a cycle g_x matching c from a given bolstering element x. This is used by Algorithm ConstructLongCycle, which computes g_x for every x returned by Algorithm BolsteringElements, and returns the longest g_x.

Remark 4.9. Several properties of bolstering elements can be checked algorithmically using only black-box operations. Let $\operatorname{supp} c = \{u, v, w\}$. Let x be bolstering with respect to c and $u \in \operatorname{fix} x$.

1. Let $m := \min\{\alpha, \beta\}$. Then m is the least natural number such that $c^{(x^{m+1})}c$ does not have order 5. Note that necessarily $m < n/2$.
2. $\alpha = \beta$ if and only if $c^{(x^{m+1})} \in \{c, c^2\}$.
3. $|\alpha - \beta| = 1$ if and only if $c^{(x^{m+2})}c$ does not have order 5.
4. If $\alpha \neq \beta$, then $w \notin v^{\langle x \rangle}$, i.e. x is of the first form, if and only if $c^{(x^{m+1})}c$ has order 2.
5. Assume $|\alpha - \beta| > 1$. If $w \in v^{\langle x \rangle}$, then $\alpha > \beta$ if and only if $c^{(x^{m+2})}$ and $c^{(x^{m+1}c)}$ commute. If $w \notin v^{\langle x \rangle}$, then $\alpha < \beta$ if and only if $c^{(x^{m+2})}$ and $c^{(x^{m+1}c)}$ commute.

When called with input a black-box group G isomorphic to an alternating group A_n or a symmetric group S_n and elements $c, x \in G$ such that c is a 3-cycle and x is a bolstering element with respect to c, the following algorithm determines a cycle g_x of length k matching c. It returns g_x and its length k.

Algorithm 4.10 (BuildCycle).
Input: Elements c, x of a group G and $N \in \mathbb{N}$.
Output: A number $k \in \mathbb{N}$ and an element $g \in G$, or fail.
Algorithm: Determine $m := \min\{\alpha, \beta\}$ and check whether $|\alpha - \beta| \geqslant 2$ as described in Remark 4.9. If $m \geqslant N/2$, return fail. Compute $y := c \cdot c^x \cdot c^{(x^2)} \cdots c^{(x^m)}$. If $|\alpha - \beta| \leqslant 1$, return $2m + 1, y$. Otherwise set $d := c^{(x^{m+1})}$ and

$$e := \begin{cases} d^{xc}, & \text{if } w \in v^{\langle x \rangle} \text{ and } \alpha > \beta, \\ (d^{(xc^2)})^2, & \text{if } w \in v^{\langle x \rangle} \text{ and } \alpha < \beta, \\ d^{(xc^2)}, & \text{if } w \notin v^{\langle x \rangle} \text{ and } \alpha > \beta, \\ (d^{(xc)})^2, & \text{if } w \notin v^{\langle x \rangle} \text{ and } \alpha < \beta, \end{cases}$$

where we can decide whether $w \in v^{\langle x \rangle}$ and $\alpha > \beta$ using Remark 4.9. Set $z := d^e$ and determine m' as described in Remark 4.4. If $m' \geqslant N/2$, return FAIL. Otherwise compute $g := y \cdot z \cdot z^{(x^2)} \cdots z^{(x^{2(m'-1)})}$. Return $2m' + 2m + 1, g$.

Lemma 4.11. *Let $7 \leqslant n \leqslant N \in \mathbb{N}$, $c \in S_n$ be a 3-cycle and x a bolstering element with respect to c. Then* BuildCycle *with input c, x, N returns k and g such that g is a k-cycle matching c.*

Proof. This is an application of Remarks 4.4 and 4.9, where it is easy to check that z has the form given in Remark 4.4, e.g., if $w \in v^{\langle x \rangle}$ and $\alpha > \beta$ we have $d = (u, a_{\beta+1}, v)$ and $e = (v, a_{\beta+2}, a_1)$, hence $z = (u, a_{\beta+1}, a_{\beta+2})$. □

Lemma 4.12. *Let G be a finite group, $c, x \in G$ arbitrary elements, and $N \in \mathbb{N}$. Then* BuildCycle *with input c, x, N runs in $\mathcal{O}(N\mu)$ time and requires storage of a constant number of group elements.*

Proof. By storing $c^{(x^{i-1})}$, the next element $c^{(x^i)}$ can be computed in constant time. Since m and m' are bounded by $N/2$, the lemma follows. □

Algorithm 4.13 (ConstructLongCycle).
Input: A group G, an element $c \in G$, $0 < \varepsilon < 1$ and $N \in \mathbb{N}$.
Output: A number $k \in \mathbb{N}$ and an element $g \in G$ or FAIL.
Algorithm:

1. Let $L := \emptyset$ and $B :=$ BolsteringElements$(G, c, \varepsilon/2, N)$. If B contains less than $\lceil \frac{7}{4} \log(2/\varepsilon) \rceil$ elements, return FAIL.
2. Call BuildCycle for each bolstering element $x \in B$. If this fails for some x, return FAIL. Otherwise return k and g computed by BuildCycle with maximal k.

Lemma 4.14. *Let $9 \leqslant N \in \mathbb{N}$, $0 < \varepsilon < 1$, $G \in \{S_n, A_n\}$ for some $9 \leqslant n \leqslant N$ and $c \in G$ be a 3-cycle. Then, with probability at least $1 - \varepsilon$,* ConstructLongCycle *with input G, c, ε, N returns k and g such that $k \geqslant \max(3n/4, 9)$ and g is a k-cycle matching c.*

Proof. Step 1 succeeds with probability at least $1 - \varepsilon/2$, cf. Lemma 4.7. Since $7/4 \log(2/\varepsilon) \geqslant 1/2 \log_{3/4}(\varepsilon/2)$, Proposition 5.13 yields that, with probability at least $1 - \varepsilon/2$, BuildCycle constructs at least one k-cycle with $k \geqslant \max(3n/4, 9)$. □

Lemma 4.15. *Let G be a finite group, $c \in G$ an arbitrary element, $0 < \varepsilon < 1$ and $N \in \mathbb{N}$. Then* ConstructLongCycle *with input G, c, ε, N runs in $\mathcal{O}(N \log \varepsilon^{-1}(\mu + \rho))$ time and requires storage of $\mathcal{O}(\log \varepsilon^{-1})$ group elements.*

Proof. This follows from Lemmas 4.8 and 4.12. □

4.3. Auxiliary algorithms

In this section we describe short algorithms which are called by Algorithm StandardGenerators. For our discussion, we assume we are given a group G isomorphic to A_n or S_n and that c is a 3-cycle and g a k-cycle matching c. We perform computations mainly in $\langle g, c \rangle \cong A_k$.

The first algorithm decides whether a point $i \in \operatorname{supp} g$ is fixed by a given element $r \in G$.

Remark 4.16. Let $a_1, \ldots, a_7 \in \mathbb{N}$ be pairwise distinct and

$$A := \big\{ \{1, 2, i\} \colon 3 \leqslant i \leqslant 6 \big\}.$$

If the sets $\{a_1, a_2, a_3\}$, $\{a_1, a_4, a_5\}$, $\{a_1, a_6, a_7\}$ intersect each set in A non-trivially, then $a_1 \in \{1, 2\}$.

This observation allows us to recognise a fixed point of an arbitrary element $r \in G$ by examining the intersection of the supports of some aptly chosen elements. If c is a 3-cycle and g a matching cycle, the following algorithm decides whether the single point in the intersection of the supports of c and $c^{(g^2)}$ is fixed by r.

Algorithm 4.17 (IsFixedPoint).
Input: Elements g, c, r of a group G.
Output: TRUE or FALSE.
Algorithm: Define

$$X := \left\{c^r, c^{g^2 r}, c^{g^2 c^{(g^3)} c^{(g^4)} r}\right\}$$

and

$$H_1 := \left\{c^2, c^{c^g}, c^{c^g c^{(g^3)}}, c^{c^g (c^{(g^3)})^2}, c^{c^g (c^{(g^3)})^2 c^{(g^4)}}\right\}.$$

If there is an element $x \in X$ such that $[x, h] = 1_G$ for at least two different $h \in H_1$, then return FALSE. Otherwise define

$$H_2 := \left\{c, c^g, c^{g c^{(g^3)}}, c^{g (c^{(g^3)})^2}, c^{g (c^{(g^3)})^2 c^{(g^4)}}\right\}.$$

If there is an element $x \in X$ such that $[x, h] = 1_G$ for at least two different $h \in H_2$, then return FALSE. Otherwise return TRUE.

Lemma 4.18. *Let $7 \leqslant k \leqslant n$, $c \in \mathrm{S}_n$ be a 3-cycle, $g \in \mathrm{S}_n$ a k-cycle matching c and $r \in \mathrm{S}_n$ an arbitrary element.* IsFixedPoint(g, c, r) *returns* TRUE *if and only if the unique point contained in both* $\operatorname{supp} c$ *and* $\operatorname{supp} c^{(g^2)}$ *is fixed by r.*

Proof. Without loss of generality, let $c = (1, 2, 3)$ and $g = (1, 2, \ldots, k)$. We find $\operatorname{supp} c \cap \operatorname{supp} c^{(g^2)} = \{3\}$, $H_1 = \{(1, 3, j)\colon\ j \in \{2, 4, 5, 6, 7\}\}$, $H_2 = \{(2, 3, j)\colon\ j \in \{1, 4, 5, 6, 7\}\}$ and $X = \{(1, 2, 3)^r, (3, 4, 5)^r, (3, 6, 7)^r\}$.

Assume that IsFixedPoint returns FALSE. Then there are elements $x \in X$ and $h_1, h_2 \in H_1$ (or in H_2) commuting with x. Suppose $3 \in \operatorname{fix} r$. Since then $3 \in \operatorname{supp} x \cap \operatorname{supp} h_1 \cap \operatorname{supp} h_2$ and h_1, h_2 commute with x, we obtain $\operatorname{supp} h_1 = \operatorname{supp} x = \operatorname{supp} h_2$, a contradiction. Thus $3 \notin \operatorname{fix} r$.

Conversely assume that IsFixedPoint returns TRUE. Then, for each $x \in X$, there exist $h_1, \ldots, h_4 \in H_1$ with $\operatorname{supp} h_i \cap \operatorname{supp} x \neq \emptyset$, and similarly for H_2. The result now follows by Remark 4.16. □

Lemma 4.19. *Let G be a finite group and $g, c, r \in G$ arbitrary elements. Then* IsFixedPoint *with input g, c, r uses a constant number of group operations and requires storage of a constant number of group elements.*

Proof. This is immediate. □

Let G be a black-box group isomorphic to an alternating or symmetric group, $c \in G$ a 3-cycle, $g \in G$ a k-cycle matching c, and r another element of G. Assume without loss of generality that $g = (1, 2, \ldots, k)$ and $c = (1, 2, 3)$. If r satisfies $|\operatorname{supp} r \cap \operatorname{supp} g| \geqslant 1$ and $|\operatorname{fix} r \cap \operatorname{supp} g| \geqslant 2$, the next algorithm computes a conjugate $\widetilde{r} = r^x$ such that $\widetilde{r}$ fixes the points 1 and 2, but not the point 3. Here we identify the point $j \in \{1, \ldots, k\}$ with the 3-cycle $c^{g^{(j-3)}}$.

Algorithm 4.20 (AdjustCycle).
Input: Elements g, c, r of a group G and $k \in \mathbb{N}$.
Output: An element $\widetilde{r} \in G$ conjugate to r or FAIL.

Algorithm: Compute the set

$$F := \left\{1 \leqslant j \leqslant k\colon \textsc{IsFixedPoint}\left(g, c^{(g^{j-3})}, r\right) = \textsc{true}\right\}.$$

If $|F| < 2$ or $|F| = k$, then return FAIL. Otherwise, define f_1 as the smallest and f_2 as the second smallest number in F. Define m as the smallest natural number not in F. Define the element $x \in G$ according to the following table:

$F \cap \{1,2,3,4\}$	x
$\{1,2,3,4\}$ or $\{1,2,3\}$	$c^{(gc^2)^{m-3}c}c$
$\{1,2,4\}$ or $\{1,2\}$	1_G
$\{1,3,4\}$	c^g
$\{1,3\}$	$(c^2)^g$
$\{1,4\}$ or $\{1\}$	$c^{(gc^2)^{f_2-3}c}$
$\{2,3,4\}$ or $\{2,4\}$	c^{c^g}
$\{2,3\}$	$(c^2)^{c^g}$
$\{2\}$	$c^{(gc^2)^{f_2-3}c^g}$
$\{3,4\}$ or $\{3\}$	$(c^2)^{(gc^2)^{f_2-3}}c^2$
$\{4\}$ or $\emptyset$	$c^{(gc^2)^{f_2-3}c^{(gc^2)^{f_1-3}}}$

Return $\widetilde{r} := r^x$.

Lemma 4.21. *Let $7 \leqslant k_0 \leqslant k \leqslant n \in \mathbb{N}$, $c = (1,2,3)$, $g = (1,2,\ldots,k)$ and $r \in S_n$ be a k_0-cycle. If r has in* $\operatorname{supp} g$ *at least two fixed points and one moved point, then $\widetilde{r} :=$ AdjustCycle(g,c,r,k) is a k_0-cycle fixing the points* 1 *and* 2 *and moving* 3*. Moreover, the difference* $\operatorname{supp} r - \operatorname{supp} g$ *lies in* $\operatorname{supp}\widetilde{r}$*.*

Proof. If r has two fixed points and a moved point in $\operatorname{supp} g$, the algorithm returns a k_0-cycle $\widetilde{r}$. We want to show that $\widetilde{r}$ fixes the points 1 and 2 but moves the point 3. By Lemma 4.18, we have $F = \operatorname{fix} r \cap \operatorname{supp} g$. Then the table defining x looks as follows:

$F \cap \{1,2,3,4\}$	x
$\{1,2,3,4\}$ or $\{1,2,3\}$	$(1,2)(3,m)$
$\{1,2,4\}$ or $\{1,2\}$	1_G
$\{1,3,4\}$	$(2,3,4)$
$\{1,3\}$	$(2,4,3)$
$\{1,4\}$ or $\{1\}$	$(2,3,f_2)$
$\{2,3,4\}$ or $\{2,4\}$	$(1,3,4)$
$\{2,3\}$	$(1,4,3)$
$\{2\}$	$(1,3,f_2)$
$\{3,4\}$ or $\{3\}$	$(1,f_2)(2,3)$
$\{4\}$ or $\emptyset$	$(1,f_1)(2,f_2)$

Thus, in each case $\widetilde{r} = r^x$ fixes 1 and 2 but not 3. Since $x \in \langle g, c\rangle$, it fixes every element in $\{1,\ldots,n\} - \operatorname{supp} g$, so $(\operatorname{supp} r - \operatorname{supp} g) \subset \operatorname{supp}\widetilde{r}$ holds. □

Lemma 4.22. *Let G be a finite group, $g,c,r \in G$ arbitrary elements and $k \in \mathbb{N}$.* AdjustCycle *with input g,c,r,k runs in $\mathcal{O}(k\mu)$ time and requires storage of a constant number of group elements.*

Proof. This follows by standard arguments. □

Using elements provided by AdjustCycle, the next algorithm appends new points to the cycle g. Since g will always be a cycle of odd length, new points can only be appended in pairs. Because of this we need an element s, a 'storage cycle', storing the first new point until we encounter a second one. The output $\widetilde{s}$ assumes the role of s the next time AppendPoints is called.

Algorithm 4.23 (AppendPoints).
Input: Elements g, c, r, s of a group G and $k, k_0 \in \mathbb{N}$.
Output: Two elements $\widetilde{g}, \widetilde{s} \in G$ and $\widetilde{k} \in \mathbb{N}$.
Algorithm:

1. Set $\widetilde{g} := g$, $\widetilde{s} := s$ and $\widetilde{k} := k$.
2. For each $1 \leqslant j < k_0$, set $x_j := c^{(r^j)}$. If $[x_j, \widetilde{g}c^2] = 1_G$, then perform Step 3.
3. If $\widetilde{s} = 1_G$, then set $\widetilde{s} := x_j$. If $\widetilde{s} \neq 1_G$ and $\widetilde{s} \neq x_j$, then set $\widetilde{k} := \widetilde{k} + 2$, $\widetilde{g} := \widetilde{g}\widetilde{s}^{(x_j^2)}$ and $\widetilde{s} := 1_G$.
4. Return $\widetilde{g}$, $\widetilde{s}$ and $\widetilde{k}$.

Lemma 4.24. *Let $7 \leqslant k_0 \leqslant k \leqslant n \in \mathbb{N}$, $c = (1, 2, 3)$, $g = (1, 2, \ldots, k)$ and $r \in \mathrm{S}_n$ be a k_0-cycle fixing the points 1 and 2 and moving 3. Let $s \in \mathrm{S}_n$ be either the identity element or $s = (1, 2, b)$ for some $b \in \{1, \ldots, n\} - \operatorname{supp} g$. Let $\widetilde{g}, \widetilde{s}, \widetilde{k} :=$ AppendPoints(g, c, r, s, k, k_0). Then $\widetilde{g}$ is a $\widetilde{k}$-cycle matching c, and $\operatorname{supp} r \cup \operatorname{supp} g \cup \operatorname{supp} s = \operatorname{supp} \widetilde{g} \cup \operatorname{supp} \widetilde{s}$.*

Proof. Let $r = (3, a_1, \ldots, a_{k_0-1})$ with $4 \leqslant a_j \leqslant n$. Then $x_j = (1, 2, a_j)$, so x_j and $\widetilde{g}c^2$ commute if and only if $a_j \notin \operatorname{supp} \widetilde{g}$. If, in this case, $\widetilde{s}$ is the identity, the new point is stored in $\widetilde{s}$. If $\widetilde{s} = x_j$, the point is already stored in s. Otherwise we find $\widetilde{s} = (1, 2, b)$ for some $b \notin (\operatorname{supp} \widetilde{g} \cup \{a_j\})$. Now, $\widetilde{g}$ is set to $(1, 2, \ldots, k, b, a_j)$, becoming a $\widetilde{k}$-cycle matching c. Since all a_j are treated in this manner, clearly $\operatorname{supp} r \subset (\operatorname{supp} \widetilde{g} \cup \operatorname{supp} \widetilde{s})$ holds. □

Lemma 4.25. *Let G be a finite group, $g, c, r, s \in G$ arbitrary elements and $k, k_0 \in \mathbb{N}$. Then* AppendPoints *with input g, c, r, s, k, k_0 runs in $\mathcal{O}(k_0 \mu)$ time and requires storage of a constant number of group elements.*

Proof. This is immediate. □

4.4. Construction of standard generators

Let G be a black-box group isomorphic to an alternating or symmetric group, $c \in G$ a 3-cycle and $g \in G$ a k-cycle matching c. The first algorithm in this section uses these elements to construct standard generators of the alternating group of the same degree as G.

The main algorithm RecogniseSnAn ties up all algorithms in this chapter and results of [3] to either constructively recognise the group or decide that it is not isomorphic to an alternating or symmetric group with high probability.

Algorithm 4.26 (StandardGenerators).
Input: A group G, elements $g, c \in G$, $0 < \varepsilon < 1$ and $k, N \in \mathbb{N}$.
Output: Elements $\widetilde{g}, \widetilde{c} \in G$ and $\widetilde{k} \in \mathbb{N}$ or FAIL.
Algorithm:

1. Set $s := 1_G$, $k_0 := k - 2$, $r := gc^2$, $\widetilde{k} := k$ and $\widetilde{g} := g$.
2. Choose a list R of $\lceil \log(10/3)^{-1}(\log N + \log \varepsilon^{-1}) \rceil$ random conjugates of r. For each $x \in R$, perform Step 3.
3. Set $m :=$ AdjustCycle$(\widetilde{g}, c, x, \widetilde{k})$. If $m =$ FAIL, then return FAIL.
   Set $\widetilde{g}, s, \widetilde{k} :=$ AppendPoints$(\widetilde{g}, c, m, s, \widetilde{k}, k_0)$. If $\widetilde{k} > N$, then return FAIL.
4. If $s = 1_G$, set $\widetilde{g} := c^2\widetilde{g}$ and $\widetilde{c} := c$. Otherwise set $\widetilde{k} := \widetilde{k} + 1$, $\widetilde{g} := \widetilde{g}s$ and $\widetilde{c} := s$.
5. Check whether $(\widetilde{g}, \widetilde{c})$ satisfies the presentation (1) or (2) for A_k. If that is not the case, then return FAIL. Otherwise return $\widetilde{g}, \widetilde{c}, \widetilde{k}$.

Lemma 4.27. *Let $9 \leqslant k \leqslant n \leqslant N \in \mathbb{N}$, $k \geqslant 3n/4$, $G \in \{\mathrm{S}_n, \mathrm{A}_n\}$, $c \in G$ be a 3-cycle, $g \in G$ a k-cycle matching c and $0 < \varepsilon < 1$. Then, with probability at least $1 - \varepsilon$, we find $\widetilde{g}, \widetilde{c}, \widetilde{k} :=$ StandardGenerators$(G, g, c, \varepsilon, k, N) \neq$ FAIL such that $\widetilde{k} = n$ and $\widetilde{g}, \widetilde{c}$ are standard generators for A_n.*

Proof. First note that $k_0 \geqslant \lceil (7/10)n \rceil$ and r is a k_0-cycle, so the supports of $\widetilde{g}$ and a random conjugate x of r always have a common moved point. Furthermore, x has at least two fixed points in $\operatorname{supp} \widetilde{g}$ since $k = k_0 + 2$, so the algorithm cannot fail in Step 3. Lemmas 4.21 and 4.24 ensure that after Step 2 the set $\operatorname{supp} \widetilde{g} \cup \operatorname{supp} s$ contains the supports of all $x \in R$. Thus, by Theorem 5.14, we find that with probability at least $1 - \varepsilon$ the elements $\widetilde{g}$ and s have no common fixed point on $\{1, \ldots, n\}$. It is easy to check that we return the correct degree and standard generators. □

Lemma 4.28. *Let G be a group, $g, c \in G$ arbitrary elements, $0 < \varepsilon < 1$ and $k, N \in \mathbb{N}$. Then* StandardGenerators *with input $G, g, c, \varepsilon, k, N$ runs in $\mathcal{O}(N(\log N + \log \varepsilon^{-1})(\mu + \rho))$ time and requires storage of a constant number of group elements.*

Proof. The cost to check whether a presentation for A_k is satisfied requires $\mathcal{O}(N)$ group operations by [3, Lemma 4.4]. At any call of AdjustCycle and AppendPoints we have $k \leqslant N$. Thus, Lemmas 4.22 and 4.25 yield the claimed runtime. □

We can now present the main algorithm and prove the main Theorem 1.1.

Algorithm 4.29 *(*RecogniseSnAn*).*
Input: A group $G = \langle X \rangle$, $0 < \varepsilon < 1$ and $N \in \mathbb{N}$.
Output: A constructive isomorphism or fail.
Algorithm:

1. Set $T := \lceil \log_2 \varepsilon^{-1} \rceil$.
2. If $T = 0$, then return fail. Otherwise set $T := T - 1$ and compute $R :=$ ThreeCycleCandidates$(G, 1/4, N)$. If $R =$ fail, then return fail.
3. If $R = \emptyset$, go to Step 2. Otherwise choose $c \in R$ and set $R := R - \{c\}$.
4. Set $\ell :=$ ConstructLongCycle$(G, c, 1/8, N)$. If $\ell =$ fail, go to Step 3. Otherwise set $k, g := \ell \in \mathbb{N} \times G$.
5. Set $\ell :=$ StandardGenerators$(G, g, c, 1/8, k, N)$. If $\ell =$ fail, go to Step 3. Otherwise set $g, c, n := \ell \in G \times G \times \mathbb{N}$.
6. Using methods described in [3], check whether G is isomorphic to A_n or S_n. If that is the case, then return the constructive isomorphism computed during the check. Otherwise go to Step 3.

Proof of Theorem 1.1. For the first part of the statement, consider Steps 2–6. Note that ThreeCycleCandidates cannot fail if G is an alternating or symmetric group of degree at most N, so by Lemma 4.2 we obtain a set R containing a 3-cycle with probability at least 3/4. Thus, without loss of generality, let $c \in R$ be a 3-cycle. Using Lemma 4.14, we find, with probability at least 7/8, that Step 4 constructs a k-cycle matching c with $k \geqslant \max(3n/4, 9)$. Now, by Lemma 4.27, Step 5 returns the correct degree and standard generators with probability at least 7/8. Step 6 always returns a correct answer, cf. [3, Lemma 5.5 and proof of Theorem 1.2(b)]. Thus, the probability to succeed in one pass is at least $(3/4) \cdot (7/8)^2 > 1/2$. We repeat this procedure $\lceil \log_2 \varepsilon^{-1} \rceil$ times to obtain the claimed overall probability.

We now prove the second claim. Steps 2–6 are repeated up to $\lceil \log_2 \varepsilon^{-1} \rceil$ times. During one such pass we execute Step 2 only once and Steps 4–6 up to $|R|$ times. By Lemma 4.2 we have $|R| \leqslant c \log N$ for some constant $c \in \mathbb{R}$. In Step 5, note that $k, n \leqslant N$ must hold. Then the claim follows by Lemmas 4.3, 4.15, 4.28 and [3, Section 5]. □

5. Probability estimates

This section contains theoretical results which are used to establish lower bounds for the success probability of the algorithm. Several results are of independent interest. We already mentioned the probability estimates for small support involutions in the introduction. Another noteworthy result is a lower bound on the proportion on k-cycles in S_n having a common fixed point, cf. Theorem 5.14.

Note that if f is a continuous and decreasing function on the interval $[a, b+1]$, then

$$\int_a^{b+1} f(x)\,\mathrm{d}x \leqslant \sum_{k=a}^{b} f(k). \tag{3}$$

We will also use the following useful result several times.

Lemma 5.1 *(Chernoff's bound). (See [12, Lemma 2.3.3].) Let $X_1, X_2, \ldots$ be a sequence of 0–1 valued random variables such that $\mathrm{P}(X_i = 1) \geqslant p$ for any values of the previous X_j (but X_i may depend on these X_j). Then, for all integers T and $0 < \delta < 1$,*

$$\mathrm{P}\left(\sum_{i=1}^{T} X_i \leqslant (1-\delta)pT\right) \leqslant e^{-\delta^2 pT/2}.$$

5.1. Small support involutions

The aim of this section is to compute the proportion of even-order elements in A_n and S_n which power to an involution with small support. These involutions are used in the algorithm to construct 3-cycles (cf. Algorithm 4.1 and Corollary 5.10). To achieve this, we compute lower bounds for the proportion $u_b(n)$ of elements in S_n and the proportion $\widetilde{u}_b(n)$ of elements in A_n which contain jb points in cycles of lengths divisible by b but not by $2b$ and the remaining $(n - jb)$ points in cycles of length not divisible by b for some integer j satisfying $1 \leqslant j \leqslant 4\sqrt{n}/(3b)$. To obtain involutions, we choose b to be a certain power of two.

Let $t_b(bn)$ denote the proportion of all permutations in S_{bn} such that all cycle lengths are a multiple of b but no cycle length is a multiple of $2b$. Define $t_b(0) := 1$. Observe that $t_b(b) = 1/b$, since the only allowable permutations are the b-cycles and the proportion of b-cycles in S_b is $1/b$. The proof of the following lemma refines the ideas in [10] to obtain the explicit lower bound given below.

Lemma 5.2. *Let $n, b \in \mathbb{N}$. Then $t_b(bn) \geqslant (b^2 3^{1/(2b)} n^{1-1/(2b)})^{-1}$.*

Proof. The proof is by induction on n. For $n = 1$ we have $t_b(b) = 1/b$ and the claim holds. Consider $t_b((n+1)b)$. If 1 lies in a cycle of length jb, then j has to be odd. Choosing $jb-1$ out of $(n+1)b-1$ points and arranging them yields $\frac{((n+1)b-1)!}{((n-j+1)b)!}$ such cycles. On the remaining $(n+1-j)b$ points we may choose any permutation whose cycles have lengths divisible by b but not by $2b$. We obtain the recursion

$$\big((n+1)b\big)! \cdot t_b\big((n+1)b\big) = \sum_{\substack{j=1 \\ j \text{ odd}}}^{n+1} \big((n+1)b-1\big)! \cdot t_b\big((n+1-j)b\big),$$

and thus

$$(n+1)b \cdot t_b\big((n+1)b\big) = \sum_{\substack{j=1 \\ j \text{ odd}}}^{n+1} t_b\big((n+1-j)b\big).$$

Let us first assume that n is even. The induction hypothesis yields

$$
\begin{aligned}
(n+1)b \cdot t_b\big((n+1)b\big) &\geqslant 1 + \sum_{\substack{j=1 \\ j \text{ odd}}}^{n-1} \frac{1}{b^2 3^{1/(2b)} (n+1-j)^{1-1/(2b)}} \\
&= 1 + \sum_{k=1}^{n/2} \frac{1}{b^2 3^{1/(2b)} (2k)^{1-1/(2b)}} \\
&\geqslant 1 + \left(\frac{2}{3}\right)^{1/(2b)} \int_1^{n/2+1} \frac{1}{b^2 2x^{1-1/(2b)}}\,\mathrm{d}x \\
&= 1 + \left(\frac{2}{3}\right)^{1/(2b)} \frac{1}{b} \big(x^{1/(2b)}\big|_{x=1}^{n/2+1}\big) \\
&\geqslant \frac{1}{b 3^{1/(2b)}} \big((n+2)^{1/(2b)} - 2^{1/(2b)} + b 3^{1/(2b)}\big) \\
&\geqslant \frac{1}{b 3^{1/(2b)}} (n+1)^{1/(2b)}.
\end{aligned}
$$

A similar estimation holds for odd n, using $t_b(b) = 1/b$; in either case we see

$$
t_b\big((n+1)b\big) \geqslant \frac{1}{b^2 3^{1/(2b)}} (n+1)^{1/(2b)-1},
$$

so the result follows by induction. □

Lemma 5.3. *Let* $f(x) := ((n/b - x)^{1/b} x^{1-1/(2b)})^{-1}$, *where* $n \geqslant 404$ *and* $1 \leqslant b \leqslant 4\sqrt{n}/3$. *Then* f *is positive and decreasing for* $0 < x \leqslant 4\sqrt{n}/(3b) + 1$.

Proof. Clearly f is positive on the given interval. Moreover,

$$
\frac{\mathrm{d}}{\mathrm{d}x} f(x) = \frac{bx + n - 2bn + 2b^2 x}{2(n - bx)bx} f(x)
$$

and $bx + n - 2bn + 2b^2 x < 0$ for $x \leqslant 4\sqrt{n}/(3b) + 1$, which proves the claim. □

Let $\mathrm{s}_{\neg b}(n)$ denote the proportion of elements in S_n with no cycle of length a multiple of b. Applying the inequality from [2, Theorem 2.3(b)] we get

$$
\mathrm{s}_{\neg b}(n) \geqslant \frac{b^{1/b}}{\Gamma(1 - 1/b) n^{1/b}} \left(1 - \frac{1}{n}\right), \tag{4}
$$

where Γ denotes the Γ-function. Now we are in a position to prove the following lemma which is essential for the proof of Theorem 1.2.

Lemma 5.4. *Let* $404 \leqslant n \in \mathbb{N}$. *Define* $b := 2^{\lceil \log_2(\frac{1}{3}\log(n)) \rceil}$. *Then* $u_b(n) \geqslant 1/(16\log(n))$ *and* $u_{2b}(n) \geqslant 1/(21\log(n))$.

Proof. Clearly

$$u_b(n) = \sum_{j=1}^{\lfloor \frac{4\sqrt{n}}{3b} \rfloor} s_{\neg b}(n - jb) \cdot t_b(jb).$$

Set $c(b) := \Gamma(1 - 1/b)^{-1}(1 - (404 - \frac{4}{3}\sqrt{404})^{-1})$; then $s_{\neg b}(n - jb) \geqslant c(b) \cdot (b/(n - jb))^{1/b}$. Together with Lemmas 5.2 and 5.3 we obtain

$$\begin{aligned}
u_b(n) &\geqslant \frac{c(b)}{3^{1/(2b)} b^2} \sum_{j=1}^{\lfloor \frac{4\sqrt{n}}{3b} \rfloor} \frac{1}{(n/b - j)^{1/b}} \frac{1}{j^{1-1/(2b)}} \\
&\geqslant \frac{c(b)}{3^{1/(2b)} b^2} \int_1^{\lfloor \frac{4\sqrt{n}}{3b} \rfloor + 1} \frac{1}{(n/b)^{1/b}} \frac{1}{j^{1-1/(2b)}} \,\mathrm{d}j \\
&\geqslant \frac{c(b)}{3^{1/(2b)} b^{2-1/b} n^{1/b}} \int_1^{\frac{4\sqrt{n}}{3b}} j^{1/(2b)-1} \,\mathrm{d}j \\
&= \frac{2c(b)}{3^{1/(2b)} b^{1-1/b} n^{1/b}} j^{1/(2b)} \Big|_{j=1}^{\frac{4\sqrt{n}}{3b}} \\
&> \frac{2c(b)}{bn^{1/b}} \left(\left(\frac{4\sqrt{n}}{3b} \right)^{1/(2b)} - 1 \right).
\end{aligned}$$

By definition, $b = 2^{\lceil \log_2(\frac{1}{3}\log(n)) \rceil}$, thus $\frac{1}{3}\log(n) \leqslant b < \frac{2}{3}\log(n)$. Note that $\frac{1}{3}\log(n) > 2$ for $n \geqslant 404$ implies $b \geqslant 4$. Moreover, $1/(bn^{1/b})$ is increasing in b for $0 < b < \log(n)$, and Γ is decreasing on the interval $(0, 1)$, so $c(b)$ is increasing for $b > 1$. Lastly $(4\sqrt{n}/(3b))^{1/(2b)} - 1$ is decreasing in b for $0 < b \leqslant 4\sqrt{n}/3$. Altogether we obtain

$$\begin{aligned}
u_b(n) &\geqslant \frac{2c(4)}{bn^{1/b}} \left(\left(\frac{4\sqrt{n}}{3b} \right)^{1/(2b)} - 1 \right) \\
&\geqslant \frac{6c(4)}{\log(n) n^{3/\log(n)}} \left(\left(\frac{2\sqrt{n}}{\log(n)} \right)^{3/(4\log(n))} - 1 \right).
\end{aligned}$$

Since $(2\sqrt{n}/\log(n))^{3/(4\log(n))} - 1$ is increasing on the interval $[404, \infty)$ and $n^{(3/\log(n))} = e^3$, this yields

$$u_b(n) \geqslant \frac{6c(4)}{e^3 \log(n)} \left(\left(\frac{2\sqrt{404}}{\log(404)} \right)^{3/(4\log(404))} - 1 \right) \geqslant \frac{1}{16\log(n)}.$$

A similar argument establishes the bound for $u_{2b}(n)$. □

Lemma 5.5. *For all $b, n \in \mathbb{N}$,*

$$\widetilde{u}_b(n) \geqslant \left(1 - \frac{1}{b-1}\right) u_b(n).$$

Proof. Denote by $a_{\neg b}(n)$ the proportion of elements in A_n with no cycle of length a multiple of b, and by $c_{\neg b}(n) = 2s_{\neg b}(n) - a_{\neg b}(n)$ the proportion of such elements in $S_n - A_n$. Every element in S_{jb} can be supplemented with an element of A_{n-jb} or $S_{n-jb} - A_{n-jb}$ to get an element of A_n, hence

$$\widetilde{u}_b(n) \geqslant \sum_{j=1}^{\lfloor \frac{4\sqrt{n}}{3b} \rfloor} \min\{a_{\neg b}(n-jb), c_{\neg b}(n-jb)\} \cdot t_b(jb).$$

Using the bounds $(1 - 1/(b-1))s_{\neg b}(n) \leqslant a_{\neg b}(n) \leqslant (1 + 1/(b-1))s_{\neg b}(n)$ from [2, Theorem 3.3(b)] we get $c_{\neg b}(n) \geqslant (1 - 1/(b-1))s_{\neg b}(n)$, which yields the result. □

Before proving Theorem 1.2, we state the following immediate corollary.

Corollary 5.6. *Let $9 \leqslant n \in \mathbb{N}$, $G \in \{A_n, S_n\}$ and $T := \lceil 13 \log n \log \varepsilon^{-1} \rceil$. The probability that among T random elements of G there is an element x of even order satisfying $|\operatorname{supp} x^{(|x|/2)}| \leqslant \lfloor 4\sqrt{n}/3 \rfloor$ is at least $1 - \varepsilon$.*

Proof of Theorem 1.2. The proportion in S_n equals $\sum_{b \in B_n} u_b(n)$ and in A_n it equals $\sum_{b \in B_n} \widetilde{u}_b(n)$, where $B_n := \{2^t : \ 1 \leqslant t \leqslant \lfloor \log_2(\lfloor 4\sqrt{n}/3 \rfloor) \rfloor\}$. First, let $n \geqslant 404$ and $b_0 := 2^{\lceil \log_2(\frac{1}{3}\log(n)) \rceil}$. Then Lemmas 5.4 and 5.5 yield

$$\sum_{b \in B_n} u_b(n) \geqslant \sum_{b \in B_n} \widetilde{u}_b(n) \geqslant \widetilde{u}_{b_0}(n) + \widetilde{u}_{2b_0}(n) \geqslant \frac{1}{13 \log(n)}.$$

For $36 \leqslant n \leqslant 403$ we can check

$$\sum_{b \in B_n} \widetilde{u}_b(n) \geqslant \sum_{b \in B_n} \left(1 - \frac{1}{b-1}\right) \sum_{j=1}^{\lfloor \frac{4\sqrt{n}}{3b} \rfloor} s_{\neg b}(n-jb) \cdot t_b(jb) \geqslant \frac{1}{13 \log(n)}$$

case by case, using the bounds in Lemma 5.2 and (4). Lastly, note that the desired property depends only on the cycle type. For $9 \leqslant n \leqslant 35$, we confirm the claim by investigating each conjugacy class of S_n and A_n and thus directly computing the exact proportion. □

5.2. Products of k-involutions

We call a product of k disjoint transpositions a k-involution. Our method to construct a 3-cycle uses the product of two random k-involutions r and s such that $\operatorname{supp}(r) \cap \operatorname{supp}(s)$ contains a single element. Since we are in a black-box setting, given an involution r we know neither k nor $\operatorname{supp}(r)$ explicitly. However, if k is small enough, then a random conjugate of r which does not commute with r satisfies our hypothesis with high probability, cf. Theorem 1.3. Furthermore, there are enough non-commuting conjugates of r. Note that we can find involutions with small k by Theorem 1.2.

First, we need some auxiliary lemmas.

Lemma 5.7. *Let $f(k) := (1 - 2k/(9k^2/4 - 2k + 1))^{2k}$. Then $f(k)$ is increasing for $k \geqslant 2$.*

Proof. Let $g(k) := \frac{9}{4}k^2 - 2k + 1$. The derivative of $f(k)$ is

$$\left(2\log\left(1 - \frac{2k}{g(k)}\right) + \frac{2k}{g(k)^2} \frac{g(k)}{\frac{9}{4}k^2 - 4k + 1}\left(-2g(k) + 2k\left(\frac{9}{2}k - 2\right)\right)\right) f(k).$$

Thus, using $\log(1+x) \geqslant x/(1+x)$, we find

$$\frac{\mathrm{d}}{\mathrm{d}k} f(k) \geqslant \left(\frac{-4k}{\frac{9}{4}k^2 - 4k + 1} + \frac{2k}{(\frac{9}{4}k^2 - 2k + 1)(\frac{9}{4}k^2 - 4k + 1)} \left(\frac{9}{2}k^2 - 2 \right) \right) f(k)$$

and for $k \geqslant 2$ it is easy to check that both factors are positive. □

Let $s \in \mathrm{S}_n$ be a fixed k-involution. Denote by $\mathrm{inv}(n,k)$ the number of k-involutions in S_n. Then

$$\mathrm{inv}(n,k) = \frac{|\mathrm{S}_n|}{|\mathrm{C}_{\mathrm{S}_n}(s)|} = \frac{n!}{2^k k!(n-2k)!}.$$

Let $\mathrm{trip}(n,k)$ denote the proportion of k-involutions $r \in \mathrm{S}_n$ such that r and s move a single common point.

Note that if k is even, then $\mathrm{inv}(n,k)$ is also the number of k-involutions in A_n, and $\mathrm{trip}(n,k)$ equals the proportion of k-involutions $r \in \mathrm{A}_n$ such that $|\mathrm{supp}(r) \cap \mathrm{supp}(s)| = 1$. Thus for the results in this section it does not matter whether we consider the alternating or the symmetric group.

Lemma 5.8. *Let* $9 \leqslant n \in \mathbb{N}$ *and* $1 \leqslant k \leqslant 2\sqrt{n}/3$. *Then* $\mathrm{trip}(n,k) \geqslant \min\{\mathrm{trip}(n,1), \mathrm{trip}(n, \lfloor 2\sqrt{n}/3 \rfloor)\}$.

Proof. We have

$$\mathrm{trip}(n,k) = \frac{2k(n-2k)\,\mathrm{inv}(n-2k-1,k-1)}{\mathrm{inv}(n,k)} = \frac{4k^2(n-2k)!^2}{n!(n-4k+1)!}.$$

It suffices to show that $\mathrm{trip}(n,k+1)/\mathrm{trip}(n,k)$ is decreasing in k. To see this, consider the derivative of the quotient. We find

$$\frac{\mathrm{d}}{\mathrm{d}k} \frac{\mathrm{trip}(n,k+1)}{\mathrm{trip}(n,k)} = \alpha(n,k) \frac{-2(k+1)}{((n-2k-1)(n-2k)k)^3}$$

for some polynomial $\alpha(n,k) \in \mathbb{Z}[n,k]$. Since $(n-2k-1) > 0$ holds for $n \geqslant 9$ and $k \leqslant 2\sqrt{n}/3$, we only need to show that $\alpha(n,k) \geqslant 0$. Write $\alpha = \alpha_+ + \alpha_-$ such that $\alpha_+(n,k) \in \mathbb{Z}_{>0}[n,k]$ and $\alpha_-(n,k) \in \mathbb{Z}_{<0}[n,k]$. Since $1 \leqslant k \leqslant 2\sqrt{n}/3$, we obtain

$$\alpha(n,k) \geqslant \alpha_+(n,1) + \alpha_-(n, 2\sqrt{n}/3) =: \beta(\sqrt{n}) \in \mathbb{Q}[\sqrt{n}].$$

Using Sturm sequences (cf. [6, Theorem 4.1.10]), it is easy to see that β has no roots for $\sqrt{n} \geqslant 28$, so $\alpha(n,k) \geqslant \beta(\sqrt{n}) \geqslant 0$. Thus, the claim holds for $n \geqslant 28^2$. For $9 \leqslant n \leqslant 28^2 - 1$ and $1 \leqslant k \leqslant \lfloor 2\sqrt{n}/3 \rfloor$, we check the claim case by case. □

Using this result we can now prove the first claim of Theorem 1.3.

Proof of Theorem 1.3(1). By Lemma 5.8 it suffices to check the inequality for $k = 1$ and $k = \lfloor 2\sqrt{n}/3 \rfloor$. The first case is easy to verify, so consider the second case. Note that

$$\frac{\mathrm{trip}(n,k)}{\mathrm{trip}(n+1,k)} = \frac{(n+1)(n-4k+2)}{(n-2k+1)^2} = 1 + \frac{n-(4k^2-1)}{(n-2k+1)^2},$$

so $\mathrm{trip}(n,k)$ increases in n for $n \leqslant 4k^2 - 1$, which holds for $n \geqslant 39$. We consider this case first. Since $n \geqslant \lceil 9k^2/4 \rceil$, we see

$$\begin{aligned}\operatorname{trip}(n,k) &\geqslant \operatorname{trip}\left(\left\lceil \frac{9}{4}k^2 \right\rceil, k\right)\\ &= \frac{4k^2}{(\lceil \frac{9}{4}k^2\rceil - 4k + 1)} \prod_{i=1}^{2k} \frac{\lceil \frac{9}{4}k^2\rceil - 4k + i}{\lceil \frac{9}{4}k^2\rceil - 2k + i}\\ &\geqslant \frac{4k^2}{(\frac{9}{4}k^2)} \prod_{i=1}^{2k}\left(1 - \frac{2k}{\frac{9}{4}k^2 - 2k + 1}\right) = \frac{16}{9}\left(1 - \frac{2k}{\frac{9}{4}k^2 - 2k + 1}\right)^{2k}.\end{aligned}$$

The claim follows by Lemma 5.7, since $k \geqslant 4$. For $10 \leqslant n \leqslant 38$ we check $\operatorname{trip}(n, \lfloor 2\sqrt{n}/3 \rfloor) \geqslant 10/(3n)$ case by case. Finally, for $n \leqslant 9$ we compute the proportion explicitly. □

Theorem 1.3(1) shows that we can construct a 3-cycle by looking at $\mathcal{O}(n)$ conjugates of an involution with small support. Unfortunately, considering that many conjugates would result in a final algorithm with complexity $\widetilde{\mathcal{O}}(n^2)$. Thus we do not use this result to construct the 3-cycles directly, but instead use it as a lower bound for the proportion of non-commuting conjugates.

Corollary 5.9. *Let $9 \leqslant n \in \mathbb{N}$, $1 \leqslant k \leqslant 2\sqrt{n}/3$, $0 < \varepsilon < 1$, $G \in \{A_n, S_n\}$ and $s \in G$ be a k-involution. Let $Z := \lceil \frac{3n}{5} \lceil 3 \log \varepsilon^{-1} \rceil \rceil$. Then, with probability at least $1 - \varepsilon$, a set of Z random conjugates of s contains at least $\lceil 3 \log \varepsilon^{-1} \rceil$ elements not commuting with s.*

Proof. Use the proportion established in Theorem 1.3(1) and Chernoff's bound (Lemma 5.1) with $\delta := 1/2$. □

Next we prove the second part of Theorem 1.3 by establishing a bound for the conditional probability that two k-involutions s and r satisfy $|\operatorname{supp} r \cap \operatorname{supp} s| = 1$, given that they do not commute. Note that in this case $(sr)^2$ is a 3-cycle, so we immediately obtain the following corollary.

Corollary 5.10. *Let $9 \leqslant n \in \mathbb{N}$, $1 \leqslant k \leqslant 2\sqrt{n}/3$, $0 < \varepsilon < 1$, $G \in \{A_n, S_n\}$ and $s \in G$ be a k-involution. Let $Z := \lceil 3 \log \varepsilon^{-1} \rceil$. Then, with probability at least $1 - \varepsilon$, a set of Z random conjugates of s not commuting with s contains an element r such that $(sr)^2$ is a 3-cycle.*

Proof of Theorem 1.3(2). Let s be a fixed k-involution and denote by Σ the proportion of k-involutions r such that $(sr)^2$ is a 3-cycle among all k-involutions not commuting with s. The proportion Σ can be computed explicitly for $n \leqslant 9$, so assume in the following that $n \geqslant 10$. Let $T := \{t \in s^{S_n} : |\operatorname{supp} t \cap \operatorname{supp} s| = 1\}$ and $C := \{c \in s^{S_n} : |\operatorname{supp} c \cap \operatorname{supp} s| = 0\}$. Then $(st)^2$ is a 3-cycle for every $t \in T$ and $[s, c] = 1_G$ for every $c \in C$. We find $|T| = 2k(n-2k)\operatorname{inv}(n-2k-1, k-1)$ and $|C| = \operatorname{inv}(n-2k, k)$, so the conditional probability Σ is bounded below by

$$\frac{|T|}{\operatorname{inv}(n,k) - |C|} = \frac{4k^2(n-2k)!^2}{(n-4k+1)(n!(n-4k)! - (n-2k)!^2)}.$$

This term is greater or equal to $1/3$ if and only if

$$\left(1 + \frac{12k^2}{n-4k+1}\right) \prod_{i=1}^{2k} \frac{n-4k+i}{n-2k+i} \geqslant 1. \tag{5}$$

Define $g(n,k) := (1+12k^2/(n-4k+1))(1-2k/(n-2k+1))^{2k}$; the claim follows if $g(n,k) \geqslant 1$. For this purpose, consider the derivative

$$\frac{\mathrm{d}}{\mathrm{d}n} g(n,k) = \frac{8k^2(-n+6k^2+k-1)}{(n-2k+1)(n-4k+1)^2}\left(1-\frac{2k}{n-2k+1}\right)^{2k}.$$

Note that $k \leqslant 2\sqrt{n}/3$ by assumption and hence $n \geqslant 9k^2/4$.

Assume first $n \geqslant 6k^2+k-1$. Then $\frac{\mathrm{d}}{\mathrm{d}n} g(n,k) \leqslant 0$, and $\lim_{n\to\infty} g(n,k) = 1$ implies $g(n,k) \geqslant 1$. Now assume $9k^2/4 \leqslant n < 6k^2+k-1$ and $k \geqslant 36$. Then $\frac{\mathrm{d}}{\mathrm{d}n} g(n,k) > 0$, hence

$$\begin{aligned} g(n,k) &\geqslant g\big(9k^2/4,k\big) \\ &= \left(1+\frac{12k^2}{9k^2/4-4k+1}\right)\left(1-\frac{2k}{9k^2/4-2k+1}\right)^{2k} =: h(k)f(k). \end{aligned}$$

Since $h(k) \geqslant 57/9$ and $f(k)$ increases for $k \geqslant 2$ by Lemma 5.7, we get $g(n,k) \geqslant 57/9 \cdot f(36) > 1$.

Finally, for $9k^2/4 \leqslant n < 6k^2+k-1$ and $1 \leqslant k \leqslant 35$ we verify inequality (5) case by case. □

5.3. Pre-bolstering elements

Let $G = \mathrm{S}_n$ or $G = \mathrm{A}_n$, and let $c \in G$ be a 3-cycle. In the algorithm, we use pre-bolstering elements to construct a long cycle matching c. Recall that an element r is pre-bolstering with respect to c if

$$r = (w, u, a_1, \ldots, a_\alpha)(v, b_1, \ldots, b_\beta)(\ldots)$$

or

$$r = (w, u, a_1, \ldots, a_\alpha, v, b_1, \ldots, b_\beta)(\ldots)$$

with $\operatorname{supp} c = \{u, v, w\}$ and $\alpha, \beta \geqslant 2$. If $k = \alpha+\beta+3$, we call the element k-pre-bolstering. Note that $k \geqslant 7$. Denote by $L_{c,G}(k)$ the number of k-pre-bolstering elements of G with respect to c.

Lemma 5.11. *Let $7 \leqslant k \leqslant n \in \mathbb{N}$ and $G \in \{\mathrm{A}_n, \mathrm{S}_n\}$. Then we have $L_{c,\mathrm{S}_n}(k) = 12(n-3)!(k-6)$ and $L_{c,\mathrm{A}_n}(k) = 6(n-3)!(k-6)$. Moreover,*

$$\frac{1}{|G|}\sum_{k=7}^{n} L_{c,G}(k) \geqslant \frac{2}{5n}.$$

Proof. A standard counting argument yields the formulae for $L_{c,G}(k)$. Thus, for $G \in \{\mathrm{A}_n, \mathrm{S}_n\}$, we obtain

$$\begin{aligned} \frac{1}{|G|}\sum_{k=7}^{n} L_{c,G}(k) &= \frac{6}{n}\left(1-\frac{8n-28}{(n-1)(n-2)}\right) \\ &\geqslant \frac{6}{n}\left(1-\frac{8\cdot 7-28}{(7-1)(7-2)}\right) = \frac{2}{5n}. \end{aligned}$$ □

Using Chernoff's bound, we obtain a terminating condition for Algorithm BolsteringElements.

Proposition 5.12. *Let $7 \leqslant n \in \mathbb{N}$, $G \in \{A_n, S_n\}$, $c \in G$ be a 3-cycle, $0 < \varepsilon < 1$ and $1/2 < \alpha \leqslant 4/5$. Let $S = \lceil 5n \max(\frac{25}{18}\lceil \frac{1}{2}\log_\alpha \varepsilon\rceil, (5/4)^4 \log \varepsilon^{-1})\rceil$. The probability that among S random elements at least $\lceil \frac{1}{2}\log_\alpha \varepsilon\rceil$ are k-pre-bolstering with respect to c for some $7 \leqslant k \leqslant n$ is at least $1-\varepsilon$.*

Proof. Use Lemma 5.11 and Chernoff's bound with $\delta := 16/25$. □

The next proposition establishes the second bound: a lower bound on the proportion of k-pre-bolstering elements in G with $\alpha n \leqslant k \leqslant n$ among the k-pre-bolstering elements with $7 \leqslant k \leqslant n$. This ensures that ConstructLongCycle constructs long cycles with high probability.

Proposition 5.13. *Let $9 \leqslant n \in \mathbb{N}$, $G \in \{A_n, S_n\}$, $c = (c_1, c_2, c_3) \in G$ be a 3-cycle, $0 < \varepsilon < 1$ and $3/4 \leqslant \alpha \leqslant 4/5$. Let $R = \lceil \frac{1}{2}\log_\alpha \varepsilon\rceil$ and $r_1, \ldots, r_R \in G$ random elements such that r_i is k_i-pre-bolstering with respect to c. The probability that there is at least one k_j with $k_j \geqslant \max(9, \lceil \alpha n\rceil + 1)$ is at least $1-\varepsilon$.*

Proof. We want to show that the proportion of k_i-pre-bolstering elements with $k_i \geqslant \lceil \alpha n\rceil + 1$ among all pre-bolstering elements is at least $1-\alpha^2$. For $n = 9$ we verify the claim directly, so assume in the following $n \geqslant 10$. Then $\lceil \alpha n\rceil + 1 \geqslant 9$, and we find

$$\begin{aligned}\frac{\sum_{k=\max(\lceil \alpha n\rceil+1,9)}^{n} L_{c,G}(k)}{\sum_{k=7}^{n} L_{c,G}(k)} &= \frac{(n-6)(n-5) - 2\sum_{k=1}^{\lceil \alpha n\rceil-6} k}{(n-6)(n-5)} \\ &= 1 - \frac{(\lceil \alpha n\rceil - 6)(\lceil \alpha n\rceil - 5)}{(n-6)(n-5)} \\ &> 1 - \frac{\alpha^2(n-6)(n-5)}{(n-6)(n-5)}.\end{aligned}$$

The claim now follows by a standard argument. □

5.4. Common fixed points of k-cycles

The final result ensures that we construct an n- or an $(n-1)$-cycle in StandardGenerators and thus find the correct degree of the group with high probability.

Theorem 5.14. *Let $0 < \varepsilon < 1$, $0 < \alpha < 1$ and $n, k, t \in \mathbb{N}$ with $\alpha n \leqslant k < n$ and*

$$t \geqslant \frac{1}{\log((1-\alpha)^{-1})}\left(\log n + \log \varepsilon^{-1}\right).$$

The probability that t random k-cycles in S_n have a common fixed point is at most ε.

Proof. Denote by $P_{\text{fix}}(n,k,t)$ the probability that t random k-cycles in S_n have a common fixed point. Let $r \in S_n$ be a k-cycle and $1 \leqslant m_1, \ldots, m_j \leqslant n$ pairwise different points. If r fixes each of the m_i, then the probability that another random point m_{j+1} is fixed by r equals $(n-k-j)/(n-j) = 1-k/(n-j)$. Thus, the probability that $m_1, \ldots, m_{j+1}$ are common fixed points of t random k-cycles equals

$$\prod_{i=0}^{j}\left(1 - \frac{k}{n-i}\right)^t.$$

Define $c_j := (-1)^j\binom{n}{j+1}\prod_{i=0}^{j}(1-k/(n-i))^t$ (note that $c_j = 0$ for $j \geqslant n-k$); a standard inclusion–exclusion principle shows $P_{\text{fix}}(n,k,t) = \sum_{j=0}^{n-k-1} c_j$. We will prove

$$\left|\frac{c_j}{c_{j+1}}\right| = \frac{j+2}{n-j-1}\cdot\left(1-\frac{k}{n-j-1}\right)^{-t} \geqslant 1$$

for $j+1 < n-k$. To this end, note that

$$t \geqslant \frac{1}{\log((1-\alpha)^{-1})}\log\left(\frac{n-2}{2}\right) \geqslant \frac{1}{\log((1-\alpha)^{-1})}\log\left(\frac{n-j-1}{j+2}\right)$$
$$= \frac{\log(\frac{j+2}{n-j-1})}{\log(\frac{(n-j-1)(1-\alpha)}{n-j-1})} \geqslant \frac{\log(\frac{j+2}{n-j-1})}{\log(\frac{n-j-1-k}{n-j-1})},$$

thus $t\cdot\log(1-k/(n-j-1)) \leqslant \log((j+2)/(n-j-1))$. This implies

$$\left(1-\frac{k}{n-j-1}\right)^{t} \leqslant \frac{j+2}{n-j-1}$$

and hence $|c_j| \geqslant |c_{j+1}|$.

Since c_j has alternating sign and c_0 is positive, this yields $\sum_{j=0}^{n-k-1} c_j \leqslant c_0$. Moreover,

$$t \geqslant \frac{1}{\log((1-\alpha)^{-1})}\left(\log n + \log\varepsilon^{-1}\right) = \frac{\log(\frac{\varepsilon}{n})}{\log(1-\alpha)} \geqslant \frac{\log(\frac{\varepsilon}{n})}{\log(1-\frac{k}{n})},$$

hence $\log(\varepsilon/n) \geqslant t\cdot\log(1-k/n)$. We obtain $c_0 = n(1-k/n)^t \leqslant \varepsilon$, thus proving the claim. □

Acknowledgments

We thank the anonymous referee for many helpful suggestions.

We acknowledge support from the following DFG grants: SPP 1388 (first author), Graduiertenkolleg Experimentelle und konstruktive Algebra at RWTH Aachen University (second author), SPP 1489 (third and fourth authors) and the Australian grant: ARC DP110101153 (third author).

References

[1] L. Babai, E. Szemeredi, On the complexity of matrix group problems. I, in: Proceedings of the 25th Annual Symposium on Foundations of Computer Science, SFCS '84, IEEE Computer Society, Washington, DC, USA, 1984, pp. 229–240, http://dx.doi.org/10.1109/SFCS.1984.715919.

[2] R. Beals, C.R. Leedham-Green, A.C. Niemeyer, C.E. Praeger, Á. Seress, Permutations with restricted cycle structure and an algorithmic application, Combin. Probab. Comput. 11 (5) (2002) 447–464, http://dx.doi.org/10.1017/S0963548302005217.

[3] R. Beals, C.R. Leedham-Green, A.C. Niemeyer, C.E. Praeger, Á. Seress, A black-box group algorithm for recognizing finite symmetric and alternating groups. I, Trans. Amer. Math. Soc. 355 (5) (2003) 2097–2113, http://dx.doi.org/10.1090/S0002-9947-03-03040-X.

[4] S. Bratus, I. Pak, Fast constructive recognition of a black box group isomorphic to S_n or A_n using Goldbach's conjecture, J. Symbolic Comput. 29 (1) (2000) 33–57, http://dx.doi.org/10.1006/jsco.1999.0295.

[5] R.D. Carmichael, Abstract definitions of the symmetric and alternating groups and certain other permutation groups, Q. J. Math. 49 (1923) 226–270.

[6] H. Cohen, A Course in Computational Algebraic Number Theory, Grad. Texts in Math., vol. 138, Springer-Verlag, Berlin, 1993.

[7] The GAP Group, GAP – Groups, Algorithms, and Programming, Version 4.4.12, http://www.gap-system.org, 2008.

[8] C.R. Leedham-Green, The computational matrix group project, in: Groups and Computation, III, Columbus, OH, 1999, in: Ohio State Univ. Math. Res. Inst. Publ., vol. 8, de Gruyter, Berlin, 2001, pp. 229–247.

[9] M. Neunhöffer, Á. Seress, A data structure for a uniform approach to computations with finite groups, in: ISSAC 2006, ACM, New York, 2006, pp. 254–261, http://dx.doi.org/10.1145/1145768.1145811.

[10] A.C. Niemeyer, T. Popiel, C.E. Praeger, Ş. Yalçınkaya, On semiregular permutations of a finite set, Math. Comp. 81 (277) (2012) 605–622, http://dx.doi.org/10.1090/S0025-5718-2011-02506-8.

[11] E.A. O'Brien, Algorithms for matrix groups, in: Groups St Andrews 2009 in Bath, in: London Math. Soc. Lecture Note Ser., vol. 388, 2011, pp. 297–323.

[12] Á. Seress, Permutation Group Algorithms, Cambridge Tracts in Math., Cambridge University Press, 2003, http://books.google.de/books?id=hxFqdbfc_CMC.