Algorithms for Permutation groups

Alice Niemeyer

UWA, RWTH Aachen
The Symmetric Group

Let $\Omega$ be a finite set.
The symmetric group, $\text{Sym}(\Omega)$, is the group of all bijections from $\Omega$ to itself.
A permutation group is a subgroup of $\text{Sym}(\Omega)$. 
1960s: the Classification of finite simple groups required to work with large permutation groups.

1970s: C. Sims introduced algorithms for working with permutation groups.

These were among the first algorithms in CAYLEY and GAP.

1990s: nearly linear algorithms for permutation groups emerged. These are now in GAP and MAGMA.

Seress’ book.

A very brief summary.
From now on:

Let $\Omega$ be finite and $G \leq \text{Sym}(\Omega)$.

For $\alpha \in \Omega$ let $G_\alpha$ denote the stabiliser of $\alpha$ in $G$, i.e.

$$G_\alpha = \{g \in G \mid \alpha^g = \alpha\}.$$

If $\alpha, \beta \in \Omega$ let $G_{(\alpha,\beta)}$ denote the stabiliser of $\beta$ in $G_\alpha$, i.e.

$$G_{(\alpha,\beta)} = (G_\alpha)_\beta = \{g \in G \mid \alpha^g = \alpha \text{ and } \beta^g = \beta\}.$$
Base and Stabiliser Chain

$B = (\alpha_1, \alpha_2, \ldots, \alpha_k)$ with $\alpha_i \in \Omega$ is a base for $G$ if $G(\alpha_1, \alpha_2, \ldots, \alpha_k) = \{1\}$.

The chain of subgroups

$$G = G^{(1)} \geq G^{(2)} \geq \cdots \geq G^{(k+1)} = \{1\}$$

defined by $G^{(i+1)} = G^{(i)}_{\alpha_i}$ for $1 \leq i \leq k$ is the stabiliser chain for $B$.

$B$ is irredundant if all the inclusions in the stabiliser chain for $B$ are proper.
If $G$ is a permutation group and $B = (\alpha_1, \alpha_2, \ldots, \alpha_k)$ a base for $G$, then each element $g \in G$ is uniquely determined by $(\alpha^g_1, \alpha^g_2, \ldots, \alpha^g_k)$.

(Since $B^g = B^h$ implies $B^{gh^{-1}} = B$ and thus $gh^{-1} = 1$.)
Definition

Let $G = \langle X \rangle \leq \text{Sym}(\Omega)$ and $\alpha \in \Omega$. The orbit of $\alpha$ under $G$, denoted $\alpha^G$ is the set

$$\alpha^G := \{ \alpha^g \mid g \in G \}.$$
Example

The orbits for $G = \langle x, y, z \rangle$ with

$$x = (1, 2)(3, 5, 9)(4, 6), \quad y = (1, 3, 5)(7, 8, 10), \quad z = (4, 7, 8)$$
on $\Omega = \{1, 2, \ldots, 10\}$ are $\Omega/G$ are $\Delta_1 = \{1, 2, 3, 5, 9\}$ and $\Delta_2 = \{4, 6, 7, 8, 10\}$.
Definition

Let $G \leq \text{Sym}(\Omega)$ and $(\alpha_1, \alpha_2, \ldots, \alpha_k)$ a basis for $G$. Let $G = G^{(1)} \geq G^{(2)} \geq \cdots \geq G^{(k+1)} = \{1\}$ (where $G^{(i+1)} = G^{(i)}_{\alpha_i}$ for $1 \leq i \leq k$) be the stabiliser chain for $B$. Then $S \subseteq G$ is a strong generating set for $G$ if for every $i$ with $1 \leq i \leq k + 1$ holds $G^{(i)} = \langle S \cap G^{(i)} \rangle$. 
The Schreier-Sims Algorithm

Input: $G \leq \text{Sym}(\Omega)$
Output:
- $(\alpha_1, \alpha_2, \ldots, \alpha_k)$ a basis for $G$
- $S \subseteq G$ a strong generating set for $G$
- the orbits $\alpha_i^{G(i)}$ stored in a particular way
Example

- $G := \langle (1, 2, 3, 4, 5, 6), (2, 6)(3, 5) \rangle$.
- base: $\{1, 2\}$
- strong generating set: $S = \{(2, 6)(3, 5), (1, 2, 3, 4, 5, 6), (1, 3, 5)(2, 4, 6)\}$
- stabiliser Chain:
  \[
  G^{(1)} = G \geq G^{(2)} = \langle (2, 6)(3, 5) \rangle \geq G^{(3)} = \{1\}.
  \]
- orbits:
  \[
  1^{G^{(1)}} = \Omega, \quad 2^{G^{(2)}} = \{2, 6\}.
  \]
The Schreier-Sims Algorithm

Questions

The data structure of a base and a strong generating set together with the associated stabiliser chain allows us to answer questions about $G$ such as

- what is $|G|$?
- does $g \in \text{Sym}(\Omega)$ satisfy $g \in G$?
Example: Is \( g = (1, 4)(2, 3)(5, 6) \in G? \)

\( G := \langle (1, 2, 3, 4, 5, 6), (2, 6)(3, 5) \rangle. \)

- base: \( \{1, 2\} \)
- strong generating set: \( S = \{(2, 6)(3, 5), (1, 2, 3, 4, 5, 6), (1, 3, 5)(2, 4, 6)\} \)
- stabiliser Chain:
  \[ G^{(1)} = G \geq G^{(2)} = \langle (2, 6)(3, 5) \rangle \geq G^{(3)} = \{1\}. \]
- orbits:
  \[ 1^{G^{(1)}} = \Omega, \ 2^{G^{(2)}} = \{2, 6\}. \]
Example: Is \( g = (1, 4)(2, 3)(5, 6) \in G? \)

\[
G := \langle (1, 2, 3, 4, 5, 6), (2, 6)(3, 5) \rangle.
\]

- \( 1^g = 4 \)
- Find \( h \in G \) with \( 1^h = 4 \).
- \( h = (1, 4)(2, 5)(3, 6) \in G \).
- \( g \in G \) if and only if \( gh^{-1} \in G \).
- \( 1^{gh^{-1}} = 1 \) so \( g \in G \) if and only if \( gh^{-1} \in G^{(2)} \).
- \( gh^{-1} = (2, 6)(3, 5) \).
- \( (2, 6)(3, 5) \in S \), so \( gh^{-1} \in G^{(2)} \).
- Thus \( g \in G \).
Schreier’s Lemma

**SCHREIER’S LEMMA**

Let $G = \langle X \rangle$ be a finite group, $H \leq G$ and $T$ set of representatives of the right cosets of $H$ in $G$ such that $T$ contains 1. Denote by $\bar{g}$ the representative of $Hg$ for $g \in G$. Then $H$ is generated by

$$X_H = \{ tx(\bar{tx})^{-1} | t \in T, x \in X \}.$$
The Schreier-Sims Algorithm

Essential steps

- Compute the orbits $\alpha_i^{G(i)}$ together with $T_i$
- $T_i$ set of cosets representatives for cosets of $G^{(i+1)}$ in $G^{(i)}$
- for $\beta \in \alpha_i^{G(i)}$ find representative in $T_i$
- find generators for $G^{(i+1)}$. 
Theorem

Let $\Omega$ finite, $n = |\Omega|$ and $G = \langle X \rangle \leq \text{Sym}(\Omega)$ a permutation group. Then the complexity of the Schreier-Sims algorithm is

$$O(n^3 \log_2(|G|)^3 + |X|n^3 \log_2(|G|)).$$

Note that $|\text{Sym}(\Omega)| = n! \sim n^n$, so $\log(|\text{Sym}(\Omega)|) \sim n \log(n)$. Therefore, the complexity can be as bad as

$$O(n^6 + |X|n^4).$$
A Remark about $|B|$ 

Given a basis $B$ for $G = \langle X \rangle \leq \text{Sym}(\Omega)$, with $\Omega$ finite. Then $2^{|B|} \leq |G| \leq n^{|B|}$ or 

$$
\frac{\log(|G|)}{\log(n)} \leq |B| \leq \frac{\log(|G|)}{\log(2)}.
$$
Let $\mathcal{G}$ be a family of permutation groups. We call $\mathcal{G}$ small-base if for every $G \in \mathcal{G}$ of degree $n$ holds

$$\log |G| < \log^c(n)$$

for a constant $c$, fixed for $\mathcal{G}$. 
Theorem of Liebeck

Theorem

Let $G$ be a family of permutation groups. Every large-base primitive group in $G$ of degree $n$ involves the action of $A_n$ or $S_n$ on the set of $k$-element subsets of $\{1, \ldots, n\}$, for some $n$ and $k < n/2$.

These groups are called the giants.
Remark

Let $\mathcal{G}$ be a family of small-base permutation groups, i.e. for every $G \in \mathcal{G}$ of degree $n$ holds

$$\log |G| < \log^c(n)$$

for a constant $c$, fixed for $\mathcal{G}$. Then complexity of the Schreier-Sims algorithm is

$$O(n^3 \log_2(|G|)^3 + |X| n^3 \log_2(|G|)).$$
Remark

Let $\mathcal{G}$ be a family of small-base permutation groups, i.e. for every $G \in \mathcal{G}$ of degree $n$ holds

$$\log |G| < \log^c(n)$$

for a constant $c$, fixed for $\mathcal{G}$. Then complexity of the Schreier-Sims algorithm is

$$O(n^3 \log^c(n)^3 + |X| n^3 \log^c(n)).$$

This is only slightly more expensive than $O(n^3)$. If we can limit the length of the basis by $n$, the complexity is $O(n^6)$. 

Seress proves in his book (p. 75, Theorem 4.5.5):

**Theorem**

Let $G \leq \langle X \rangle \leq \text{Sym}(\Omega)$ with $|\Omega| = n$. Then there exists a Monte-Carlo algorithm, which computes with probability $\varepsilon$ for $\varepsilon \leq \frac{1}{n^d}$ (for a positive whole number $d$, given by the user) a basis and a strong generating system for $G$ in time

$$O(n \log(n) \log(|G|)^4 + |X|n \log(|G|))$$

and uses $O(n \log(|G|) + |X|n)$ space.

For small-base groups this algorithm is nearly linear.
Schreier-Sims for Matrix Groups

One of the first approaches to deal with Matrix Groups (Butler, 1979).
Let $G \leq \text{GL}(n, q)$. Then $G$ acts faithfully as a permutation group on $V = \mathbb{F}_q^n$ via $g : v \mapsto vg$.
Thus we can apply the Schreier-Sims algorithm to this permutation group.
Schreier-Sims for Matrix Groups

Problem

How long can the orbit $v^G$ be? It can be $q^n - 1$.

Example

$q = 3$, $n = 100$

$q^n - 1$

$= 515377520732011331036461129765621272702107522000$

Even

$3^{20} - 1 = 348678440$. 
Schreier-Sims for Matrix Groups

Problem

- In a permutation group $G \leq S_n$ the length of an orbit is at most $n$. Hence easy to compute an orbit for $n$ quite large.
- In a matrix group $G \leq \text{GL}(n, q)$ orbits can be $O(q^n)$.

Complexity

- $S_n$ linear in $n$.
- $\text{GL}(n, q)$ exponential in $n$. 
Schreier-Sims for Matrix Groups

- works well for small $n$ and $q$.
- Algorithms developed by Butler (1979)
- Murray & O’Brien (1995) consider the selection of base points
How can we rule out that our given group is a giant beforehand?

Consider first $A_n$ and $S_n$ in their natural representation.
Definition

An element \( g \in S_n \) is called *purple* if it contains in its disjoint cycle decomposition one cycle of prime length \( p \) with \( n/2 < p < n - 2 \).
Theorem (Modification of a theorem of Jordan, 1873)

$G \leq S_n$ acts transitively on $\Omega = \{1, \ldots, n\}$. If $G$ contains a purple element, then $G$ contains $A_n$. 
**Theorem**

*Let* $p$ *a prime with* $n/2 < p < n - 2$. *The proportion of purple elements in* $S_n$ *and* $A_n$ *is* $\frac{1}{p}$.\*</p>
The following Theorem was already conjectured by Bertrand (1822-1900) and proved by Chebyshev (1821-1894) in 1850.

**Theorem**

*For a positive integer $m$ with $m > 3$ there exists at least one prime $p$ with $m < p < 2m - 2$.***
Proportions in $S_n$ and $A_n$

The proportion of purple elements in $S_n$ or $A_n$ is $\frac{c}{\log(n)}$ for a small constant $c$. 
Monte-Carlo Test: is $A_n \leq G$?

Algorithm 1: \textsc{Contains} $A_n$

\begin{itemize}
  \item \textbf{Eingabe:} $G = \langle X \rangle \leq S_n$
  \item \textbf{Ausgabe:} true or false
  \item if not \textsc{IsTransitive}(G) then return false;
  \item for $i = 1 \ldots N$ do
    \begin{itemize}
      \item $g := \text{Random}(G)$;
      \item if $g$ purple then return true;
    \end{itemize}
  \item return false;
\end{itemize}
The probability that among $N$ independent, uniformly distributed random elements $g \in G$, with $A_n \leq G$, no purple elements were found is $(1 - \frac{c}{\log(n)})^N$. Thus choose $N$ such that $(1 - \frac{c}{\log(n)})^N < \varepsilon$, or

$$N > \log(\varepsilon^{-1}) \log \left( \frac{\log(n)}{\log(n) - c} \right)^{-1}.$$ 

This is the case, if

$$N > \log(\varepsilon^{-1}) \frac{\log(n)}{c}.$$ 

Thus the complexity is

$$O(\log(\varepsilon^{-1}) \log(n)(\rho + n)),$$ 

where $\rho$ is the cost of a call to RANDOM.
Problems with no known polynomial time Algorithms

Consider the following problems for permutation groups.

- set stabiliser
- centraliser of one group in another
- intersection of permutation groups
- decide whether two elements in a group are conjugate
Ákos Seress
Permutation Group Algorithms,
Cambridge Tracts in Mathematics 152,