Basic Algorithms for Permutation Groups

Alexander Hulpke

Department of Mathematics
Colorado State University
Fort Collins, CO, 80523-1874

Arizona Summer Program 2008

Ground rules

Storing all group elements is often infeasible and inefficient. Instead a group is stored by an (arbitrary) set of generators.

Describing a subgroup means finding generators for it.

Some basic tasks needed for groups given by generators thus are:

- An ability to enumerate (or run through) all elements.
- Test whether an element is in the group (without enumerating all elements).
- For elements in the group, express them as words in the generators (homomorphisms!).

We want to see how this works for permutation groups.

Group actions

A group G acts (from the right) on a set Ω if

- $\omega^1 = \omega$ for all $\omega \in \Omega$
- $(\omega^g)^h = \omega^{gh}$ for all $\omega \in \Omega$, $g, h \in G$.

In this case we define for $\omega \in \Omega$ the Orbit $\omega^G = \{\omega^g \mid g \in G\} \subset \Omega$ and the Stabilizer $\mathrm{Stab}_G(\omega) = \{g \in G \mid \omega^g = \omega\} \leq G$.

Lemma

There is a bijection between ω^G and the set $Stab_G(\omega)\backslash G$ (i.e. right cosets of $Stab_G(\omega)$ in G), given by

$$\omega^g \leftrightarrow Stab_G(\omega) \cdot g$$

In particular $\left|\omega^G\right| = \left[G{:}Stab_G(\omega)\right]$.

Some Remarks

- Every group element can be written as a product of generators and their inverses.
- If the group is finite we do not need the inverses.
- Assumption: The generating set contains all inverses unless the group is finite: Every group element is a product of generators.
- We just need to compute repeatedly images of all orbit elements under all generators.

The plain vanilla orbit algorithm

Input: A group G, given by a generating set $\mathbf{g} = \{g_1, \ldots, g_m\}$, acting on a domain Ω. Also a point $\omega \in \Omega$.
Output: return the orbit ω^G.

```
begin
 1: $\Delta := [\omega]$;
 2: for $\delta \in \Delta$ do
 3:     for $i \in \{1, \ldots, m\}$ do
 4:         $\gamma := \delta^{g_i}$;
 5:         if $\gamma \notin \Delta$ then
 6:             Append $\gamma$ to $\Delta$;
 7:         fi;
 8:     od;
 9: od;
10: return $\Delta$;
end
```

Representatives

We often do not only want to test whether a point δ lies in the orbit of ω under G, but also find an element $g \in G$ such that $\omega^g = \delta$.

To do so we simply need to multiply up with the generators which give new images. Use a transversal T which for each $\gamma \in \omega^G$ stores an element $T[\gamma]$ such that $\omega^{T[\gamma]} = \gamma$.

There is a nontrivial search problem in determining $T[\gamma]$ for given γ. Also for memory reasons, store as factored transversal, also called Schreier vector.

A special case of this is the action of G on itself by right multiplication. The orbit of 1 is the group, representatives can remember the factorization of an element as generators product. Still only feasible for small groups.

Orbit algorithm with representatives

Input: A group G, given by a generating set $\mathbf{g} = \{g_1, \ldots, g_m\}$, acting on a domain Ω. Also a point $\omega \in \Omega$.

Output: return the orbit ω^G and a transversal T.

```
begin
 1: $\Delta := [\omega]$;
 2: $T := [1]$;
 3: for $\delta \in \Delta$ do
 4:     for $i \in \{1, \ldots, n\}$ do
 5:         $\gamma := \delta^{g_i}$;
 6:         if $\gamma \notin \Delta$ then
 7:             Append $\gamma$ to $\Delta$;
 8:             Append $T[\delta] \cdot g_i$ to $T$;
 9:         fi;
10:     od;
11: od;
12: return $\Delta$, $T$;
end
```

Finding stabilizer generators

Transversal elements also are coset representatives for the cosets of $\mathrm{Stab}_G(\omega)$ in G. The following lemma thus describes stabilizer generators:

Lemma

(Schreier) *Let $G = \langle \mathbf{g} \rangle$ a finitely generated group and $S \leq G$ with $[G{:}S] < \infty$. Suppose that $\mathbf{r} = \{r_1, \ldots, r_n\}$ is a set of representatives for the cosets of S in G, such that $r_1 = 1$. For $h \in G$ we write $\overline{h}$ to denote the representative $\overline{h} := r_i$ with $Sr_i = Sh$. Let*

$$U := \{r_i g_j (\overline{r_i g_j})^{-1} \mid r_i \in \mathbf{r}, g_j \in \mathbf{g}\}$$

Then $S = \langle U \rangle$.

The set U is called a set of Schreier generators for S. It typically is highly redundant. (Eliminate redundants by element test.)

Proof

$$
\begin{aligned}
x &= g_{i_1} g_{i_2} \cdots g_{i_m} = t_1 g_{i_1} g_{i_2} \cdots g_{i_m} \quad [\text{setting } t_1 := r_1 = 1] \\
&= t_1 g_{i_1} ((\overline{t_1 g_{i_1}})^{-1} \cdot \overline{t_1 g_{i_1}}) g_{i_2} \cdots g_{i_m} \quad [\text{insert } 1] \\
&= (t_1 g_{i_1} (\overline{t_1 g_{i_1}})^{-1}) t_2 g_{i_2} \cdots g_{i_m} \quad [\text{set } t_2 := \overline{t_1 g_{i_1}}] \\
&= \underbrace{t_1 g_{i_1} (\overline{t_1 g_{i_1}})^{-1}}_{=: u_1 \in U} t_2 g_{i_2} \left((\overline{t_2 g_{i_2}})^{-1} \cdot \overline{t_2 g_{i_2}} \right) \cdots g_{i_m} \\
&= u_1 \cdot \underbrace{t_2 g_{i_2} \overline{t_2 g_{i_2}}^{-1}}_{=: u_2 \in U} \cdot t_3 g_{i_3} \cdots g_{i_m} \quad [\text{set } t_3 = \overline{t_2 g_{i_2}}] \\
&\vdots \\
&= u_1 u_2 \cdot \cdots \cdot u_{m-1} \cdot t_m g_m = u_1 u_2 \cdot \cdots \cdot u_{m-1} \cdot \underbrace{t_m g_m \underbrace{\overline{t_m g_m}}_{=1}}_{=: u_m}
\end{aligned}
$$

Orbit/Stabilizer algorithm

```
begin
 1: $\Delta := [\omega]; T := [1]; S := \langle 1 \rangle;$
 2: for $\delta \in \Delta$ do
 3:     for $i \in \{1, \ldots, n\}$ do
 4:         $\gamma := \delta^{g_i};$
 5:         if $\gamma \notin \Delta$ then
 6:             Append $\gamma$ to $\Delta$;
 7:             Append $T[\delta] \cdot g_i$ to $T$;
 8:         else
 9:             $S := \langle S, T[\delta] \cdot g_i \cdot T[\gamma]^{-1} \rangle;$
10:         fi;
11:     od;
12: od;
13: return $\Delta, T, S;$
end
```

Stabilizer Chains

Let $G \leq S_n$ given by generators. Pick a point β_1 and compute

- Orbit β_1^G
- Generators for $G^{(1)} := \mathrm{Stab}_G(\beta_1)$.
- A transversal = Coset representatives for $G^{(1)}$ in G.

Repeat with a new point β_2 with $G^{(1)}$ in place of G.

Eventually $G^{(m)} = \mathrm{Stab}_{S_{m-1}}(\beta_m) = \mathrm{Stab}_G(\beta_1, \ldots, \beta_m) = \langle 1 \rangle$. The set $\{\beta_1, \ldots, \beta_m\}$ is called a base of G, the list of subgroups is a stabilizer chain for G, a union of generators of the $G^{(i)}$ a set of strong generators.

Properties

- $m \leq \log_2(|G|)$ if irredundant.
- We can write any $g \in G$ in the form $g = r_m r_{m-1} \cdots r_1$ with r_i a coset representative for $G^{(i)}$ in $G^{(i-1)}$ (sifting).
- Every group element is determined uniquely by the base images.
- Correspondence between G and cartesian product of orbits – enumeration of G.
- $|G|$ is the product of the orbit lengths.
- In practice, use Schreier Vectors for transversal storage.

Example

Let $G = A_4$ with base $(1, 2)$. Then $G = G^{(0)} = \langle(1, 2, 3), (2, 3, 4)\rangle$, $G^{(1)} = \mathrm{Stab}_G(1) = \langle(2, 3, 4)\rangle$ and $G^{(2)} = \mathrm{Stab}_G(1, 2) = \langle\rangle$.

```
rec(generators:=[(1,2,3),(2,3,4)],
    orbit:=[1,2,3,4],
    transversal:=[(),(1,2,3),(1,3,2),(1,4,2)],
    stabilizer := rec(
       generators:=[(2,3,4)],
       orbit:=[2,3,4],
       transversal:=[(),(2,3,4),(2,4,3)],
       stabilizer:= rec(
         generators:=[])  )  )
```

Sifting

A principal operation for stabilizer chains is to sift an element x through a chain, level by level.

If β is the next base point, consider $\gamma := \beta^x$, if γ is in the orbit of β, divide the transversal element for γ off of x. (Otherwise we have proven that the element is not in the group!)
Then go to the next level.

If at the end the remainder is trivial, the sifted element is contained in the group described by the stabilizer chain (and we have expressed it as a product of coset elements).
Otherwise the remainder is a "canonical" coset representative

Sifting Algorithm

Input: A stabilizer chain C for a group G and an element $g \in G$
Output: A list $L = [b_1, b_2, \ldots, b_m]$ of coset representatives, such that $g = b_m b_{m-1} \cdots b_1$.

```
begin
 1: $L := []$;
 2: while $C$.generators $<>$ [] do
 3:     $\beta := C$.orbit[1];
 4:     $\delta = \beta^g$;
 5:     $r := C$.transversal[$\delta$];
 6:     $g := g/r$;
 7:     Add $r$ to $L$;
 8:     $C := C$.stabilizer;
 9: od;
10: return $L$
end
```

Problem

The number of Schreier generators is large, but they are highly redundant.

One approach, the Schreier-Sims algorithm, is to do the calculation recursively, working with partial stabilizer generating sets and doing for each new Schreier generator an element test in the partial chain below.
As soon as an image $\gamma := \beta^x$ is not yet in the orbit, we consider the (partially sifted) element as a new generator on that level and continue expanding the orbit, in turn creating new Schreier generators on a lower level.

A newer approach is to use only some random (or random products of) Schreier generators. One needs to verify the result. (For example by checking the order deduced from the finished chain.)

If we know the group order or even have already one stabilizer chain, finding a new chain for a particular base is much easier.

Algorithmic Consequences

- Test element containment in G by sifting through a chain.
- The sifting remainder is a "canonical" coset representative.
- If we keep word expression for all transversal elements and stabilizer generators, sifting finds a (long!) word in the generators for each $g \in G$.
- Calculate the stabilizer of tuples as well as elements mapping a tuple of points in a prescribed way.
- In practice, use Schreier Vectors for transversal storage.

Homomorphisms

Since stabilizer chains can be used to decompose into generators, they offer the evaluation of group homomorphisms which are defined by prescribing the images for a set of group generators. Instead of tracing through a word expression, however the following approach is much easier: suppose we have a homomorphism $\varphi\colon G \to H$. (For simplicity assume that both G and H are permutation groups.)

We now form the direct product $D := G \times H$ is a permutation group of degree $\deg G + \deg H$ (simply shifting the points on which H acts to be disjoint from the points for G); for example $S_3 \times S_4$ would be generated by

$$\langle (1,2,3), (1,2), (4,5,6,7), (4,5) \rangle$$

Now consider the subgroup S of D, generated by elements of the form (g, g^{φ}).
We calculate a stabilizer chain for S, using only base points for G.

If we now sift $(g, 1)$ through this chain, we get the remainder $(1, (g^{\varphi})^{-1})$ and thus can calculate the image g^{φ}.

The stabilizer in S of all the points corresponding to H yields the kernel of φ. Similarly one can check whether φ actually is a homomorphism or compute preimages.

Backtrack

Backtrack algorithms for permutation groups find (one or all) elements with particular properties in a group $G \leq S_n$ by running (in principle) through all elements of G, using a tree with branchings at each level corresponding to the orbits in a stabilizer chain.

To improve performance, problem-specific prunings are applied to the search tree, for example if we want to centralize $(1,2)(3,4)$, then any element mapping 1 to 3 must map 2 to 4.

While the complexity is exponential, practical performance is often servicable. Backtrack algorithms are used to calculate in permutation groups:

- Centralizer
- Normalizer
- Set Stabilizer
- and the corresponding representatives

Structural computations

The central routine for all higher level routines is the determination of a composition series. This is done by repeated application of the following paradigm:

> *Given a permutation group G, either prove that G is simple; or find – from a suitable action – a homomorphism (which we can evaluate by performing the action)* $\varphi\colon G \to H$ *such that H is a permutation group of degree smaller than that of G or* $N := \ker\varphi > \langle 1 \rangle$.

Actions are initially actions on orbits and blocks, as well as on suitably defined substructures of the group.
More about this next week.

Disguised Permutation groups

Some groups can be considered as permutation groups on a set different than the natural numbers. By labelling the underlying set, permutation group methods become available.
Currently GAP actually computes in these groups by “shadowing” them with an isomorphic permutation action.

- Matrix groups over finite fields, acting on vectors.
- Automorphism groups of groups, acting on substs of group elements.
- Once the permutation image is established it is often possible to reduce the domain using ad-hoc methods.