LMS/EPSRC Course
Computational Group Theory
St Andrews 2013

Permutation Groups 1: Orbits and Stabilizers

Alexander Hulpke
Department of Mathematics
Colorado State University
Fort Collins, CO, 80523, USA
http://www.math.colostate.edu/~hulpke

Elements

Work in $G \leq S_\Omega$, typically $\Omega = \{1,...,n\}$.

Can represent elements of G on the computer, each permutation requires (roughly) $n \cdot \log_2(n)$ bits. Obvious multiplication/inverse.

$n = 10^5$, 1GB memory: 5000 elements

As $|S_\Omega| = n!$ we cannot assume to store many elements.

Ground Rules

Instead represent/store subgroups by generators. At most $\log_2(|U|)$ needed. Often in praxis <10.

We can store *some* extra elements for data structures.

Utilize (natural) group actions:

- Permutations on points
- Matrices on Vectors
- On group elements

Group Actions

We now assume that the group G acts on the set Ω from the right: $g: \omega \to \omega^g$. (Here and in GAP always from the right.) The natural questions are to find:

ORBIT: ω^G of $\omega \in \Omega$. (Length, Elements).

STAB: Stabilizer of $\omega \in \Omega$ as subgroup of G.

TRANSPORTER: For $\omega, \delta \in \Omega$ find element $g \in G$ such that $\omega^g = \delta$ (or confirm that no such an element exists).

Basic Orbit Algorithm

G acts on Ω. Orbit ω^G of $\omega \in \Omega$ consists of all images ω^g for $g \in G$.

Each g is a product of generators (and inverses if G is infinite – assume included)

Take iteratively images of all points obtained under all generators.

Cost: $|\omega^G| \cdot$#gens.

Input: $G = \langle g_1,\ldots,g_m \rangle$, acting on Ω. Seed $\omega \in \Omega$.

Output: The orbit ω^G.

```
begin
  Δ := [ω];
  for δ ∈ Δ do
    for i ∈ {1,...,m} do
      γ := δ^{g_i};
      if γ ∉ Δ then
        Append γ to Δ;
      fi;
    od;
  od;
  return Δ;
end.
```

Modification: Transporters

Keeping track, we also get a *Transversal* of transporters $T[\delta]$, such that $\omega^{T[\delta]} = \delta$.

These $T[\delta]$ are also are reps. for (right) cosets of $\mathrm{Stab}_G(\omega)$ in G.

If $\omega^g = \delta$ and $\omega^h = \gamma$, then $\delta^x = \gamma$ for $x = g^{-1}h$, solving the general

```
begin
  Δ := [ω]; T := [1];
  for δ ∈ Δ do
    for i ∈ {1,...,m} do
      γ := δ^{g_i};
      if γ ∉ Δ then
        Append γ to Δ;
        Append T[δ] · g_i to T;
      fi;
    od;
  od;
  return Δ;
end.
```

Schreier's Lemma

If $\omega^a=\delta$, $\delta^b=\gamma$, $\omega^c=\gamma$, then $a\cdot b/c \in \mathrm{Stab}_G(\omega)$.

By SCHREIER's lemma (→ Max's lecture 3) such elements, formed in all ways, generate $\mathrm{Stab}_G(\omega)$:

Lemma: $G=\langle \mathbf{g}\rangle$ finitely gen., $S\leq G$ with $[G:S]<\infty$.
Suppose $\mathbf{r}=\{r_1,\ldots,r_n\}$ set of representatives for cosets of S in G, such that $r_1=1$.
For $h\in G$ write $\underline{h}:=r_i$ for the chosen representative such that $Sr_i=Sh$. Let

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

Then $S=\langle U\rangle$.
U is called a set of *Schreier generators* for S.

Modification: Stabilizer

The transversal gives coset representatives, the image of ω identifies cosets.

Thus we can form Schreier generators.

At this point $S := \langle S,\ T[\delta] \cdot g_i\ /T[\gamma] \rangle$ just produces a generating set.

```
begin
  Δ := [ω];T:=[1];S:=⟨1⟩;
  for δ ∈ Δ do
    for i ∈ {1,...,m} do
      γ := δ^g_i;
      if γ ∉ Δ then
        Append γ to Δ;
        Append T[δ] · g_i to T;
      else
        S:=⟨S, T[δ] · g_i /T[γ]⟩;
      fi;
    od;
  od;
  return Δ;
end.
```

Remarks on Performance

- Need to store whole orbit – Available memory limits scope.
- Store transversal T in factored form to save memory – *Schreier vector*. (Issue: balanced tree of low depth)
- Cost of basic algorithm is dominated by test $\gamma \in \Delta$? to check for new points – Data structures.
- There is a **huge** number of Schreier generators: Index of stabilizer $\times$ # group generators.

  Usually many of them are redundant (or even trivial).

Schreier generators

The number of Schreier generators cannot be reduced in general, as in free groups (→ Max's lectures) all are needed.

- If there is a membership test in the partial stabilizer, test each new generator whether it is already in. (Still does not produce minimal sets!)
- Let $S=\{s_1,s_2,\ldots,s_n\}$ be a generating set. A ***random subproduct*** of S is a product $x=\prod_i s_i^{\epsilon_i}$ with the ϵ_i chosen independently by random from $\{0,1\}$.

Using Random Subproducts

Lemma: Let $U = \langle S \rangle$ have subgroup chains of maximum length $m \leq \log_2(|U|)$. Then for every $\delta > 0$ there exists a constant c, such that $c \cdot m$ random subproducts generate U with probability $1 - \delta$.

Theorem (Babai, Cooperman, Finkelstein, Luks, Seress, '95):

There is an algorithm that computes for all $\delta > 0$ in $\mathcal{O}(|S| \log m)$ operations a set of size $\mathcal{O}(m)$ that generates U with probability $1 - \delta$.

Dictionaries

To test $\gamma \in \Delta$? (and to determine $T[\gamma]$) one can

- Search linearly though Δ. (Orbit length n requires $\mathcal{O}(n^2)$ element comparisons)
- Keep a sorted copy of Δ. (needs $<$ test, $\mathcal{O}(n \log(n))$
- Determine index number for γ. (bit list, $\mathcal{O}(n)$)
- # Search in a hash table. (Hash key, almost $\mathcal{O}(n)$)

In GAP, the `Dictionary` data type provides a uniform interface.

All nontrivial approaches require dedicated handling for each data type. (Many objects do not have unique representations!)

Variants: Stabilizer Order

If storage or time requirements are an issue the following variants might help if $|G|$ is known:

- Known orbit length, partial stabilizer order can give early termination. If we can calculate subgroup orders, can stop if the largest proper divisor of $[G:S]$ is smaller than the orbit length.

- Use of Birthday paradox to estimate orbit length – indicate that full stabilizer is known.

More often than not I end up re-implementing an orbit algorithm instead of using a generic default...

Consequences / Summary

The orbit algorithm and its variants let us solve

ORBIT, STABILIZER and TRANSPORTER as long as the orbit fits into memory.

By keeping track of the transversal, we write transversal elements as product of generators.

If we let G act on itself this allows for element lists, centralizer, normalizer *in small groups*.

To deal with larger cases, we need to use more group theory!

Variant: Spinning Algorithm

Take as Ω an algebraic structure, return the smallest substructure containing a seed.

Instead of an orbit, Δ is generating set for the closure. Map all elements of Δ under all group generators.

Add to Δ if image γ not in ⟨Δ⟩. (Group action preserves closure.)

Applications are e.g. normal closure, submodule.

Group Actions in GAP

In GAP group actions are done by the operations:

- `Orbit`, `Orbits`
- `Stabilizer`, `RepresentativeAction` (Orbit/Stabilizer algorithm, sometimes backtrack, → lecture 2).
- `Action` (Permutation image of action) and `ActionHomomorphism` (homomorphism to permutation image with image in symmetric group)

The arguments are in general are:

- A group G. (Will act by its `GeneratorsOfGroup`.)
- A domain Ω (may be left out for `Orbit`, `Stabilizer`, but may improve performance).
- Point ω, or list of point seeds for `Orbits`.

Action functions

The last argument is an *action function* actfun(ω,g):=ω^g. This is a GAP function that implements the actual action. Some predefined actions are:

- OnPoints: action via ^. The default if not given.
- OnTuples, OnSets: Lists or sets (i.e. sorted lists) of points.
- OnSetsSets, OnSetsTuples, etc.
- OnRight: right multiplication *. (e.g. on cosets)
- OnLines: Projective action on vectors scaled to have first nonzero entry 1.
- OnSubspacesByCanonicalBasis: Subspaces, given as list of RREF basis vectors.
- Permuted: Permuting list entries.

Optional Arguments

G may act via a homomorphism φ. (Say, a matrix group acting on enumerated vectors.) One can compute (in particular stabilizers) by giving two further list arguments:

gens A list of group generators,
imgs Images of these generators under φ.

Action Homomorphisms by default have codomain S_n. For large *n* this is inefficient. Append the string argument "surjective" to force the codomain equal to the image.

Action on cosets: Internal use of `PositionCanonical` (position of a *standard* equivalent object) allows:

`ActionHomomorphism(G,RightTransversal(G,U),OnRight,"surjective");` to get the action on the cosets of *U* in *G*.

`FactorCosetAction` produces the same result.

KiKA
von ARD und ZDF

ENDE DES 1. TEILS