Enumerating orbits of Co_1 on $\mathbb{P}(\mathbb{F}_5^{24})$

Lehrstuhl D für Mathematik
RWTH Aachen

Perth 2006

All of this is joint work with:

- Robert A. Wilson
- Felix Noeske
- Jürgen Müller
- Frank Lübeck
- Christoph Köhler

Long orbits

Gunter Malle classified long orbits of quasi-simple groups:

Definition (Long orbit)

G: quasi-simple group, $\rho : G \to \mathrm{End}_{\mathbb{F}_q}(\mathbb{F}_q^d)$,
induces an action on the projective space $\mathbb{P}(\mathbb{F}_q^d)$
An orbit is called long if it has at least $\frac{q^{d-1}-1}{q-1}$ elements.

Recently he posed the following question:

Question

Does $2.\mathrm{Co}_1$ have a long orbit in its action on $\mathbb{F}_5^{24}$?

The Action

$2.\mathrm{Co}_1 = \mathrm{Aut}(\Lambda)$ where Λ is the Leech lattice:

the unique 24-dimensional, even, unimodular lattice with no vectors of norm 2.

Co_1 : one of the 26 sporadic simple groups.

$\Longrightarrow$ 24-dimensional integral representation of $2.\mathrm{Co}_1$

We consider this representation mod 5.

$\longrightarrow$ Download two matrices in $\mathbb{F}_5^{24\times 24}$ from Rob's page.

The Size

$$|\mathrm{Co}_1| = 4\,157\,776\,806\,543\,360\,000 \approx 4 \cdot 10^{18}$$

$$|\mathbb{P}(\mathbb{F}_5^{24})| = \frac{5^{24}-1}{5-1} = 14\,901\,161\,193\,847\,656 \approx 15 \cdot 10^{15}$$

Is there an orbit of length at least

$$\frac{5^{23}-1}{5-1} = 2\,980\,232\,238\,769\,531 \approx 3 \cdot 10^{15} \quad ?$$

Storing a field element in 4 Bits, we would need at least

1 387 778 Gigabytes ≈ 1.4 Petabytes

of memory to simply store all elements of such an orbit.

Standard orbit enumeration

Algorithm (Orbit enumeration)

Input: $G = \langle g_1, \ldots, g_r \rangle$ acting on X, $x \in X$
set $l := [x]$
for z in l:
    for g in $[g_1, \ldots, g_r]$:
        if zg in l:
            compute stabiliser element
        else:
            append zg to l
Output: l and generators for $\mathrm{Stab}_G(x)$

We need to

- store all points in memory,
- look up points efficiently, and
- compute xG and $\mathrm{Stab}_G(x)$ without knowing $|G|$.

Storing U-suborbits

$U < G$ a helper subgroup $\longrightarrow$ archive U-suborbits!

We want:

- given $x \in X$, store xU and compute $|xU|$
- given $z \in X$, decide whether z lies in a stored xU

To this end, let $\bar{\ }: X \to Y$ be a homomorphism of U-sets:

- enumerate Y completely
- choose one element in each U-orbit of Y arbitrarily
- call these U-minimal
- for $y \in Y$, store a $u_y \in U$ such that yu_y is U-minimal
- for U-minimal $y \in Y$, store generators of $\mathrm{Stab}_U(y)$
- call $x \in X$ U-minimal, if $\bar{x} \in Y$ is U-minimal

Algorithm

Store xU by storing all U-minimal elements in xU.

Storing U-suborbits II

If $x \in X$ is U-minimal (i.e. $\bar{x} \in Y$ is U-minimal), then

$x\mathrm{Stab}_U(\bar{x})$ is the set of U-minimal elements in xU

Algorithm (Storing xU)

Input: $x \in X$
look up $u_{\bar{x}}$ and compute $z := xu_{\bar{x}}$
enumerate and store $z\mathrm{Stab}_U(\bar{z})$
find $\mathrm{Stab}_U(z) \leq \mathrm{Stab}_U(\bar{z})$ and thus $|zU| = |xU|$

Algorithm (Looking up $z \in X$)

Input: $z \in X$, some stored xU
look up $u_{\bar{z}}$ and compute $w := zu_{\bar{z}}$
look up w in list of stored points
$z \in xU$ iff w already stored

Orbit by suborbits

Algorithm (Orbit by suborbits)

Input: $G = \langle g_1, \dots, g_r \rangle$ acting on X, $x \in X$
store xU and set $l := [x]$
repeat forever:
    for z in l:
        for g in $[g_1, \dots, g_r]$:
            if zgU already stored:
                compute stabiliser element
            else:
                store zgU
                append zg to l
    exit if orbit and stabiliser ready
    for z in l:
        for u in generators of U:
            append zu to l
Output: l, U-suborbits, generators for $\mathrm{Stab}_G(x)$

Using two helper subgroups

$U < V < G$ two helper subgroups

$$\underbrace{\hat{\ } : X \to Z}_{\text{hom. of } V\text{-sets}} \quad , \quad \underbrace{\bar{\ } : Z \to Y \quad , \quad \bar{\ } : X \to Z \to Y}_{\text{hom. of } U\text{-sets}}$$

$\Longrightarrow$ can archive U-suborbits in Z and X!

Preparations:

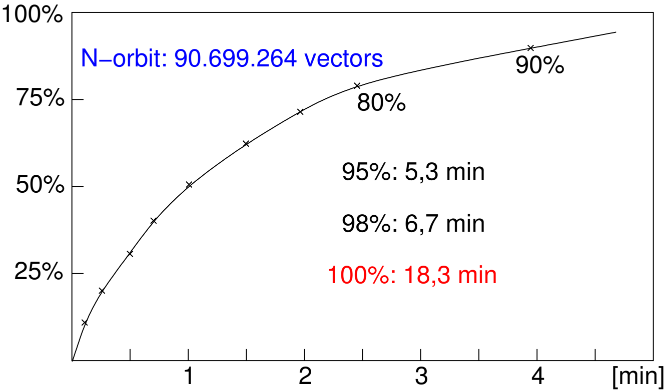
- enumerate Z completely by U-orbits
- choose one U-minimal point in each V-orbit of Z, call it V-minimal
- call $x \in X$ V-minimal, iff $\hat{x} \in Z$ is V-minimal
- compute a transversal $\mathcal{T} : V = \dot{\bigcup}_{t \in \mathcal{T}} tU$
- for every U-minimal point in $z \in Z$ store:
    - $\mathrm{Stab}_V(z)$ if z is V-minimal
    - nothing if the V-minimal point of zV lies in zU
    - an element $t_z \in \mathcal{T}$ such that zt_zU contains the V-minimal point of zV

V-minimalising

Algorithm (V-minimalisation)

Input: $x \in X$
lookup $u \in U$ such that $w := xu$ is U-minimal
($\Longrightarrow \hat{w} \in V$ is U-minimal)
if $\hat{w}U$ does not contain the V-minimal point of $\hat{w}V$:
    look up $t \in \mathcal{T}$ such that $\hat{w}tU$ contains it
    set $w := wt$
    look up $u' \in U$ such that wu' is U-minimal
    set $w := wu'$
else:
    set $t := 1$ and $u' := 1$
(now w is U-minimal and
    $\hat{w}U$ contains the V-minimal point of $\hat{w}V$)
unless $\hat{w}$ is the V-minimal point:
    find $s \in \mathrm{Stab}_U(\bar{w})$ with $\hat{w}s$ V-minimal
Output: $v_x := utu's \in V$ such that xv_x is V-minimal

Evolution of an orbit enumeration



This is a typical time evolution for orbit enumerations!

A half is enough!

Assume we

- know $|G|$,
- already have enumerated some part of xG, and
- already know some $S < \mathrm{Stab}_G(x)$ and $|S|$.

Then:

$$2 \cdot \text{Size(enumerated part)} \cdot |S| \geq |G|$$

if and only if

- S already is the full stabiliser $\mathrm{Stab}_G(x)$ and
- we already have enumerated at least half of $|xG|$

because if $S < \mathrm{Stab}_G(x)$ then the index is at least 2.

Finding homomorphisms

Let G act linearly on a F-vectorspace M:

$$\rho : G \to \mathrm{End}_F(M)$$

$N < M$ a G-invariant subspace,
$\pi : M \to M/N$ the canonical map.

Then the following diagram commutes for all $g \in G$:

$$\begin{tikzcd} M \arrow[r, "\cdot g"] \arrow[d, "\pi"'] & M \arrow[d, "\pi"] \\ M/N \arrow[r, "\cdot g"] & M/N \end{tikzcd}$$

with the induced action on M/N.

The same holds for the projective action, if we replace

- M by $\mathbb{P}(M)$ and
- $\mathbb{P}(M/N)$ by $\mathbb{P}(M/N) \cup \{0\}$.

Finding orbits

We can now enumerate halves of orbits.

But how do we avoid enumerating them more than once?

Assume we "know" a half of xG, then for some $w \in X$ we can still check, whether $w \in xG$:

Algorithm (Membership test in half-orbit)

Input: $w \in X$ and at least a half of xG.
for 100 random elements $g \in G$:
    if wg in half of xG:
        return True
return False

Find bigger orbits by random search.

But how to find small orbits?

Finding the small orbits

Short orbits have big stabilisers.

Guess stabilisers:

- use maximal subgroups ($\rightarrow$ Rob's WWW Atlas)
- find invariant subspaces ($\rightarrow$ MEATAXE)

Guess elements of stabilisers:

- use conjugacy class reps. ($\rightarrow$ Rob's WWW Atlas)
- try vectors in eigenspaces

Finding all orbits

Build up a database of halves of pairwise disjoint orbits.

Produce representative candidates for the small orbits.

Produce random representatives for the big orbits.

For all vectors: Test if they are in a known orbit half.
If not, enumerate half of new orbit.

Do this until the sum of the orbit lengths is the total number of points.

Verification of Disjointness

How can we prove that two orbits are different knowing only half of them?

Solution: Enumerate 51% of the orbits!

Lemma (Disjointness)

Two subsets of xG of size $> |xG|/2$ intersect nontrivially.

Algorithm (Disjointness proof)

Input: $M \subseteq xG$ with $2 \cdot |M| > |xG|$ and
$M' \subseteq x'G$ with $2 \cdot |M'| > |x'G|$
assume both M and M' are unions of V-sets

Check whether a V-orbit rep. of M is in M' or not.

The Result

The orbit lengths of the 48 orbits of Co_1 on $\mathbb{P}(\mathbb{F}_5^{24})$ are:

98 280	636 539 904 000	103 119 464 448 000
8 386 560	1 080 188 928 000	180 459 062 784 000
199 017 000	1 611 241 632 000	180 459 062 784 000
226 044 000	4 687 248 384 000	193 348 995 840 000
2 314 690 560	4 687 248 384 000	262 485 909 504 000
4 577 391 000	9 374 496 768 000	300 765 104 640 000
4 629 381 120	12 889 933 056 000	524 971 819 008 000
16 982 784 000	12 889 933 056 000	721 836 251 136 000
46 872 483 840	17 186 577 408 000	773 395 983 360 000
67 135 068 000	17 823 117 312 000	824 955 715 584 000
93 744 967 680	21 873 825 792 000	824 955 715 584 000
318 269 952 000	21 873 825 792 000	1 203 060 418 560 000
402 810 408 000	32 998 228 623 360	1 443 672 502 272 000
407 586 816 000	51 559 732 224 000	1 924 896 669 696 000
407 586 816 000	69 296 280 109 056	2 165 508 753 408 000
563 934 571 200	103 119 464 448 000	2 887 345 004 544 000

Long limit: 2 980 232 238 769 531

$\Longrightarrow$ no long orbit!

Total: 14 901 161 193 847 656

Memory and Runtime Data

In the end, the calculation

- used three helper subgroups: $U_1 < U_2 < U_3 < \mathrm{Co}_1$
- of orders: 10 752, 371 589 120 and 89 181 388 800,
- using quotients of codimensions 8, 8 and 16,
- needed 2.3 Gigabytes of main memory on one PC
- and about 2.5 hours of CPU time,
- stored about 30 000 000 vectors altogether
- thereby saving a factor of about 500 000 000, and
- was performed in GAP using the orb package.