# Orbiter User's Guide

Anton Betten

August 16, 2018

**Abstract**

We discuss how to use the program system Orbiter for the classification of combinatorial objects.

## 1 Introduction

This is a User's guide to the program package Orbiter [1] [2] for the classification of combinatorial objects. Orbiter is a library of C++ classes for algebraic combinatorial objects. It does not have a user interface, but it comes with several pre-programmed applications which can be invoked using the command line or using makefiles. This means that there are basically two ways in which Orbiter can be used. The first way is recommended for novice users. Those kinds of users would use Orbiter applications through unix ccommand lines, possibly using makefiles to simplify the typing com commands. The second type of user are more experienced programmers. They would build their own applications using the C++ library.

## 2 Installing Orbiter

Orbiter is distributed through github (`https://github.com`). Search for abetten/orbiter or go to

`https://github.com/abetten/orbiter`

Once there, look for the green button called "Clone or download". Click that button. It offers two options: "Open in Desktop" and "Download ZIP" Choose one of them to download orbiter. Orbiter is compiled with makefiles. For windows users, it can be installed by using `cygwin`. For mac users, the mac development suite needs to be installed. For linux users, the compiler environment (for instance gnu C++) needs to be installed. We also need to have a terminal (console) where we can type commands. Assuming that we have the compiler tools and a terminal window avalable, the installation proceeds as follows:

Enter the directory `ORBITER/src` and type

```
make
```

This should create a lot of text output to the console. Assuming that the command executes without errors, orbiter is now installed. In order to test it, go to the subdirectory `ORBITER/examples`. There are several subdirectories containing test problems. The test problems are calls to orbiter, asking it so solve small problems. For instance, the subdirectory groups contains some problems related to groups. Once in the directory `ORBITER/examples/groups`, issue the command `make` to run the first problem. To see what kind of problems are available, open the file `makefile` and see what targets are defined. You can type `make target` where `target` is any of the targets in the makefile to run that particular problem. Most problems will create a lot of output to the console.

# 3 Theoretical Background

Combinatorial objects often can be identified with orbits of group actions. Hence classifying combinatorial objects often times is equivalent to computing the orbits of a group acting on a set. Let $G$ be a group and let $X$ be a finite set. For two elements $x, y$ of $X$, write $x \sim y$ if $x$ and $y$ belong to the same $G$-orbit. A transversal for the orbits of $G$ on $X$ is a sequence $T = (t_1, \ldots, t_n)$ for some integer $n$ such that for every $x \in X$ there is exactly one $t_i$ with $x \sim t_i$. The classification problem for the orbits of $G$ on $X$ is computing a transversal for the orbits. The recognition problem for the group $G$ acting on the set $X$ is the following: Given an element $x \in X$, determine the element $t_i \in T$ auch that $x \sim t_i$. Here, $T$ is the transversal that was computed previously. The constructive recognition problem for the group $G$ acting on the set $X$ is the following: Given an element $x \in X$, determine the element $t_i \in T$ auch that $x \sim t_i$ and in adition find a group element $g \in G$ such that $xg = t_i$. The main challenge is to find efficient algorithms for the classification problem, the recognition problem and the constructive recognition problem. With these, the isomorphism problem is solved easily as well. Namely, given arbitrary element $x, y \in X$, we can determine an element $g \in G$ with $xg = y$ or determine that no such element exists using two invocations of the constructive recognition procedure. Namely, let $h_1 \in G$ be such that $xh_1 = t_i$ for some $i \leq n$. Also, let $h_2 \in G$ be such that $yh_2 = t_j$ for some $j \leq n$. If $i = j$ then $g := h_1 h_2^{-1}$ is the isomorphism from $x$ to $y$. If $i \neq j$ then no such isomorphism exists.

Standard orbit algorithms are related to the idea of Schreier trees. The complexity of these algorithms is linear in the size of the orbit. For many combinatorial objects, posets can be used to aid the classification task. The group induces an action on the poset and the orbits of the group on the poset are computed. Often times, the orbits on the objects at a specific layer in the poset correspond to the combinatorial objects that are desired. For the theory of group action on posets, see [10]. For a description of poset based classification, see [3]. If poset classification is used, the algorithm can proceed much faster than the ordinary orbit algorithm on the set of objects. Poset classification is the process by which the orbits of $G$ on a poset are classified. Additional information is computed also. This additional information described how the orbits are related. An early application of this can be seen in [4], where the orbits of the Mathieu group acting on the set of subsets of the set $\{1, \ldots, 24\}$ are computed and a diagram is presented which contains detailed information about how the orbits are related in the sense of [10].

In order to use Orbiter, a group $G$ is defined. Then, an action of this group on a set $X$ is constructed. Using the induced action on $k$-subsets or $k$-subspaces, a poset action on a lattice $\mathcal{L}$ is defined. Then, a subposet $\mathcal{P}$ of the lattice is selected, and the orbit of the group $G$ on this poset $\mathcal{P}$ are computed. The orbit representatives and the stabilizer groups are listed and stored. We will describe these steps next.

The algorithm described in [3] is based on earlier work of Schmalz [13]. Schmalz was concerned with computing double coset representatives in certain groups. The groups that Schmalz was woring on needed to have good "ladders" of subgroups. A ladder of subgroups is a sequences of subgroups $G_1, G_2, \ldots, G_r$ where two consecutive groups are related. This means that for $i = 1, \ldots, r-1$ we have $G_i \leq G_{i+1}$ or $G_i \geq G_{i+1}$. A subgroup ladder is good if the indices of consecutive terms are small. The problem of computing orbits on $k$-subsets and orbits on $k$-dimensional subspaces can be formulated equivalently as a problem of computing double cosets in either the symmetric group or the full semilinear matrix groups. For these groups, good subgroup ladders exist. Because of the use of subgroup ladders, Schmalz coined the term "Leiterspiel" for his algorithm. The closest translation into English would perhaps be "Snakes and Ladders." Orbiter is based on the Schmalz algorithm.

There are other algorithms which can be used to classify combinatorial objects. Most notably there is the method of canonical ancestors proposed by McKay [9], which may be seen as part of the family of algorithms which are called "orderly generation", and which have been discovered and refined many times, see [11], [5], [7], [8], [12], [6].

The Schmalz algorithm differs from orderly generation in some important ways. First, while orderly generation proceeds depth-first, the algorithm of Schmalz proceeds in a breadth-first manner. While orderly generation only keeps the current object in memory, the algorithm of Schmalz builds up the whole tree of orbits, storing quite a lot of information. While the orderly generation algorithm relies on a backtrack procedure to compute canonical forms, the algorithm of Schmalz does not backtrack. It uses the data structure about previously computed orbits to perform recognition. The differences in the two families are sufficiently large to expect different behavior of the algorithms on different kinds of problems.

All groups in Orbiter are finite permutation groups. However, we distinguish between groups of linear or affine type and ordinary permutation groups. Groups of linear type are groups that act linearly or affinely (possibly semilinearly) on the elements of a vector space. This allows us to encode group elements efficiently using matrices. For semilinear actions, we store a field automorphism. For affine actions, we store a translation vector. All other groups need their elements stored as permutations. Memory issues are very important in Orbiter, so finding the most efficient way to encode a group element matters. At times, a representation of matrices over finite fields as bitvectors is utilized, reducing the storage requirement even further.

Any group in Orbiter must come as permutation group. What this means is that a fixed permutation representation is chosen from the beginning. Later, new groups actions can be defined. It is important to know that any group always has a default permutation representation, independent of how the group elements are represented. For projective linear groups, a labeling of the points of $\mathrm{PG}(n-1,q)$ is used. For general linear and affine groups, a labeling of the vectors in $\mathbb{F}_q^n$ is used. For orthogonal groups, a labeling of the points of $Q^\epsilon(n-1,q)$ is used. For permutation groups, the set $\{0,\ldots,d-1\}$ is used, for some $d$.

# 4 Orbiter Applications

This section describes some of the available Orbiter applications and how they can be used. Mainly, Orbiter applications rely on the command line to being told what to do. Options are passed along using certain command line keys, which must be known. Suppose we want to create one of the orthogonal groups over a finite field. The orbiter application to do so is called `orthogonal_group.out`. Since an orthogonal group $O^\epsilon(d,q)$ involves three parameters, the Orbiter application requires us to use three options. By default, most Orbiter parameters are integers. They are `-epsilon` $\langle\epsilon\rangle$, `-d` $\langle d\rangle$ and `-q` $\langle q\rangle$. For instance, the command

```
orthogonal_group.out -v 2 -epsilon -1 -d 6 -q 2
```

creates the group $O^-(6,2)$ of order 51840. The parameter `-v` $\langle v\rangle$ specifies the verbose level. Higher values of $v$ mean more text output. This group $O^-(6,2)$ acts on the 27 points of the $Q^-(5,2)$ quadric. The quadric is given by the quadratic form

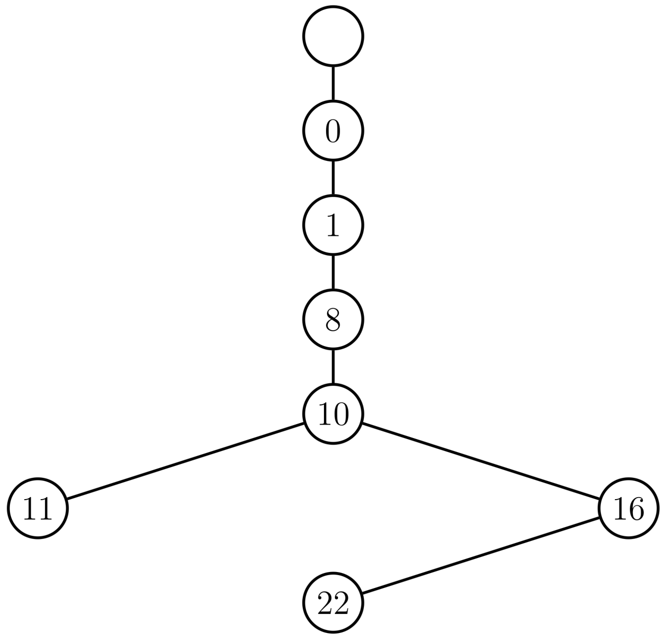
$$x_0x_1 + x_2x_3 + x_4^2 + x_4x_5 + x_5^2 = 0.$$

Generators for the group as $6\times 6$ matrices over $\mathbb{F}_2$ are listed in the output, as are corresponding permutations of the 27 points. Finally, the generators are written in machine readable compact form. The full output produced by this command is shown in Appendix A.

The next problem we wish to consider is classifying ovoids in $\Omega^-(5,2)$ under the group $O^-(6,2)$ from before. A partial ovoid is a set $S=\{P_1,\ldots,P_n\}$ of points $P_i=\langle v_i\rangle$ such that $\beta(v_i,v_j)\neq 0$ for all $i\neq j$. The partial ovoids in $\Omega^-(5,2)$ can be classified using the orbiter command

```
ovoid.out -v 5 -epsilon -1 -n 5 -q 2 -draw_poset -embedded -W
```

This produces the following diagram:




which shows that the set $\{0,1,8,10,16,22\}$ is the only partial ovoid in $\Omega^-(5,2)$ up to equivalence. The numbers stand for points in $\Omega^-(5,2)$ as listed in the output of the `orthogonal_group.out` command, and can be found in the output in Appendix A. They are

$$\begin{aligned}
0 &= (1,0,0,0,0,0),\\
1 &= (0,1,0,0,0,0),\\
8 &= (1,1,1,1,0,0),\\
10 &= (1,1,1,0,1,0),\\
16 &= (1,1,1,0,0,1),\\
22 &= (1,1,1,0,1,1).
\end{aligned}$$

# 5 Orbiter Programming

This section is devoted to the users who wish to write their own applications using the Orbiter library.

The program which was used to create the orthogonal groups is shown in Appendix B.

## A Output from the orthogonal group $O^{-}(6,2)$ command

```
~/DEV.18/GITHUB/orbiter/ORBITER/SRC/APPS/GROUPS/orthogonal_group.out -v 2 -epsilon -1 -d 6 -q 2
-v 2
-epsilon -1
-d 6
-q 2
action::init_orthogonal_group verbose_level=2
action::init_orthogonal_group before A->init_projective_group
action::init_orthogonal_group before O->init
orthogonal::init: epsilon=-1 n=6 (= vector space dimension) m=2 (= Witt index) q=2 verbose_level=2
choose_anisotropic_form over GF(2)
longinteger_object::create_from_base_10_string() str = 7
object = 7
unipoly_domain::create_object_by_rank_longinteger rank=7
quotient 3 remainder 1
quotient 1 remainder 1
quotient 0 remainder 1
unipoly_domain::create_object_by_rank_string X^{2} + X + 1
choosing the following primitive polynomial:
X^{2} + X + 1
choose_anisotropic_form over GF(2): choosing c1=1, c2=1, c3=1
orthogonal::init computing Gram matrix
orthogonal::init computing Gram matrix done
count_T1 epsilon = -1 not yet implemented, returning 0
count_T1 epsilon = -1 not yet implemented, returning 0
count_T1 epsilon = -1 not yet implemented, returning 0
nb_points=27
action::init_orthogonal_group after O->init
action::init_orthogonal_group before AO->init
action::init_orthogonal_group we will create the orthogonal group now
action::init_orthogonal_group with reflections, before order_PO_epsilon
order_PO_epsilon
Witt index = 2
order_PO epsilon = -1 m=2 q=2
order_PO_minus 2^(2*3) = 64
order_PO_minus 2^2 - 1 = 3
order_PO_minus 2^4 - 1 = 15
order_PO_minus 2^3 + 1 = 9
order_PO_minus the order of PO^-(6,2) is 51840
order_PO_epsilon  f_semilinear=0 epsilon=-1 k=5 q=2 order=51840
action::init_orthogonal_group the target group order is 51840
action::init_orthogonal_group before create_orthogonal_group
action::init_orthogonal_group after create_orthogonal_group
action::init_orthogonal_group done
The group O^-(6,2) has order 51840 and permutation degree 27
The points on which the group acts are:
0 / 27 : ( 1, 0, 0, 0, 0, 0 )
1 / 27 : ( 0, 1, 0, 0, 0, 0 )
2 / 27 : ( 0, 0, 1, 0, 0, 0 )
3 / 27 : ( 1, 0, 1, 0, 0, 0 )
4 / 27 : ( 0, 1, 1, 0, 0, 0 )
5 / 27 : ( 0, 0, 0, 1, 0, 0 )
6 / 27 : ( 1, 0, 0, 1, 0, 0 )
7 / 27 : ( 0, 1, 0, 1, 0, 0 )
8 / 27 : ( 1, 1, 1, 1, 0, 0 )
9 / 27 : ( 1, 1, 0, 0, 1, 0 )
10 / 27 : ( 1, 1, 1, 0, 1, 0 )
11 / 27 : ( 1, 1, 0, 1, 1, 0 )
12 / 27 : ( 0, 0, 1, 1, 1, 0 )
13 / 27 : ( 1, 0, 1, 1, 1, 0 )
14 / 27 : ( 0, 1, 1, 1, 1, 0 )
15 / 27 : ( 1, 1, 0, 0, 0, 1 )
16 / 27 : ( 1, 1, 1, 0, 0, 1 )
17 / 27 : ( 1, 1, 0, 1, 0, 1 )
18 / 27 : ( 0, 0, 1, 1, 0, 1 )
19 / 27 : ( 1, 0, 1, 1, 0, 1 )
```

```
20 / 27 : ( 0, 1, 1, 1, 0, 1 )
21 / 27 : ( 1, 1, 0, 0, 1, 1 )
22 / 27 : ( 1, 1, 1, 0, 1, 1 )
23 / 27 : ( 1, 1, 0, 1, 1, 1 )
24 / 27 : ( 0, 0, 1, 1, 1, 1 )
25 / 27 : ( 1, 0, 1, 1, 1, 1 )
26 / 27 : ( 0, 1, 1, 1, 1, 1 )
Generators are:
generator 0 / 11 is:
1 0 0 0 0 0
0 1 0 0 0 0
0 0 1 0 0 0
0 0 0 1 0 0
0 0 0 0 1 0
0 0 0 0 1 1
as permutation:
(15, 21)(16, 22)(17, 23)(18, 24)(19, 25)(20, 26)
generator 1 / 11 is:
1 0 0 0 0 0
0 1 0 0 0 0
0 0 1 0 0 0
0 0 0 1 0 0
0 0 0 0 0 1
0 0 0 0 1 0
as permutation:
(9, 15)(10, 16)(11, 17)(12, 18)(13, 19)(14, 20)
generator 2 / 11 is:
1 0 0 0 0 0
0 1 0 0 0 0
0 0 1 0 0 0
0 0 0 1 0 0
0 0 0 0 1 1
0 0 0 0 0 1
as permutation:
(9, 21)(10, 22)(11, 23)(12, 24)(13, 25)(14, 26)
generator 3 / 11 is:
1 0 0 0 0 0
0 1 0 0 0 0
0 0 1 0 0 0
0 0 1 1 1 1
0 0 1 0 0 1
0 0 1 0 1 0
as permutation:
(5, 24)(6, 25)(7, 26)(8, 23)(9, 16)(10, 15)
generator 4 / 11 is:
1 0 0 0 0 0
0 1 0 0 0 0
0 0 1 0 0 0
0 0 1 1 0 1
0 0 1 0 1 1
0 0 1 0 1 0
as permutation:
(5, 18, 24)(6, 19, 25)(7, 20, 26)(8, 17, 23)(9, 22, 16)(10, 21, 15)
generator 5 / 11 is:
1 0 0 0 0 0
0 1 0 0 0 0
0 0 1 1 1 0
0 0 1 1 1 1
0 0 1 0 0 1
0 0 0 1 0 1
as permutation:
(2, 12)(3, 13)(4, 14)(5, 24, 18)(6, 25, 19)(7, 26, 20)(8, 15, 17, 10, 23, 21)(9, 16, 22)
generator 6 / 11 is:
1 0 0 0 0 0
0 1 0 0 0 0
0 0 1 1 0 1
0 0 1 0 0 0
0 0 1 0 1 0
```

```
0 0 1 0 1 1
as permutation:
(2, 18, 12, 24, 5)(3, 19, 13, 25, 6)(4, 20, 14, 26, 7)(8, 17, 21, 15, 22)(9, 10, 23, 16, 11)
generator 7 / 11 is:
1 1 0 1 0 1
0 1 0 0 0 0
0 1 1 0 0 0
0 0 0 1 0 0
0 1 0 0 1 0
0 0 0 0 0 1
as permutation:
(0, 17)(2, 4)(3, 19)(6, 15)(8, 16)(9, 23)(10, 25)(11, 21)(13, 22)(18, 20)
generator 8 / 11 is:
1 1 0 1 1 0
0 1 0 0 0 0
0 1 1 1 1 0
0 0 1 1 1 1
0 0 1 0 0 1
0 1 0 1 0 1
as permutation:
(0, 11)(2, 14)(4, 12)(5, 24, 18)(6, 16, 19, 9, 25, 22)(7, 26, 20)(8, 23, 17)(10, 15, 21)
generator 9 / 11 is:
1 0 0 0 0 0
1 1 1 0 1 1
0 0 1 0 0 0
1 0 1 1 1 1
1 0 1 0 1 0
1 0 1 0 0 1
as permutation:
(1, 22)(4, 21)(5, 25)(6, 24)(9, 15)(10, 16)(11, 14)(12, 18)(13, 19)(17, 20)
generator 10 / 11 is:
1 0 1 1 0 1
0 1 1 1 0 1
1 1 0 0 1 0
1 1 0 0 1 1
1 1 1 0 0 0
0 0 1 1 0 0
as permutation:
(0, 19)(1, 20)(2, 9)(3, 26)(4, 25)(5, 21)(6, 14)(7, 13)(8, 15)(11, 22)(12, 16)(17, 24)
Generators are:
(15, 21)(16, 22)(17, 23)(18, 24)(19, 25)(20, 26)
(9, 15)(10, 16)(11, 17)(12, 18)(13, 19)(14, 20)
(9, 21)(10, 22)(11, 23)(12, 24)(13, 25)(14, 26)
(5, 24)(6, 25)(7, 26)(8, 23)(9, 16)(10, 15)
(5, 18, 24)(6, 19, 25)(7, 20, 26)(8, 17, 23)(9, 22, 16)(10, 21, 15)
(2, 12)(3, 13)(4, 14)(5, 24, 18)(6, 25, 19)(7, 26, 20)(8, 15, 17, 10, 23, 21)(9, 16, 22)
(2, 18, 12, 24, 5)(3, 19, 13, 25, 6)(4, 20, 14, 26, 7)(8, 17, 21, 15, 22)(9, 10, 23, 16, 11)
(0, 17)(2, 4)(3, 19)(6, 15)(8, 16)(9, 23)(10, 25)(11, 21)(13, 22)(18, 20)
(0, 11)(2, 14)(4, 12)(5, 24, 18)(6, 16, 19, 9, 25, 22)(7, 26, 20)(8, 23, 17)(10, 15, 21)
(1, 22)(4, 21)(5, 25)(6, 24)(9, 15)(10, 16)(11, 14)(12, 18)(13, 19)(17, 20)
(0, 19)(1, 20)(2, 9)(3, 26)(4, 25)(5, 21)(6, 14)(7, 13)(8, 15)(11, 22)(12, 16)(17, 24)
Generators in compact permutation form are:
11 27
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 21 22 23 24 25 26 15 16 17 18 19 20
0 1 2 3 4 5 6 7 8 15 16 17 18 19 20 9 10 11 12 13 14 21 22 23 24 25 26
0 1 2 3 4 5 6 7 8 21 22 23 24 25 26 15 16 17 18 19 20 9 10 11 12 13 14
0 1 2 3 4 24 25 26 23 16 15 11 12 13 14 10 9 17 18 19 20 21 22 8 5 6 7
0 1 2 3 4 18 19 20 17 22 21 11 12 13 14 10 9 23 24 25 26 15 16 8 5 6 7
0 1 12 13 14 24 25 26 15 16 23 11 2 3 4 17 22 10 5 6 7 8 9 21 18 19 20
0 1 18 19 20 2 3 4 17 10 23 9 24 25 26 22 11 21 12 13 14 15 8 16 5 6 7
17 1 4 19 2 5 15 7 16 23 25 21 12 22 14 6 8 0 20 3 18 11 13 9 24 10 26
11 1 14 3 12 24 16 26 23 25 15 0 4 13 2 21 19 8 5 9 7 10 6 17 18 22 20
0 22 2 3 21 25 24 7 8 15 16 14 18 19 11 9 10 20 12 13 17 4 1 23 6 5 26
19 20 9 26 25 21 14 13 15 2 10 22 16 7 6 8 12 24 18 0 1 5 11 23 17 4 3
-1
```

# B The program to create orthogonal groups

```
// orthogonal_group.C
//
// Anton Betten
// 10/17/2007
//
//
//
//

#include "orbiter.h"


// global data:

INT t0; // the system time when the program started

void usage(int argc, char **argv);
int main(int argc, char **argv);
void do_it(INT epsilon, INT n, INT q, INT verbose_level);

void usage(int argc, char **argv)
{
   cout << "usage: " << argv[0] << " [options]" << endl;
   cout << "where options can be:" << endl;
   cout << "-v <n>                  : verbose level n" << endl;
   cout << "-epsilon <epsilon>      : set form type epsilon" << endl;
   cout << "-d <d>                  : set dimension d" << endl;
   cout << "-q <q>                  : set field size q" << endl;
}



int main(int argc, char **argv)
{
   INT i;
   INT verbose_level = 0;
   INT f_epsilon = FALSE;
   INT epsilon = 0;
   INT f_d = FALSE;
   INT d = 0;
   INT f_q = FALSE;
   INT q = 0;

   t0 = os_ticks();

   for (i = 1; i < argc; i++) {
      if (strcmp(argv[i], "-v") == 0) {
         verbose_level = atoi(argv[++i]);
         cout << "-v " << verbose_level << endl;
         }
      else if (strcmp(argv[i], "-h") == 0) {
         usage(argc, argv);
         exit(1);
         }
      else if (strcmp(argv[i], "-help") == 0) {
         usage(argc, argv);
         exit(1);
         }
      else if (strcmp(argv[i], "-epsilon") == 0) {
         f_epsilon = TRUE;
         epsilon = atoi(argv[++i]);
         cout << "-epsilon " << epsilon << endl;
         }
      else if (strcmp(argv[i], "-d") == 0) {
         f_d = TRUE;
```

```
            d = atoi(argv[++i]);
            cout << "-d " << d << endl;
            }
        else if (strcmp(argv[i], "-q") == 0) {
            f_q = TRUE;
            q = atoi(argv[++i]);
            cout << "-q " << q << endl;
            }
        }
    if (!f_epsilon) {
        cout << "please use -epsilon <epsilon>" << endl;
        usage(argc, argv);
        exit(1);
        }
    if (!f_d) {
        cout << "please use -d <d>" << endl;
        usage(argc, argv);
        exit(1);
        }
    if (!f_q) {
        cout << "please use -q <q>" << endl;
        usage(argc, argv);
        exit(1);
        }
    do_it(epsilon, d, q, verbose_level);
}

void do_it(INT epsilon, INT n, INT q, INT verbose_level)
{
    finite_field *F;
    action *A;
    INT f_semilinear = FALSE;
    INT f_basis = TRUE;
    INT p, h, i, j, a;
    INT *v;

    A = new action;
    is_prime_power(q, p, h);
    if (h > 1)
        f_semilinear = TRUE;
    else
        f_semilinear = FALSE;

    v = NEW_INT(n);


    F = new finite_field;

    F->init(q, 0);

    A->init_orthogonal_group(epsilon, n, F,
        TRUE /* f_on_points */,
        FALSE /* f_on_lines */,
        FALSE /* f_on_points_and_lines */,
        f_semilinear, f_basis, verbose_level);


    if (!A->f_has_strong_generators) {
        cout << "action does not have strong generators" << endl;
        exit(1);
        }
    strong_generators *SG;
    longinteger_object go;
    action_on_orthogonal *AO = A->G.AO;
    orthogonal *O = AO->O;

    SG = A->Strong_gens;
    SG->group_order(go);
```

```
   cout << "The group " << A->label << " has order " << go << " and permutation degree " << A->degree << endl;
   cout << "The points on which the group acts are:" << endl;
   for (i = 0; i < A->degree; i++) {
      O->unrank_point(v, 1 /* stride */, i, 0 /* verbose_level */);
      cout << i << " / " << A->degree << " : ";
      INT_vec_print(cout, v, n);
      cout << endl;
      }
   cout << "Generators are:" << endl;
   for (i = 0; i < SG->gens->len; i++) {
      cout << "generator " << i << " / " << SG->gens->len << " is: " << endl;
      A->element_print_quick(SG->gens->ith(i), cout);
      cout << "as permutation: " << endl;
      A->element_print_as_permutation(SG->gens->ith(i), cout);
      cout << endl;
      }
   cout << "Generators in compact permutation form are:" << endl;
   cout << SG->gens->len << " " << A->degree << endl;
   for (i = 0; i < SG->gens->len; i++) {
      for (j = 0; j < A->degree; j++) {
         a = A->element_image_of(j, SG->gens->ith(i), 0 /* verbose_level */);
         cout << a << " ";
         }
      cout << endl;
      }
   cout << "-1" << endl;
   FREE_INT(v);
   delete A;
   delete F;
}
```

# C Ovoid search in $O^{-}(6, 2)$

```
~/DEV.18/GITHUB/orbiter/ORBITER/SRC/APPS/OVOID/ovoid.out -v 5 -epsilon -1 -n 5 -q 2 -draw_poset -embedded -W
-v5
-epsilon -1
-n 5
-q 2
-draw_poset
-embedded
epsilon=-1
projective dimension n=5
d=6
q=2
Witt index 2
ovoid_generator::init d=6
ovoid_generator::init f_siegel=1
ovoid_generator::init f_reflection=1
ovoid_generator::init f_similarity=1
ovoid_generator::init f_semisimilarity=1
action::init_orthogonal_group verbose_level=5
action::init_orthogonal_group before A->init_projective_group
action::init_projective_group
n=6 q=2
f_semilinear=0
f_basis=1
action::init_projective_group before M->init_projective_group
matrix_group::init_projective_group
n=6
q=2
f_semilinear=0
```

```
matrix_group::compute_elt_size
bits_per_digit = 1
bits_extension_degree = 0
bits_per_elt = 36
char_per_elt = 5
elt_size_INT_half = 36
elt_size_INT = 72
matrix_group::compute_elt_size done
matrix_group::init_projective_group elt_size_INT = 72
matrix_group::allocate_data
matrix_group::allocate_data done
matrix_group::setup_page_storage
matrix_group::setup_page_storage calling Elts->init()
matrix_group::setup_page_storage calling GL_one()
matrix_group::setup_page_storage calling Elts->store()
identity element stored, hdl = 0
matrix_group::setup_page_storage done
matrix_group::init_projective_group before init_base
matrix_group::init_base
matrix_group::init_base before init_base_projective
matrix_group::init_base after init_base_projective
matrix_group::init_base done
matrix_group::init_projective_group after init_base
matrix_group::init_projective_group finished
action::init_projective_group after M->init_projective_group
action::init_projective_group low_level_point_size=6
action::init_projective_group label=PGL_6_2
action::init_projective_group before setup_linear_group_from_strong_generators
action::setup_linear_group_from_strong_generators setting up a basis
action::setup_linear_group_from_strong_generators before init_matrix_group_strong_generators_builtin
action::init_matrix_group_strong_generators_builtin
action::init_matrix_group_strong_generators_builtin computing strong generators builtin group
n=6
q=2
p=2
e=1
f_semilinear=0
action::init_matrix_group_strong_generators_builtin computing strong generators builtin group finished
action::setup_linear_group_from_strong_generators after init_matrix_group_strong_generators_builtin
action::setup_linear_group_from_strong_generators before S->compute_base_orbits_known_length
action::setup_linear_group_from_strong_generators before S->compute_base_orbits_known_length
sims::compute_base_orbits_known_length: ( 63, 62, 60, 56, 48, 32 )
verbose_level=3
sims::compute_base_orbits_known_length computing level 5
sims::compute_base_orbit_known_length: computing orbit of 5-th base point 5 target_length = 32 nb_gens=5
sims::compute_base_orbit_known_length finished, 5-th base orbit of length 32
sims::compute_base_orbits_known_length level 5 base point 5 orbit length 32 has been computed
sims::compute_base_orbits_known_length computing level 4
sims::compute_base_orbit_known_length: computing orbit of 4-th base point 4 target_length = 48 nb_gens=6
sims::compute_base_orbit_known_length finished, 4-th base orbit of length 48
sims::compute_base_orbits_known_length level 4 base point 4 orbit length 48 has been computed
sims::compute_base_orbits_known_length computing level 3
sims::compute_base_orbit_known_length: computing orbit of 3-th base point 3 target_length = 56 nb_gens=7
sims::compute_base_orbit_known_length finished, 3-th base orbit of length 56
sims::compute_base_orbits_known_length level 3 base point 3 orbit length 56 has been computed
sims::compute_base_orbits_known_length computing level 2
sims::compute_base_orbit_known_length: computing orbit of 2-th base point 2 target_length = 60 nb_gens=8
sims::compute_base_orbit_known_length finished, 2-th base orbit of length 60
sims::compute_base_orbits_known_length level 2 base point 2 orbit length 60 has been computed
sims::compute_base_orbits_known_length computing level 1
sims::compute_base_orbit_known_length: computing orbit of 1-th base point 1 target_length = 62 nb_gens=9
sims::compute_base_orbit_known_length finished, 1-th base orbit of length 62
sims::compute_base_orbits_known_length level 1 base point 1 orbit length 62 has been computed
sims::compute_base_orbits_known_length computing level 0
sims::compute_base_orbit_known_length: computing orbit of 0-th base point 0 target_length = 63 nb_gens=10
sims::compute_base_orbit_known_length finished, 0-th base orbit of length 63
sims::compute_base_orbits_known_length level 0 base point 0 orbit length 63 has been computed
sims::compute_base_orbits_known_length done
```

```
action::setup_linear_group_from_strong_generators after S->compute_base_orbits_known_length
action::setup_linear_group_from_strong_generators before init_sims
action::init_sims action PGL_6_2 base_len = 6
action::init_base_from_sims, base length 6
action::init_sims done
action::setup_linear_group_from_strong_generators after init_sims
action::init_projective_group after setup_linear_group_from_strong_generators
action::init_projective_group, finished setting up PGL_6_2, a permutation group of degree 63 and of order 20158709760
action::init_orthogonal_group before O->init
orthogonal::init: epsilon=-1 n=6 (= vector space dimension) m=2 (= Witt index) q=2 verbose_level=5
choose_anisotropic_form over GF(2)
longinteger_object::create_from_base_10_string() str = 7
object = 7
unipoly_domain::create_object_by_rank_longinteger rank=7
quotient 3 remainder 1
quotient 1 remainder 1
quotient 0 remainder 1
unipoly_domain::create_object_by_rank_string X^{2} + X + 1
choosing the following primitive polynomial:
X^{2} + X + 1
choose_anisotropic_form over GF(2): choosing c1=1, c2=1, c3=1
orthogonal::init computing Gram matrix
orthogonal::init computing Gram matrix done
count_T1 epsilon = -1 not yet implemented, returning 0
T1_m(-1,2,2) = 0
count_T1 epsilon = -1 not yet implemented, returning 0
T1_mm1(-1,1,2) = 0
count_T1 epsilon = -1 not yet implemented, returning 0
T1_mm2(-1,0,2) = 0
T1(2,2) = 0
T1(1,2) = 0
T1(0,2) = 0
T2(2,2) = 6
T2(1,2) = 0
T2(0,2) = 0
nb_pts_N1(2,2) = 6
nb_pts_N1(1,2) = 1
nb_pts_N1(0,2) = 0
S_m=10
S_mm1=3
S_mm2=1
Sbar_m=9
Sbar_mm1=2
Sbar_mm2=0
N1_m=6
N1_mm1=1
N1_mm2=0
nb_points=27
action::init_orthogonal_group after O->init
action::init_orthogonal_group before AO->init
action_on_orthogonal::init
f_on_lines=0
action_on_orthogonal::init degree=27
action_on_orthogonal::init done
action::init_orthogonal_group we will create the orthogonal group now
action::init_orthogonal_group with reflections, before order_PO_epsilon
order_PO_epsilon
Witt index = 2
order_PO epsilon = -1 m=2 q=2
order_PO_minus 2^(2*3) = 64
order_PO_minus 2^2 - 1 = 3
order_PO_minus 2^4 - 1 = 15
order_PO_minus 2^3 + 1 = 9
order_PO_minus the order of PO^-(6,2) is 51840
order_PO_epsilon  f_semilinear=0 epsilon=-1 k=5 q=2 order=51840
action::init_orthogonal_group the target group order is 51840
action::init_orthogonal_group before create_orthogonal_group
action::init_orthogonal_group after create_orthogonal_group
```

```
action::init_orthogonal_group done
The finite field is:
i : inverse(i) : frobenius_power(i, 1) : alpha_power(i) : log_alpha(i)
   0 :   -1 :    0 :    1 :   -1
   1 :    1 :    1 :    1 :    1
addition table:
 0  1
 1  0

multiplication table:
 0  0
 0  1

nb_points=27
nb_lines=0
alpha=140735720683904
depth = 9
generator::init
generator::init sz = 9
generator::init A->degree=27
generator::init A2->degree=27
generator::init sz = 9
generator::init action A:
ACTION O^-(6,2) degree=27 of type action_on_orthogonal_t->matrix_group_t
low_level_point_size=6, f_has_sims=1, f_has_strong_generators=1
base: ( 1, 0, 2, 5, 9, 15 )
has sims
Order 51840 = ( 27, 16, 5, 4, 3, 2 )

generator::init action A2:
ACTION O^-(6,2) degree=27 of type action_on_orthogonal_t->matrix_group_t
low_level_point_size=6, f_has_sims=1, f_has_strong_generators=1
base: ( 1, 0, 2, 5, 9, 15 )
has sims
Order 51840 = ( 27, 16, 5, 4, 3, 2 )

generator::init computing group order
generator::init group order is 51840
generator::init sz = 9
generator::init allocating S of size 9
generator::init allocating Elt_memory
generator::init done
fname_base = ovoid_Q-1_5_2
calling init_oracle with 1000000 nodes
generator::init_oracle
generator::init_oracle done
after calling init_root_node
oracle::init_root_node() initializing root node
storing strong generators
init_root_node done
init() finished
before generator_main
generator::main
generator::main  depth = 9
f_W = 1
f_w = 0
verbose_level = 5
generator::main target_depth=9
generator::main:  calling extend_level 0
we will store schreier vectors for this level
####################################################################################################

generator::extend_level constructing nodes at depth 1
generator::extend_level from 1 nodes at depth 0
verbose_level=3
generator::extend_level 0 calling downstep
####################################################################################################
```

```
downstep depth 0 verbose_level=2
Time 0:00 : Level 0 Node 0 = 0 / 1 :  Downstep node starting

Downstep node finished :  found 27 live points in 1 orbits : progress: 0. 0 %
generator::extend_level after downstep
generator::extend_level calling upstep
generator::upstep
verbose_level = 2
##################################################################################################

extension step depth 0
verbose_level=2
f_indicate_not_canonicals=0
with 1 extension nodes
Time 0:00 : Level 0 Node 0 = 0 / 1 :  Upstep :
Time 0:00 : Level 0 Node 0 = 0 / 1 :  Upstep : progress: 100. 0 %
generator::extend_level after upstep
generator::housekeeping level=1
generator::housekeeping verbose_level=4
generator::housekeeping fname_base=ovoid_Q-1_5_2
##################################################################################################
Found 1 orbits at depth 1
0 : 1 orbits
1 : 1 orbits
total: 2
(1920) average is 1920 + 0 / 1
# 1
1 0 1920 agaaaaaaagaaaaaaabaaaaaaaaaaaaaaacaaaaaaafaaaaaaajaaaaaaapaaaaaabaaaaaaaabaaaaaaafaaaaaaaeaaaaaaadaaaaaaacaaaaa
-1 1 1 in 0:00
(1920) average is 1920 + 0 / 1

# in action O^-(6,2)
generator_housekeeping writing files
generator_housekeeping my_fname_base=ovoid_Q-1_5_2a
we have a swap
generator::housekeeping writing files done
generator_housekeeping not writing tree
generator::housekeeping done
generator::main:  calling extend_level 1
we will store schreier vectors for this level
##################################################################################################

generator::extend_level constructing nodes at depth 2
generator::extend_level from 1 nodes at depth 1
verbose_level=3
generator::extend_level 1 calling downstep
##################################################################################################

downstep depth 1 verbose_level=2
Time 0:00 : Level 1 Node 1 = 0 / 1 :  Downstep node starting

Downstep node finished :  found 16 live points in 1 orbits : progress: 0. 0 %
generator::extend_level after downstep
generator::extend_level size = 1 before write_candidates_binary_using_sv
generator::write_candidates_binary_using_sv lvl=1 fname_base=ovoid_Q-1_5_2
generator::write_candidates_binary_using_sv first node at level 1 is 1
generator::write_candidates_binary_using_sv number of nodes at level 1 is 1
written file ovoid_Q-1_5_2_lvl_1_candidates.bin of size 76
generator::extend_level calling upstep
generator::upstep
verbose_level = 2
##################################################################################################

extension step depth 1
verbose_level=2
f_indicate_not_canonicals=0
with 1 extension nodes
Time 0:00 : Level 1 Node 1 = 0 / 1 :  Upstep :
```

```
Time 0:00 : Level 1 Node 1 = 0 / 1 :  Upstep : progress: 100. 0 %
generator::extend_level after upstep
generator::housekeeping level=2
generator::housekeeping verbose_level=4
generator::housekeeping fname_base=ovoid_Q-1_5_2
##################################################################################################
Found 1 orbits at depth 2
0 : 1 orbits
1 : 1 orbits
2 : 1 orbits
total: 3
(240) average is 240 + 0 / 1
# 2
2 0 1 240 agaaaaaaahaaaaaaabaaaaaaaaaaaaaaacaaaaaaafaaaaaaajaaaaaaapaaaaaaacaaaaaaabaaaaaaafaaaaaaaeaaaaaaadaaaaaaacaaaa
-1 1 2 in 0:00
(240) average is 240 + 0 / 1

# in action O^-(6,2)
generator_housekeeping writing files
generator_housekeeping my_fname_base=ovoid_Q-1_5_2a
generator::housekeeping writing files done
generator_housekeeping not writing tree
generator::housekeeping done
generator::main:  calling extend_level 2
we will store schreier vectors for this level
##################################################################################################

generator::extend_level constructing nodes at depth 3
generator::extend_level from 1 nodes at depth 2
verbose_level=3
generator::extend_level 2 calling downstep
##################################################################################################

downstep depth 2 verbose_level=2
Time 0:00 : Level 2 Node 2 = 0 / 1 :  Downstep node starting

Downstep node finished :  found 10 live points in 1 orbits : progress: 0. 0 %
generator::extend_level after downstep
generator::extend_level size = 2 before write_candidates_binary_using_sv
generator::write_candidates_binary_using_sv lvl=2 fname_base=ovoid_Q-1_5_2
generator::write_candidates_binary_using_sv first node at level 2 is 2
generator::write_candidates_binary_using_sv number of nodes at level 2 is 1
written file ovoid_Q-1_5_2_lvl_2_candidates.bin of size 52
generator::extend_level calling upstep
generator::upstep
verbose_level = 2
##################################################################################################

extension step depth 2
verbose_level=2
f_indicate_not_canonicals=0
with 1 extension nodes
Time 0:00 : Level 2 Node 2 = 0 / 1 :  Upstep :
Time 0:00 : Level 2 Node 2 = 0 / 1 :  Upstep : progress: 100. 0 %
generator::extend_level after upstep
generator::housekeeping level=3
generator::housekeeping verbose_level=4
generator::housekeeping fname_base=ovoid_Q-1_5_2
##################################################################################################
Found 1 orbits at depth 3
0 : 1 orbits
1 : 1 orbits
2 : 1 orbits
3 : 1 orbits
total: 4
(72) average is 72 + 0 / 1
# 3
3 0 1 8 72 agaaaaaaagaaaaaaabaaaaaaaaaaaaaaacaaaaaaafaaaaaaajaaaaaaapaaaaaaadaaaaaaacaaaaaaacaaaaaaabaaaaaaadaaaaaaacaaa
```

```
-1 1 3 in 0:00
(72) average is 72 + 0 / 1

# in action O^-(6,2)
generator_housekeeping writing files
generator_housekeeping my_fname_base=ovoid_Q-1_5_2a
generator::housekeeping writing files done
generator_housekeeping not writing tree
generator::housekeeping done
generator::main:  calling extend_level 3
we will store schreier vectors for this level
##################################################################################################

generator::extend_level constructing nodes at depth 4
generator::extend_level from 1 nodes at depth 3
verbose_level=3
generator::extend_level 3 calling downstep
##################################################################################################

downstep depth 3 verbose_level=2
Time 0:00 : Level 3 Node 3 = 0 / 1 :  Downstep node starting

Downstep node finished :  found 6 live points in 1 orbits : progress: 0. 0 %
generator::extend_level after downstep
generator::extend_level size = 3 before write_candidates_binary_using_sv
generator::write_candidates_binary_using_sv lvl=3 fname_base=ovoid_Q-1_5_2
generator::write_candidates_binary_using_sv first node at level 3 is 3
generator::write_candidates_binary_using_sv number of nodes at level 3 is 1
written file ovoid_Q-1_5_2_lvl_3_candidates.bin of size 36
generator::extend_level calling upstep
generator::upstep
verbose_level = 2
##################################################################################################

extension step depth 3
verbose_level=2
f_indicate_not_canonicals=0
with 1 extension nodes
Time 0:00 : Level 3 Node 3 = 0 / 1 :  Upstep :
Time 0:00 : Level 3 Node 3 = 0 / 1 :  Upstep : progress: 100. 0 %
generator::extend_level after upstep
generator::housekeeping level=4
generator::housekeeping verbose_level=4
generator::housekeeping fname_base=ovoid_Q-1_5_2
##################################################################################################
Found 1 orbits at depth 4
0 : 1 orbits
1 : 1 orbits
2 : 1 orbits
3 : 1 orbits
4 : 1 orbits
total: 5
(48) average is 48 + 0 / 1
# 4
4 0 1 8 10 48 agaaaaaaafaaaaaaabaaaaaaaaaaaaaaacaaaaaaafaaaaaaajaaaaaaapaaaaaaaeaaaaaaadaaaaaaacaaaaaaabaaaaaaabaaaaaaac
-1 1 4 in 0:00
(48) average is 48 + 0 / 1

# in action O^-(6,2)
generator_housekeeping writing files
generator_housekeeping my_fname_base=ovoid_Q-1_5_2a
generator::housekeeping writing files done
generator_housekeeping not writing tree
generator::housekeeping done
generator::main:  calling extend_level 4
we will store schreier vectors for this level
##################################################################################################
```

```
generator::extend_level constructing nodes at depth 5
generator::extend_level from 1 nodes at depth 4
verbose_level=3
generator::extend_level 4 calling downstep
##################################################################################################

downstep depth 4 verbose_level=2
Time 0:00 : Level 4 Node 4 = 0 / 1 :  Downstep node starting

Downstep node finished :  found 3 live points in 2 orbits : progress: 0. 0 %
generator::extend_level after downstep
generator::extend_level size = 4 before write_candidates_binary_using_sv
generator::write_candidates_binary_using_sv lvl=4 fname_base=ovoid_Q-1_5_2
generator::write_candidates_binary_using_sv first node at level 4 is 4
generator::write_candidates_binary_using_sv number of nodes at level 4 is 1
written file ovoid_Q-1_5_2_lvl_4_candidates.bin of size 24
generator::extend_level calling upstep
generator::upstep
verbose_level = 2
##################################################################################################

extension step depth 4
verbose_level=2
f_indicate_not_canonicals=0
with 2 extension nodes
Time 0:00 : Level 4 Node 4 = 0 / 1 :  Upstep :
Time 0:00 : Level 4 Node 4 = 0 / 1 :  Upstep : progress: 100. 0 %
generator::extend_level after upstep
generator::housekeeping level=5
generator::housekeeping verbose_level=4
generator::housekeeping fname_base=ovoid_Q-1_5_2
##################################################################################################
Found 2 orbits at depth 5
0 : 1 orbits
1 : 1 orbits
2 : 1 orbits
3 : 1 orbits
4 : 1 orbits
5 : 2 orbits
total: 7
(240, 120) average is 180 + 0 / 2
# 5
5 0 1 8 10 11 240 agaaaaaaagaaaaaaabaaaaaaaaaaaaaaacaaaaaaafaaaaaaajaaaaaaapaaaaaaafaaaaaaaeaaaaaaadaaaaaaacaaaaaaabaaaa
5 0 1 8 10 16 120 agaaaaaaahaaaaaaabaaaaaaaaaaaaaaacaaaaaaafaaaaaaajaaaaaaapaaaaaaafaaaaaaaeaaaaaaadaaaaaaabaaaaaaacaaaa
-1 2 5 in 0:00
(240, 120) average is 180 + 0 / 2

# in action O^-(6,2)
generator_housekeeping writing files
generator_housekeeping my_fname_base=ovoid_Q-1_5_2a
generator::housekeeping writing files done
generator_housekeeping not writing tree
generator::housekeeping done
generator::main:  calling extend_level 5
we will store schreier vectors for this level
##################################################################################################

generator::extend_level constructing nodes at depth 6
generator::extend_level from 2 nodes at depth 5
verbose_level=3
generator::extend_level 5 calling downstep
##################################################################################################

downstep depth 5 verbose_level=2
Time 0:00 : Level 5 Node 5 = 0 / 2 :  Downstep node starting

Downstep node finished :  found 0 live points in 0 orbits : progress: 0. 0 %
generator::extend_level after downstep
```

```
generator::extend_level size = 5 before write_candidates_binary_using_sv
generator::write_candidates_binary_using_sv lvl=5 fname_base=ovoid_Q-1_5_2
generator::write_candidates_binary_using_sv first node at level 5 is 5
generator::write_candidates_binary_using_sv number of nodes at level 5 is 2
written file ovoid_Q-1_5_2_lvl_5_candidates.bin of size 24
generator::extend_level calling upstep
generator::upstep
verbose_level = 2
###############################################################################################

extension step depth 5
verbose_level=2
f_indicate_not_canonicals=0
with 1 extension nodes
Time 0:00 : Level 5 Node 5 = 0 / 2 :  Upstep :
Time 0:00 : Level 5 Node 5 = 0 / 2 :  Upstep : progress: 0. 0 %
generator::extend_level after upstep
generator::housekeeping level=6
generator::housekeeping verbose_level=4
generator::housekeeping fname_base=ovoid_Q-1_5_2
###############################################################################################
Found 1 orbits at depth 6
0 : 1 orbits
1 : 1 orbits
2 : 1 orbits
3 : 1 orbits
4 : 1 orbits
5 : 2 orbits
6 : 1 orbits
total: 8
(720) average is 720 + 0 / 1
# 6
6 0 1 8 10 16 22 720 agaaaaaahaaaaaaabaaaaaaaaaaaaaaacaaaaaaafaaaaaaajaaaaaaapaaaaaaagaaaaaaafaaaaaaaeaaaaaaabaaaaaaada
-1 1 7 in 0:00
(720) average is 720 + 0 / 1

# in action O^-(6,2)
generator_housekeeping writing files
generator_housekeeping my_fname_base=ovoid_Q-1_5_2a
generator::housekeeping writing files done
generator_housekeeping not writing tree
generator::housekeeping done
generator::main:  calling extend_level 6
we will store schreier vectors for this level
###############################################################################################

generator::extend_level constructing nodes at depth 7
generator::extend_level from 1 nodes at depth 6
verbose_level=3
generator::extend_level 6 calling downstep
###############################################################################################

downstep depth 6 verbose_level=2
Time 0:00 : Level 6 Node 7 = 0 / 1 :  Downstep node starting

Downstep node finished :  found 0 live points in 0 orbits : progress: 0. 0 %
generator::extend_level after downstep
generator::extend_level size = 6 before write_candidates_binary_using_sv
generator::write_candidates_binary_using_sv lvl=6 fname_base=ovoid_Q-1_5_2
generator::write_candidates_binary_using_sv first node at level 6 is 7
generator::write_candidates_binary_using_sv number of nodes at level 6 is 1
written file ovoid_Q-1_5_2_lvl_6_candidates.bin of size 12
generator::extend_level calling upstep
generator::upstep
verbose_level = 2
###############################################################################################

extension step depth 6
```

```
verbose_level=2
f_indicate_not_canonicals=0
with 0 extension nodes
Time 0:00 : Level 6 Node 7 = 0 / 1 :  Upstep :
Time 0:00 : Level 6 Node 7 = 0 / 1 :  Upstep : progress: -92233720368547758.-8 %
generator::extend_level after upstep
generator::housekeeping level=7
generator::housekeeping verbose_level=4
generator::housekeeping fname_base=ovoid_Q-1_5_2
####################################################################################################
Found 0 orbits at depth 7
0 : 1 orbits
1 : 1 orbits
2 : 1 orbits
3 : 1 orbits
4 : 1 orbits
5 : 2 orbits
6 : 1 orbits
7 : 0 orbits
total: 8
() average is 9 + 0 / 0
# 7
-1 0 8 in 0:00
() average is 9 + 0 / 0

# in action O^-(6,2)
generator_housekeeping writing files
generator_housekeeping my_fname_base=ovoid_Q-1_5_2a
generator::housekeeping writing files done
generator_housekeeping not writing tree
generator::housekeeping done
generator::draw_poset data=0
generator::draw_poset before make_auxiliary_graph
generator::draw_poset before make_graph
generator::draw_poset before make_graph
generator::draw_poset before make_poset_graph_detailed
generator::draw_poset after make_poset_graph_detailed
generator::draw_poset writing file ovoid_-1_6_2_aux_poset_lvl_9.layered_graph
generator::draw_poset writing file ovoid_-1_6_2_poset_lvl_9.layered_graph
generator::draw_poset writing file ovoid_-1_6_2_tree_lvl_9.layered_graph
generator::draw_poset writing file ovoid_-1_6_2_poset_detailed_lvl_9.layered_graph
generator::draw_poset done
0:00

```

```
MY_PATH=~/DEV.18
SRC=$(MY_PATH)/GITHUB/orbiter/ORBITER/SRC
SRC2=$(MY_PATH)/ORBITER2/SRC2

OVOID_PATH=$(SRC)/APPS/OVOID
TOOLS_PATH=$(SRC)/APPS/TOOLS


Op42:
   $(OVOID_PATH)/ovoid.out -v 2 -epsilon 1 -d 4 -q 2

Om42:
   $(OVOID_PATH)/ovoid.out -v 2 -epsilon -1 -d 4 -q 2

Op62:
   $(OVOID_PATH)/ovoid.out -v 2 -epsilon 1 -d 6 -q 2

Om62:
   $(OVOID_PATH)/ovoid.out -v 5 -epsilon -1 -n 5 -q 2 -draw_poset -embedded -W

# order 25920, degree 27 = U_4(2) = Weyl group of type E_6 = Sp_4(2) = O_5(3)


```

```
draw:
   $(TOOLS_PATH)/layered_graph_main.out -v 4 \
      -file ovoid_-1_6_2_poset_lvl_9.layered_graph \
      -draw test \
      -rad 25000 \
      -xin 1000000 \
      -yin 1000000 \
      -xout 1000000 \
      -yout 1000000 \
      -embedded \
      -scale .44 \
      -line_width 1.0 \
      -y_stretch 0.8
   pdflatex ovoid_-1_6_2_poset_lvl_9_draw.tex
   open ovoid_-1_6_2_poset_lvl_9_draw.pdf
```

# References


[1] Anton Betten. Classifying discrete objects with Orbiter. ACM Communications in Computer Algebra, Vol. 47, No. 4, Issue 186, December 2013.

[2] Anton Betten. Orbiter – a program to classify discrete objects. `https://github.com/abetten/orbiter`, 2013-2018.

[3] Anton Betten. How fast can we compute orbits of groups? In *ICMS 2018—Proceedings of the International Congress on Mathematical Software; James H. Davenport, Manuel Kauers, George Labahn, Josef Urban (ed.)*, pages 62–70. Springer, 2018.

[4] J. H. Conway and N. J. A. Sloane. *Sphere packings, lattices and groups*, volume 290 of *Grundlehren der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences]*. Springer-Verlag, New York, third edition, 1999. With additional contributions by E. Bannai, R. E. Borcherds, J. Leech, S. P. Norton, A. M. Odlyzko, R. A. Parker, L. Queen and B. B. Venkov.

[5] I. A. Faradžev. Constructive enumeration of combinatorial objects. In *Problèmes combinatoires et théorie des graphes (Colloq. Internat. CNRS, Univ. Orsay, Orsay, 1976)*, volume 260 of *Colloq. Internat. CNRS*, pages 131–135. CNRS, Paris, 1978.

[6] P. Kaski and P. Östergård. *Classification algorithms for codes and designs*, volume 15 of *Algorithms and Computation in Mathematics*. Springer-Verlag, Berlin, 2006.

[7] Jeffrey S. Leon. Permutation group algorithms based on partitions. I. Theory and algorithms. *J. Symbolic Comput.*, 12(4-5):533–583, 1991. Computational group theory, Part 2.

[8] Jeffrey S. Leon. Partitions, refinements, and permutation group computation. In *Groups and computation, II (New Brunswick, NJ, 1995)*, volume 28 of *DIMACS Ser. Discrete Math. Theoret. Comput. Sci.*, pages 123–158. Amer. Math. Soc., Providence, RI, 1997.

[9] Brendan D. McKay. Isomorph-free exhaustive generation. *J. Algorithms*, 26(2):306–324, 1998.

[10] Wilhelm Plesken. Counting with groups and rings. *J. Reine Angew. Math.*, 334:40–68, 1982.

[11] Ronald C. Read. Every one a winner or how to avoid isomorphism search when cataloguing combinatorial configurations. *Ann. Discrete Math.*, 2:107–120, 1978. Algorithmic aspects of combinatorics (Conf., Vancouver Island, B.C., 1976).

[12] Gordon F. Royle. An orderly algorithm and some applications in finite geometry. *Discrete Math.*, 185(1-3):105–115, 1998.

[13] Bernd Schmalz. Verwendung von Untergruppenleitern zur Bestimmung von Doppelnebenklassen. *Bayreuth. Math. Schr.*, (31):109–143, 1990.