

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. According to this division, this user's guide will be split up into two parts. The first part is for users who want to use Orbiter through predefined applications. The second part is more in-depth, and is targeted at users who want to program Orbiter. This introductory section provides some background on general Orbiter design issues, and hence is useful for both types of users.

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 $T = (t_1, \dots, t_n)$ 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 $t_i \in T$ such 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 $t_i \in T$ such that $x \sim t_i$ and in addition 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 $g := h_1h_2^{-1}$ 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 $\{1, \dots, 24\}$ 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 working on needed to have good “ladders” of subgroups. A ladder of subgroups is a sequences of subgroups G_1, G_2, \dots, G_r where two consecutive groups are related. This means that for $i = 1, \dots, r - 1$ 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 $\text{PG}(n-1, q)$ 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 $\{0, \dots, d-1\}$ is used, for some d .

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 available, 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 to 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 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

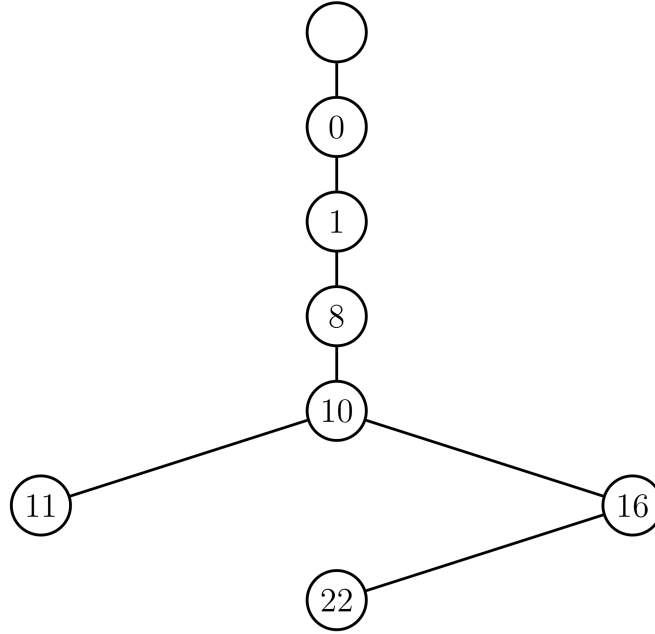
$$x_0x_1 + x_2x_3 + x_4^2 + x_4x_5 + x_5^2 = 0.$$

Generators for the group as 6×6 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 $S = \{P_1, \dots, P_n\}$ 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

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

4 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

```

1      ~/DEV.18/GITHUB/orbiter/ORBITER/SRC/APPS/GROUPS/orthogonal_group.out -v 2 -epsilon -1 -d 6 -q 2
2      -v 2
3      -epsilon -1
4      -d 6
5      -q 2
6      action::init_orthogonal_group verbose_level=2
7      action::init_orthogonal_group before A->init_projective_group
8      action::init_orthogonal_group before O->init
9      orthogonal::init: epsilon=-1 n=6 (= vector space dimension) m=2 (= Witt index) q=2 verbose_level=2
10     choose_anisotropic_form over GF(2)
11     longinteger_object::create_from_base_10_string() str = 7
12     object = 7
13     unipoly_domain::create_object_by_rank_longinteger rank=7
14     quotient 3 remainder 1
15     quotient 1 remainder 1
16     quotient 0 remainder 1
17     unipoly_domain::create_object_by_rank_string X^{2} + X + 1
18     choosing the following primitive polynomial:
19     X^{2} + X + 1
20     choose_anisotropic_form over GF(2): choosing c1=1, c2=1, c3=1
21     orthogonal::init computing Gram matrix
22     orthogonal::init computing Gram matrix done
23     count_T1 epsilon = -1 not yet implemented, returning 0
24     count_T1 epsilon = -1 not yet implemented, returning 0
25     count_T1 epsilon = -1 not yet implemented, returning 0
26     nb_points=27
27     action::init_orthogonal_group after O->init
28     action::init_orthogonal_group before A0->init
29     action::init_orthogonal_group we will create the orthogonal group now
30     action::init_orthogonal_group with reflections, before order_PO_epsilon
31     order_PO_epsilon
32     Witt index = 2
33     order_PO epsilon = -1 m=2 q=2
34     order_PO_minus 2^(2*3) = 64
35     order_PO_minus 2^2 - 1 = 3
36     order_PO_minus 2^4 - 1 = 15
37     order_PO_minus 2^3 + 1 = 9
38     order_PO_minus the order of PO^{-(6,2)} is 51840
39     order_PO_epsilon f.semilinear=0 epsilon=-1 k=5 q=2 order=51840
40     action::init_orthogonal_group the target group order is 51840
41     action::init_orthogonal_group before create_orthogonal_group
42     action::init_orthogonal_group after create_orthogonal_group
43     action::init_orthogonal_group done
44     The group  $O^-(6, 2)$  has order 51840 and permutation degree 27
45     The points on which the group acts are:
46     0 / 27 : ( 1, 0, 0, 0, 0, 0 )
47     1 / 27 : ( 0, 1, 0, 0, 0, 0 )
48     2 / 27 : ( 0, 0, 1, 0, 0, 0 )
49     3 / 27 : ( 1, 0, 1, 0, 0, 0 )
50     4 / 27 : ( 0, 1, 1, 0, 0, 0 )
51     5 / 27 : ( 0, 0, 0, 1, 0, 0 )
52     6 / 27 : ( 1, 0, 0, 1, 0, 0 )
53     7 / 27 : ( 0, 1, 0, 1, 0, 0 )
54     8 / 27 : ( 1, 1, 1, 1, 0, 0 )
55     9 / 27 : ( 1, 1, 0, 0, 1, 0 )
56     10 / 27 : ( 1, 1, 1, 0, 1, 0 )
57     11 / 27 : ( 1, 1, 0, 1, 1, 0 )
58     12 / 27 : ( 0, 0, 1, 1, 1, 0 )
59     13 / 27 : ( 1, 0, 1, 1, 1, 0 )
60     14 / 27 : ( 0, 1, 1, 1, 1, 0 )
61     15 / 27 : ( 1, 1, 0, 0, 0, 1 )
62     16 / 27 : ( 1, 1, 1, 0, 0, 1 )
63     17 / 27 : ( 1, 1, 0, 1, 0, 1 )
64     18 / 27 : ( 0, 0, 1, 1, 0, 1 )
65     19 / 27 : ( 1, 0, 1, 1, 0, 1 )

```

```

66 20 / 27 : ( 0, 1, 1, 1, 0, 1 )
67 21 / 27 : ( 1, 1, 0, 0, 1, 1 )
68 22 / 27 : ( 1, 1, 1, 0, 1, 1 )
69 23 / 27 : ( 1, 1, 0, 1, 1, 1 )
70 24 / 27 : ( 0, 0, 1, 1, 1, 1 )
71 25 / 27 : ( 1, 0, 1, 1, 1, 1 )
72 26 / 27 : ( 0, 1, 1, 1, 1, 1 )
73 Generators are:
74 generator 0 / 11 is:
75 1 0 0 0 0 0
76 0 1 0 0 0 0
77 0 0 1 0 0 0
78 0 0 0 1 0 0
79 0 0 0 0 1 0
80 0 0 0 0 1 1
81 as permutation:
82 (15, 21)(16, 22)(17, 23)(18, 24)(19, 25)(20, 26)
83 generator 1 / 11 is:
84 1 0 0 0 0 0
85 0 1 0 0 0 0
86 0 0 1 0 0 0
87 0 0 0 1 0 0
88 0 0 0 0 0 1
89 0 0 0 0 1 0
90 as permutation:
91 (9, 15)(10, 16)(11, 17)(12, 18)(13, 19)(14, 20)
92 generator 2 / 11 is:
93 1 0 0 0 0 0
94 0 1 0 0 0 0
95 0 0 1 0 0 0
96 0 0 0 1 0 0
97 0 0 0 0 1 1
98 0 0 0 0 0 1
99 as permutation:
100 (9, 21)(10, 22)(11, 23)(12, 24)(13, 25)(14, 26)
101 generator 3 / 11 is:
102 1 0 0 0 0 0
103 0 1 0 0 0 0
104 0 0 1 0 0 0
105 0 0 1 1 1 1
106 0 0 1 0 0 1
107 0 0 1 0 1 0
108 as permutation:
109 (5, 24)(6, 25)(7, 26)(8, 23)(9, 16)(10, 15)
110 generator 4 / 11 is:
111 1 0 0 0 0 0
112 0 1 0 0 0 0
113 0 0 1 0 0 0
114 0 0 1 1 0 1
115 0 0 1 0 1 1
116 0 0 1 0 1 0
117 as permutation:
118 (5, 18, 24)(6, 19, 25)(7, 20, 26)(8, 17, 23)(9, 22, 16)(10, 21, 15)
119 generator 5 / 11 is:
120 1 0 0 0 0 0
121 0 1 0 0 0 0
122 0 0 1 1 1 0
123 0 0 1 1 1 1
124 0 0 1 0 0 1
125 0 0 0 1 0 1
126 as permutation:
127 (2, 12)(3, 13)(4, 14)(5, 24, 18)(6, 25, 19)(7, 26, 20)(8, 15, 17, 10, 23, 21)(9, 16, 22)
128 generator 6 / 11 is:
129 1 0 0 0 0 0
130 0 1 0 0 0 0
131 0 0 1 1 0 1
132 0 0 1 0 0 0
133 0 0 1 0 1 0

```

```

134 0 0 1 0 1 1
135 as permutation:
136 (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)
137 generator 7 / 11 is:
138 1 1 0 1 0 1
139 0 1 0 0 0 0
140 0 1 1 0 0 0
141 0 0 0 1 0 0
142 0 1 0 0 1 0
143 0 0 0 0 0 1
144 as permutation:
145 (0, 17)(2, 4)(3, 19)(6, 15)(8, 16)(9, 23)(10, 25)(11, 21)(13, 22)(18, 20)
146 generator 8 / 11 is:
147 1 1 0 1 1 0
148 0 1 0 0 0 0
149 0 1 1 1 1 0
150 0 0 1 1 1 1
151 0 0 1 0 0 1
152 0 1 0 1 0 1
153 as permutation:
154 (0, 11)(2, 14)(4, 12)(5, 24, 18)(6, 16, 19, 9, 25, 22)(7, 26, 20)(8, 23, 17)(10, 15, 21)
155 generator 9 / 11 is:
156 1 0 0 0 0 0
157 1 1 1 0 1 1
158 0 0 1 0 0 0
159 1 0 1 1 1 1
160 1 0 1 0 1 0
161 1 0 1 0 0 1
162 as permutation:
163 (1, 22)(4, 21)(5, 25)(6, 24)(9, 15)(10, 16)(11, 14)(12, 18)(13, 19)(17, 20)
164 generator 10 / 11 is:
165 1 0 1 1 0 1
166 0 1 1 1 0 1
167 1 1 0 0 1 0
168 1 1 0 0 1 1
169 1 1 1 0 0 0
170 0 0 1 1 0 0
171 as permutation:
172 (0, 19)(1, 20)(2, 9)(3, 26)(4, 25)(5, 21)(6, 14)(7, 13)(8, 15)(11, 22)(12, 16)(17, 24)
173 Generators are:
174 (15, 21)(16, 22)(17, 23)(18, 24)(19, 25)(20, 26)
175 (9, 15)(10, 16)(11, 17)(12, 18)(13, 19)(14, 20)
176 (9, 21)(10, 22)(11, 23)(12, 24)(13, 25)(14, 26)
177 (5, 24)(6, 25)(7, 26)(8, 23)(9, 16)(10, 15)
178 (5, 18, 24)(6, 19, 25)(7, 20, 26)(8, 17, 23)(9, 22, 16)(10, 21, 15)
179 (2, 12)(3, 13)(4, 14)(5, 24, 18)(6, 25, 19)(7, 26, 20)(8, 15, 17, 10, 23, 21)(9, 16, 22)
180 (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)
181 (0, 17)(2, 4)(3, 19)(6, 15)(8, 16)(9, 23)(10, 25)(11, 21)(13, 22)(18, 20)
182 (0, 11)(2, 14)(4, 12)(5, 24, 18)(6, 16, 19, 9, 25, 22)(7, 26, 20)(8, 23, 17)(10, 15, 21)
183 (1, 22)(4, 21)(5, 25)(6, 24)(9, 15)(10, 16)(11, 14)(12, 18)(13, 19)(17, 20)
184 (0, 19)(1, 20)(2, 9)(3, 26)(4, 25)(5, 21)(6, 14)(7, 13)(8, 15)(11, 22)(12, 16)(17, 24)
185 Generators in compact permutation form are:
186 11 27
187 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
188 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
189 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
190 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
191 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
192 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
193 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
194 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
195 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
196 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
197 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
198 -1
199

```


B The program to create orthogonal groups

```
1 // orthogonal_group.C
2 //
3 // Anton Betten
4 // 10/17/2007
5 //
6 //
7 //
8 //
9
10 #include "orbiter.h"
11
12
13 // global data:
14
15 INT t0; // the system time when the program started
16
17 void usage(int argc, char **argv);
18 int main(int argc, char **argv);
19 void do_it(INT epsilon, INT n, INT q, INT verbose_level);
20
21 void usage(int argc, char **argv)
22 {
23     cout << "usage: " << argv[0] << " [options]" << endl;
24     cout << "where options can be:" << endl;
25     cout << "-v <n>                : verbose level n" << endl;
26     cout << "-epsilon <epsilon>        : set form type epsilon" << endl;
27     cout << "-d <d>                  : set dimension d" << endl;
28     cout << "-q <q>                  : set field size q" << endl;
29 }
30
31
32
33 int main(int argc, char **argv)
34 {
35     INT i;
36     INT verbose_level = 0;
37     INT f_epsilon = FALSE;
38     INT epsilon = 0;
39     INT f_d = FALSE;
40     INT d = 0;
41     INT f_q = FALSE;
42     INT q = 0;
43
44     t0 = os_ticks();
45
46     for (i = 1; i < argc; i++) {
47         if (strcmp(argv[i], "-v") == 0) {
48             verbose_level = atoi(argv[++i]);
49             cout << "-v " << verbose_level << endl;
50         }
51         else if (strcmp(argv[i], "-h") == 0) {
52             usage(argc, argv);
53             exit(1);
54         }
55         else if (strcmp(argv[i], "-help") == 0) {
56             usage(argc, argv);
57             exit(1);
58         }
59         else if (strcmp(argv[i], "-epsilon") == 0) {
60             f_epsilon = TRUE;
61             epsilon = atoi(argv[++i]);
62             cout << "-epsilon " << epsilon << endl;
63         }
64         else if (strcmp(argv[i], "-d") == 0) {
65             f_d = TRUE;
```

```

66         d = atoi(argv[++i]);
67         cout << "-d " << d << endl;
68     }
69     else if (strcmp(argv[i], "-q") == 0) {
70         f_q = TRUE;
71         q = atoi(argv[++i]);
72         cout << "-q " << q << endl;
73     }
74 }
75 if (!f_epsilon) {
76     cout << "please use -epsilon <epsilon>" << endl;
77     usage(argc, argv);
78     exit(1);
79 }
80 if (!f_d) {
81     cout << "please use -d <d>" << endl;
82     usage(argc, argv);
83     exit(1);
84 }
85 if (!f_q) {
86     cout << "please use -q <q>" << endl;
87     usage(argc, argv);
88     exit(1);
89 }
90 do_it(epsilon, d, q, verbose_level);
91 }
92
93 void do_it(INT epsilon, INT n, INT q, INT verbose_level)
94 {
95     finite_field *F;
96     action *A;
97     INT f_semlinear = FALSE;
98     INT f_basis = TRUE;
99     INT p, h, i, j, a;
100    INT *v;
101
102    A = new action;
103    is_prime_power(q, p, h);
104    if (h > 1)
105        f_semlinear = TRUE;
106    else
107        f_semlinear = FALSE;
108
109    v = NEW_INT(n);
110
111
112    F = new finite_field;
113
114    F->init(q, 0);
115
116    A->init_orthogonal_group(epsilon, n, F,
117        TRUE /* f_on_points */,
118        FALSE /* f_on_lines */,
119        FALSE /* f_on_points_and_lines */,
120        f_semlinear, f_basis, verbose_level);
121
122
123    if (!A->f_has_strong_generators) {
124        cout << "action does not have strong generators" << endl;
125        exit(1);
126    }
127    strong_generators *SG;
128    longinteger_object go;
129    action_on_orthogonal *AO = A->G.AO;
130    orthogonal *O = AO->O;
131
132    SG = A->Strong_gens;
133    SG->group_order(go);

```

```

134
135     cout << "The group " << A->label << " has order " << go << " and permutation degree " << A->degree << endl;
136     cout << "The points on which the group acts are:" << endl;
137     for (i = 0; i < A->degree; i++) {
138         O->unrank_point(v, 1 /* stride */, i, 0 /* verbose_level */);
139         cout << i << " / " << A->degree << " : ";
140         INT_vec_print(cout, v, n);
141         cout << endl;
142     }
143     cout << "Generators are:" << endl;
144     for (i = 0; i < SG->gens->len; i++) {
145         cout << "generator " << i << " / " << SG->gens->len << " is: " << endl;
146         A->element_print_quick(SG->gens->ith(i), cout);
147         cout << "as permutation: " << endl;
148         A->element_print_as_permutation(SG->gens->ith(i), cout);
149         cout << endl;
150     }
151     cout << "Generators in compact permutation form are:" << endl;
152     cout << SG->gens->len << " " << A->degree << endl;
153     for (i = 0; i < SG->gens->len; i++) {
154         for (j = 0; j < A->degree; j++) {
155             a = A->element_image_of(j, SG->gens->ith(i), 0 /* verbose_level */);
156             cout << a << " ";
157         }
158         cout << endl;
159     }
160     cout << "-1" << endl;
161     FREE_INT(v);
162     delete A;
163     delete F;
164 }
165
166
167

```

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

```

1 ~/DEV.18/GITHUB/orbiter/ORBITER/SRC/APPS/OVOID/ovoid.out -v 5 -epsilon -1 -n 5 -q 2 -draw_poset -embedded -W
2 -v5
3 -epsilon -1
4 -n 5
5 -q 2
6 -draw_poset
7 -embedded
8 epsilon=-1
9 projective dimension n=5
10 d=6
11 q=2
12 Witt index 2
13 ovoid_generator::init d=6
14 ovoid_generator::init f.siegel=1
15 ovoid_generator::init f.reflection=1
16 ovoid_generator::init f.similarity=1
17 ovoid_generator::init f.semisimilarity=1
18 action::init_orthogonal_group verbose_level=5
19 action::init_orthogonal_group before A->init_projective_group
20 action::init_projective_group
21 n=6 q=2
22 f_semilinear=0
23 f_basis=1
24 action::init_projective_group before M->init_projective_group
25 matrix_group::init_projective_group
26 n=6
27 q=2
28 f_semilinear=0

```

```

29     matrix_group::compute_elt_size
30     bits_per_digit = 1
31     bits_extension_degree = 0
32     bits_per_elt = 36
33     char_per_elt = 5
34     elt_size_INT_half = 36
35     elt_size_INT = 72
36     matrix_group::compute_elt_size done
37     matrix_group::init_projective_group elt_size_INT = 72
38     matrix_group::allocate_data
39     matrix_group::allocate_data done
40     matrix_group::setup_page_storage
41     matrix_group::setup_page_storage calling Elts->init()
42     matrix_group::setup_page_storage calling GL_one()
43     matrix_group::setup_page_storage calling Elts->store()
44     identity element stored, hdl = 0
45     matrix_group::setup_page_storage done
46     matrix_group::init_projective_group before init_base
47     matrix_group::init_base
48     matrix_group::init_base before init_base_projective
49     matrix_group::init_base after init_base_projective
50     matrix_group::init_base done
51     matrix_group::init_projective_group after init_base
52     matrix_group::init_projective_group finished
53     action::init_projective_group after M->init_projective_group
54     action::init_projective_group low_level_point.size=6
55     action::init_projective_group label=PGL_6.2
56     action::init_projective_group before setup_linear_group_from_strong_generators
57     action::setup_linear_group_from_strong_generators setting up a basis
58     action::setup_linear_group_from_strong_generators before init_matrix_group_strong_generators_builtin
59     action::init_matrix_group_strong_generators_builtin
60     action::init_matrix_group_strong_generators_builtin computing strong generators builtin group
61     n=6
62     q=2
63     p=2
64     e=1
65     f_semilinear=0
66     action::init_matrix_group_strong_generators_builtin computing strong generators builtin group finished
67     action::setup_linear_group_from_strong_generators after init_matrix_group_strong_generators_builtin
68     action::setup_linear_group_from_strong_generators before S->compute_base_orbits_known_length
69     action::setup_linear_group_from_strong_generators before S->compute_base_orbits_known_length
70     sims::compute_base_orbits_known_length: ( 63, 62, 60, 56, 48, 32 )
71     verbose_level=3
72     sims::compute_base_orbits_known_length computing level 5
73     sims::compute_base_orbit_known_length: computing orbit of 5-th base point 5 target_length = 32 nb_gens=5
74     sims::compute_base_orbit_known_length finished, 5-th base orbit of length 32
75     sims::compute_base_orbits_known_length level 5 base point 5 orbit length 32 has been computed
76     sims::compute_base_orbits_known_length computing level 4
77     sims::compute_base_orbit_known_length: computing orbit of 4-th base point 4 target_length = 48 nb_gens=6
78     sims::compute_base_orbit_known_length finished, 4-th base orbit of length 48
79     sims::compute_base_orbits_known_length level 4 base point 4 orbit length 48 has been computed
80     sims::compute_base_orbits_known_length computing level 3
81     sims::compute_base_orbit_known_length: computing orbit of 3-th base point 3 target_length = 56 nb_gens=7
82     sims::compute_base_orbit_known_length finished, 3-th base orbit of length 56
83     sims::compute_base_orbits_known_length level 3 base point 3 orbit length 56 has been computed
84     sims::compute_base_orbits_known_length computing level 2
85     sims::compute_base_orbit_known_length: computing orbit of 2-th base point 2 target_length = 60 nb_gens=8
86     sims::compute_base_orbit_known_length finished, 2-th base orbit of length 60
87     sims::compute_base_orbits_known_length level 2 base point 2 orbit length 60 has been computed
88     sims::compute_base_orbits_known_length computing level 1
89     sims::compute_base_orbit_known_length: computing orbit of 1-th base point 1 target_length = 62 nb_gens=9
90     sims::compute_base_orbit_known_length finished, 1-th base orbit of length 62
91     sims::compute_base_orbits_known_length level 1 base point 1 orbit length 62 has been computed
92     sims::compute_base_orbits_known_length computing level 0
93     sims::compute_base_orbit_known_length: computing orbit of 0-th base point 0 target_length = 63 nb_gens=10
94     sims::compute_base_orbit_known_length finished, 0-th base orbit of length 63
95     sims::compute_base_orbits_known_length level 0 base point 0 orbit length 63 has been computed
96     sims::compute_base_orbits_known_length done

```

```

97  action::setup_linear_group_from_strong_generators after S->compute_base_orbits_known_length
98  action::setup_linear_group_from_strong_generators before init_sims
99  action::init_sims action PGL_6_2 base_len = 6
100 action::init_base_from_sims, base length 6
101 action::init_sims done
102 action::setup_linear_group_from_strong_generators after init_sims
103 action::init_projective_group after setup_linear_group_from_strong_generators
104 action::init_projective_group, finished setting up PGL_6_2, a permutation group of degree 63 and of order 20158709760
105 action::init_orthogonal_group before 0->init
106 orthogonal::init: epsilon=-1 n=6 (= vector space dimension) m=2 (= Witt index) q=2 verbose_level=5
107 choose_anisotropic_form over GF(2)
108 longinteger_object::create_from_base_10_string() str = 7
109 object = 7
110 unipoly_domain::create_object_by_rank_longinteger rank=7
111 quotient 3 remainder 1
112 quotient 1 remainder 1
113 quotient 0 remainder 1
114 unipoly_domain::create_object_by_rank_string X^{2} + X + 1
115 choosing the following primitive polynomial:
116 X^{2} + X + 1
117 choose_anisotropic_form over GF(2): choosing c1=1, c2=1, c3=1
118 orthogonal::init computing Gram matrix
119 orthogonal::init computing Gram matrix done
120 count_T1 epsilon = -1 not yet implemented, returning 0
121 T1_m(-1,2,2) = 0
122 count_T1 epsilon = -1 not yet implemented, returning 0
123 T1_mm1(-1,1,2) = 0
124 count_T1 epsilon = -1 not yet implemented, returning 0
125 T1_mm2(-1,0,2) = 0
126 T1(2,2) = 0
127 T1(1,2) = 0
128 T1(0,2) = 0
129 T2(2,2) = 6
130 T2(1,2) = 0
131 T2(0,2) = 0
132 nb_pts_N1(2,2) = 6
133 nb_pts_N1(1,2) = 1
134 nb_pts_N1(0,2) = 0
135 S_m=10
136 S_mm1=3
137 S_mm2=1
138 Sbar_m=9
139 Sbar_mm1=2
140 Sbar_mm2=0
141 N1_m=6
142 N1_mm1=1
143 N1_mm2=0
144 nb_points=27
145 action::init_orthogonal_group after 0->init
146 action::init_orthogonal_group before A0->init
147 action_on_orthogonal::init
148 f_on_lines=0
149 action_on_orthogonal::init degree=27
150 action_on_orthogonal::init done
151 action::init_orthogonal_group we will create the orthogonal group now
152 action::init_orthogonal_group with reflections, before order_P0_epsilon
153 order_P0_epsilon
154 Witt index = 2
155 order_P0_epsilon = -1 m=2 q=2
156 order_P0_minus 2^(2*3) = 64
157 order_P0_minus 2^2 - 1 = 3
158 order_P0_minus 2^4 - 1 = 15
159 order_P0_minus 2^3 + 1 = 9
160 order_P0_minus the order of P0^{-(6,2)} is 51840
161 order_P0_epsilon f_semiinear=0 epsilon=-1 k=5 q=2 order=51840
162 action::init_orthogonal_group the target group order is 51840
163 action::init_orthogonal_group before create_orthogonal_group
164 action::init_orthogonal_group after create_orthogonal_group

```

```

165 action::init_orthogonal_group done
166 The finite field is:
167 i : inverse(i) : frobenius.power(i, 1) : alpha.power(i) : log_alpha(i)
168 0 : -1 : 0 : 1 : -1
169 1 : 1 : 1 : 1 : 1
170 addition table:
171 0 1
172 1 0
173
174 multiplication table:
175 0 0
176 0 1
177
178 nb_points=27
179 nb_lines=0
180 alpha=140735720683904
181 depth = 9
182 generator::init
183 generator::init sz = 9
184 generator::init A->degree=27
185 generator::init A2->degree=27
186 generator::init sz = 9
187 generator::init action A:
188 ACTION 0~-(6,2) degree=27 of type action_on_orthogonal_t->matrix_group_t
189 low_level.point.size=6, f_has_sims=1, f_has_strong_generators=1
190 base: ( 1, 0, 2, 5, 9, 15 )
191 has sims
192 Order 51840 = ( 27, 16, 5, 4, 3, 2 )
193
194 generator::init action A2:
195 ACTION 0~-(6,2) degree=27 of type action_on_orthogonal_t->matrix_group_t
196 low_level.point.size=6, f_has_sims=1, f_has_strong_generators=1
197 base: ( 1, 0, 2, 5, 9, 15 )
198 has sims
199 Order 51840 = ( 27, 16, 5, 4, 3, 2 )
200
201 generator::init computing group order
202 generator::init group order is 51840
203 generator::init sz = 9
204 generator::init allocating S of size 9
205 generator::init allocating Elt_memory
206 generator::init done
207 fname_base = ovoid.Q-1.5.2
208 calling init_oracle with 1000000 nodes
209 generator::init_oracle
210 generator::init_oracle done
211 after calling init_root_node
212 oracle::init_root_node() initializing root node
213 storing strong generators
214 init_root_node done
215 init() finished
216 before generator_main
217 generator::main
218 generator::main depth = 9
219 f_W = 1
220 f_w = 0
221 verbose_level = 5
222 generator::main target_depth=9
223 generator::main: calling extend_level 0
224 we will store schreier vectors for this level
225 #####
226
227 generator::extend_level constructing nodes at depth 1
228 generator::extend_level from 1 nodes at depth 0
229 verbose_level=3
230 generator::extend_level 0 calling downstep
231 #####
232

```

```

233 downstep depth 0 verbose_level=2
234 Time 0:00 : Level 0 Node 0 = 0 / 1 : Downstep node starting
235
236 Downstep node finished : found 27 live points in 1 orbits : progress: 0. 0 %
237 generator::extend_level after downstep
238 generator::extend_level calling upstep
239 generator::upstep
240 verbose_level = 2
241 #####
242
243 extension step depth 0
244 verbose_level=2
245 f_indicate_not_canonicals=0
246 with 1 extension nodes
247 Time 0:00 : Level 0 Node 0 = 0 / 1 : Upstep :
248 Time 0:00 : Level 0 Node 0 = 0 / 1 : Upstep : progress: 100. 0 %
249 generator::extend_level after upstep
250 generator::housekeeping level=1
251 generator::housekeeping verbose_level=4
252 generator::housekeeping fname_base=ovoid.Q-1.5.2
253 #####
254 Found 1 orbits at depth 1
255 0 : 1 orbits
256 1 : 1 orbits
257 total: 2
258 (1920) average is 1920 + 0 / 1
259 # 1
260 1 0 1920 agaaaaaaagaaaaabaaaaaaaaaaaaaaaacaaaaaafaaaaaaajaaaaaapaaaaabaaaaaaabaaaaaafaaaaaaeaaaaaadaaaaaacaaaaa
261 -1 1 1 in 0:00
262 (1920) average is 1920 + 0 / 1
263
264 # in action 0^-(6,2)
265 generator.housekeeping writing files
266 generator.housekeeping my_fname_base=ovoid.Q-1.5.2a
267 we have a swap
268 generator::housekeeping writing files done
269 generator.housekeeping not writing tree
270 generator::housekeeping done
271 generator::main: calling extend_level 1
272 we will store schreier vectors for this level
273 #####
274
275 generator::extend_level constructing nodes at depth 2
276 generator::extend_level from 1 nodes at depth 1
277 verbose_level=3
278 generator::extend_level 1 calling downstep
279 #####
280
281 downstep depth 1 verbose_level=2
282 Time 0:00 : Level 1 Node 1 = 0 / 1 : Downstep node starting
283
284 Downstep node finished : found 16 live points in 1 orbits : progress: 0. 0 %
285 generator::extend_level after downstep
286 generator::extend_level size = 1 before write_candidates.binary_using_sv
287 generator::write_candidates_binary_using_sv lvl=1 fname_base=ovoid.Q-1.5.2
288 generator::write_candidates_binary_using_sv first node at level 1 is 1
289 generator::write_candidates_binary_using_sv number of nodes at level 1 is 1
290 written file ovoid.Q-1.5.2.lvl.1.candidates.bin of size 76
291 generator::extend_level calling upstep
292 generator::upstep
293 verbose_level = 2
294 #####
295
296 extension step depth 1
297 verbose_level=2
298 f_indicate_not_canonicals=0
299 with 1 extension nodes
300 Time 0:00 : Level 1 Node 1 = 0 / 1 : Upstep :

```

```

301 Time 0:00 : Level 1 Node 1 = 0 / 1 : Upstep : progress: 100. 0 %
302 generator::extend_level after upstep
303 generator::housekeeping level=2
304 generator::housekeeping verbose_level=4
305 generator::housekeeping fname_base=ovoid.Q-1.5.2
306 #####
307 Found 1 orbits at depth 2
308 0 : 1 orbits
309 1 : 1 orbits
310 2 : 1 orbits
311 total: 3
312 (240) average is 240 + 0 / 1
313 # 2
314 2 0 1 240 agaaaaaaahaaaaaaabaaaaaaaaaaaaacaaaaaaafaaaaaaajaaaaaaapaaaaaacaaaaabaaaaaafaaaaaaeaaaaadaaaaaacaaa
315 -1 1 2 in 0:00
316 (240) average is 240 + 0 / 1
317
318 # in action 0~-(6,2)
319 generator_housekeeping writing files
320 generator_housekeeping my_fname_base=ovoid.Q-1.5.2a
321 generator::housekeeping writing files done
322 generator_housekeeping not writing tree
323 generator::housekeeping done
324 generator::main: calling extend_level 2
325 we will store schreier vectors for this level
326 #####
327
328 generator::extend_level constructing nodes at depth 3
329 generator::extend_level from 1 nodes at depth 2
330 verbose_level=3
331 generator::extend_level 2 calling downstep
332 #####
333
334 downstep depth 2 verbose_level=2
335 Time 0:00 : Level 2 Node 2 = 0 / 1 : Downstep node starting
336
337 Downstep node finished : found 10 live points in 1 orbits : progress: 0. 0 %
338 generator::extend_level after downstep
339 generator::extend_level size = 2 before write_candidates.binary_using_sv
340 generator::write_candidates_binary_using_sv lvl=2 fname_base=ovoid.Q-1.5.2
341 generator::write_candidates_binary_using_sv first node at level 2 is 2
342 generator::write_candidates_binary_using_sv number of nodes at level 2 is 1
343 written file ovoid.Q-1.5.2_lvl2_candidates.bin of size 52
344 generator::extend_level calling upstep
345 generator::upstep
346 verbose_level = 2
347 #####
348
349 extension step depth 2
350 verbose_level=2
351 f_indicate_not_canonicals=0
352 with 1 extension nodes
353 Time 0:00 : Level 2 Node 2 = 0 / 1 : Upstep :
354 Time 0:00 : Level 2 Node 2 = 0 / 1 : Upstep : progress: 100. 0 %
355 generator::extend_level after upstep
356 generator::housekeeping level=3
357 generator::housekeeping verbose_level=4
358 generator::housekeeping fname_base=ovoid.Q-1.5.2
359 #####
360 Found 1 orbits at depth 3
361 0 : 1 orbits
362 1 : 1 orbits
363 2 : 1 orbits
364 3 : 1 orbits
365 total: 4
366 (72) average is 72 + 0 / 1
367 # 3
368 3 0 1 8 72 agaaaaaaagaaaaaaabaaaaaaaaaaaaacaaaaaaafaaaaaaajaaaaaaapaaaaaadaaaaaacaaaaaacaaaaabaaaaaadaaaaaacaaa

```



```

369 -1 1 3 in 0:00
370 (72) average is 72 + 0 / 1
371
372 # in action 0~-(6,2)
373 generator.housekeeping writing files
374 generator.housekeeping my_fname.base=ovoid.Q-1.5.2a
375 generator::housekeeping writing files done
376 generator.housekeeping not writing tree
377 generator::housekeeping done
378 generator::main: calling extend_level 3
379 we will store schreier vectors for this level
380 #####
381
382 generator::extend_level constructing nodes at depth 4
383 generator::extend_level from 1 nodes at depth 3
384 verbose_level=3
385 generator::extend_level 3 calling downstep
386 #####
387
388 downstep depth 3 verbose_level=2
389 Time 0:00 : Level 3 Node 3 = 0 / 1 : Downstep node starting
390
391 Downstep node finished : found 6 live points in 1 orbits : progress: 0. 0 %
392 generator::extend_level after downstep
393 generator::extend_level size = 3 before write_candidates.binary_using_sv
394 generator::write_candidates_binary_using_sv lvl=3 fname.base=ovoid.Q-1.5.2
395 generator::write_candidates_binary_using_sv first node at level 3 is 3
396 generator::write_candidates_binary_using_sv number of nodes at level 3 is 1
397 written file ovoid.Q-1.5.2_lvl3_candidates.bin of size 36
398 generator::extend_level calling upstep
399 generator::upstep
400 verbose_level = 2
401 #####
402
403 extension step depth 3
404 verbose_level=2
405 f_indicate_not_canonicals=0
406 with 1 extension nodes
407 Time 0:00 : Level 3 Node 3 = 0 / 1 : Upstep :
408 Time 0:00 : Level 3 Node 3 = 0 / 1 : Upstep : progress: 100. 0 %
409 generator::extend_level after upstep
410 generator::housekeeping level=4
411 generator::housekeeping verbose_level=4
412 generator::housekeeping fname.base=ovoid.Q-1.5.2
413 #####
414 Found 1 orbits at depth 4
415 0 : 1 orbits
416 1 : 1 orbits
417 2 : 1 orbits
418 3 : 1 orbits
419 4 : 1 orbits
420 total: 5
421 (48) average is 48 + 0 / 1
422 # 4
423 4 0 1 8 10 48 agaaaaaafaaaaaabaaaaaaaaaaaaacaaaaaafaaaaaaajaaaaaapaaaaaaeaaaaaadaaaaaacaaaaabaaaaabaaaaaaac
424 -1 1 4 in 0:00
425 (48) average is 48 + 0 / 1
426
427 # in action 0~-(6,2)
428 generator.housekeeping writing files
429 generator.housekeeping my_fname.base=ovoid.Q-1.5.2a
430 generator::housekeeping writing files done
431 generator.housekeeping not writing tree
432 generator::housekeeping done
433 generator::main: calling extend_level 4
434 we will store schreier vectors for this level
435 #####
436

```

```

437 generator::extend_level constructing nodes at depth 5
438 generator::extend_level from 1 nodes at depth 4
439 verbose_level=3
440 generator::extend_level 4 calling downstep
441 #####
442
443 downstep depth 4 verbose_level=2
444 Time 0:00 : Level 4 Node 4 = 0 / 1 : Downstep node starting
445
446 Downstep node finished : found 3 live points in 2 orbits : progress: 0. 0 %
447 generator::extend_level after downstep
448 generator::extend_level size = 4 before write_candidates.binary_using_sv
449 generator::write_candidates.binary_using_sv lvl=4 fname.base=ovoid.Q-1.5.2
450 generator::write_candidates.binary_using_sv first node at level 4 is 4
451 generator::write_candidates.binary_using_sv number of nodes at level 4 is 1
452 written file ovoid.Q-1.5.2.lvl.4.candidates.bin of size 24
453 generator::extend_level calling upstep
454 generator::upstep
455 verbose_level = 2
456 #####
457
458 extension step depth 4
459 verbose_level=2
460 f_indicate_not_canonicals=0
461 with 2 extension nodes
462 Time 0:00 : Level 4 Node 4 = 0 / 1 : Upstep :
463 Time 0:00 : Level 4 Node 4 = 0 / 1 : Upstep : progress: 100. 0 %
464 generator::extend_level after upstep
465 generator::housekeeping level=5
466 generator::housekeeping verbose_level=4
467 generator::housekeeping fname.base=ovoid.Q-1.5.2
468 #####
469 Found 2 orbits at depth 5
470 0 : 1 orbits
471 1 : 1 orbits
472 2 : 1 orbits
473 3 : 1 orbits
474 4 : 1 orbits
475 5 : 2 orbits
476 total: 7
477 (240, 120) average is 180 + 0 / 2
478 # 5
479 5 0 1 8 10 11 240 agaaaaaaagaaaaaaabaaaaaaaaaaaaaaaaacaaaaaaafaaaaaaajaaaaaaapaaaaaaafaaaaaaeaaaaaadaaaaaacaaaaaabaaa
480 5 0 1 8 10 16 120 agaaaaaaahaaaaaaabaaaaaaaaaaaaaaaaacaaaaaaafaaaaaaajaaaaaaapaaaaaaafaaaaaaeaaaaaadaaaaaabaaaaaacaaa
481 -1 2 5 in 0:00
482 (240, 120) average is 180 + 0 / 2
483
484 # in action 0~(6,2)
485 generator.housekeeping writing files
486 generator.housekeeping my_fname.base=ovoid.Q-1.5.2a
487 generator::housekeeping writing files done
488 generator.housekeeping not writing tree
489 generator::housekeeping done
490 generator::main: calling extend_level 5
491 we will store schreier vectors for this level
492 #####
493
494 generator::extend_level constructing nodes at depth 6
495 generator::extend_level from 2 nodes at depth 5
496 verbose_level=3
497 generator::extend_level 5 calling downstep
498 #####
499
500 downstep depth 5 verbose_level=2
501 Time 0:00 : Level 5 Node 5 = 0 / 2 : Downstep node starting
502
503 Downstep node finished : found 0 live points in 0 orbits : progress: 0. 0 %
504 generator::extend_level after downstep

```

```

505 generator::extend_level size = 5 before write_candidates.binary_using_sv
506 generator::write_candidates.binary_using_sv lvl=5 fname_base=ovoid.Q-1.5.2
507 generator::write_candidates.binary_using_sv first node at level 5 is 5
508 generator::write_candidates.binary_using_sv number of nodes at level 5 is 2
509 written file ovoid.Q-1.5.2_lvl.5_candidates.bin of size 24
510 generator::extend_level calling upstep
511 generator::upstep
512 verbose_level = 2
513 #####
514
515 extension step depth 5
516 verbose_level=2
517 f_indicate_not_canonicals=0
518 with 1 extension nodes
519 Time 0:00 : Level 5 Node 5 = 0 / 2 : Upstep :
520 Time 0:00 : Level 5 Node 5 = 0 / 2 : Upstep : progress: 0. 0 %
521 generator::extend_level after upstep
522 generator::housekeeping level=6
523 generator::housekeeping verbose_level=4
524 generator::housekeeping fname_base=ovoid.Q-1.5.2
525 #####
526 Found 1 orbits at depth 6
527 0 : 1 orbits
528 1 : 1 orbits
529 2 : 1 orbits
530 3 : 1 orbits
531 4 : 1 orbits
532 5 : 2 orbits
533 6 : 1 orbits
534 total: 8
535 (720) average is 720 + 0 / 1
536 # 6
537 6 0 1 8 10 16 22 720 agaaaaaaahaaaaabaaaaaaacaaaaaafaaaaaaajaaaaaaapaaaaaaagaaaaaafaaaaaaeaaaaaabaaaaaad
538 -1 1 7 in 0:00
539 (720) average is 720 + 0 / 1
540
541 # in action 0^(6,2)
542 generator.housekeeping writing files
543 generator.housekeeping my_fname_base=ovoid.Q-1.5.2a
544 generator::housekeeping writing files done
545 generator.housekeeping not writing tree
546 generator::housekeeping done
547 generator::main: calling extend_level 6
548 we will store schreier vectors for this level
549 #####
550
551 generator::extend_level constructing nodes at depth 7
552 generator::extend_level from 1 nodes at depth 6
553 verbose_level=3
554 generator::extend_level 6 calling downstep
555 #####
556
557 downstep depth 6 verbose_level=2
558 Time 0:00 : Level 6 Node 7 = 0 / 1 : Downstep node starting
559
560 Downstep node finished : found 0 live points in 0 orbits : progress: 0. 0 %
561 generator::extend_level after downstep
562 generator::extend_level size = 6 before write_candidates.binary_using_sv
563 generator::write_candidates.binary_using_sv lvl=6 fname_base=ovoid.Q-1.5.2
564 generator::write_candidates.binary_using_sv first node at level 6 is 7
565 generator::write_candidates.binary_using_sv number of nodes at level 6 is 1
566 written file ovoid.Q-1.5.2_lvl.6_candidates.bin of size 12
567 generator::extend_level calling upstep
568 generator::upstep
569 verbose_level = 2
570 #####
571
572 extension step depth 6

```

```

573 verbose_level=2
574 f_indicate_not_canonicals=0
575 with 0 extension nodes
576 Time 0:00 : Level 6 Node 7 = 0 / 1 : Upstep :
577 Time 0:00 : Level 6 Node 7 = 0 / 1 : Upstep : progress: -92233720368547758.-8 %
578 generator::extend_level after upstep
579 generator::housekeeping level=7
580 generator::housekeeping verbose_level=4
581 generator::housekeeping fname_base=ovoid_Q-1.5.2
582 #####
583 Found 0 orbits at depth 7
584 0 : 1 orbits
585 1 : 1 orbits
586 2 : 1 orbits
587 3 : 1 orbits
588 4 : 1 orbits
589 5 : 2 orbits
590 6 : 1 orbits
591 7 : 0 orbits
592 total: 8
593 () average is 9 + 0 / 0
594 # 7
595 -1 0 8 in 0:00
596 () average is 9 + 0 / 0
597
598 # in action 0^(6,2)
599 generator.housekeeping writing files
600 generator.housekeeping my_fname_base=ovoid_Q-1.5.2a
601 generator::housekeeping writing files done
602 generator.housekeeping not writing tree
603 generator::housekeeping done
604 generator::draw_poset data=0
605 generator::draw_poset before make_auxiliary_graph
606 generator::draw_poset before make_graph
607 generator::draw_poset before make_graph
608 generator::draw_poset before make_poset_graph.detailed
609 generator::draw_poset after make_poset_graph.detailed
610 generator::draw_poset writing file ovoid-1.6.2.aux_poset_lvl.9.layered_graph
611 generator::draw_poset writing file ovoid-1.6.2.poset_lvl.9.layered_graph
612 generator::draw_poset writing file ovoid-1.6.2.tree_lvl.9.layered_graph
613 generator::draw_poset writing file ovoid-1.6.2.poset_detailed_lvl.9.layered_graph
614 generator::draw_poset done
615 0:00
616

1 MY_PATH=~/.DEV.18
2 SRC=$(MY_PATH)/GITHUB/orbiter/ORBITER/SRC
3 SRC2=$(MY_PATH)/ORBITER2/SRC2
4
5 OVOID_PATH=$(SRC)/APPS/OVOID
6 TOOLS_PATH=$(SRC)/APPS/TOOLS
7
8
9 Op42:
10 $(OVOID_PATH)/ovoid.out -v 2 -epsilon 1 -d 4 -q 2
11
12 Om42:
13 $(OVOID_PATH)/ovoid.out -v 2 -epsilon -1 -d 4 -q 2
14
15 Op62:
16 $(OVOID_PATH)/ovoid.out -v 2 -epsilon 1 -d 6 -q 2
17
18 Om62:
19 $(OVOID_PATH)/ovoid.out -v 5 -epsilon -1 -n 5 -q 2 -draw_poset -embedded -W
20
21 # order 25920, degree 27 = U.4(2) = Weyl group of type E.6 = Sp.4(2) = 0.5(3)
22
23

```

```

24 draw:
25     $(TOOLS_PATH)/layered_graph_main.out -v 4 \
26         -file ovoid-1.6.2_poset_lvl.9.layered_graph \
27         -draw test \
28         -rad 25000 \
29         -xin 1000000 \
30         -yin 1000000 \
31         -xout 1000000 \
32         -yout 1000000 \
33         -embedded \
34         -scale .44 \
35         -line_width 1.0 \
36         -y_stretch 0.8
37 pdflatex ovoid-1.6.2_poset_lvl.9.draw.tex
38 open ovoid-1.6.2_poset_lvl.9.draw.pdf
39
40

```

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.