# The Simple 7-(33,8,10)-Designs with Automorphism Group PΓL(2,32)

Alfred Wassermann

**Abstract**

Lattice basis reduction in combination with an efficient backtracking algorithm is used to find all (4 996 426) simple 7-(33,8,10) designs with automorphism group PΓL(2,32). The paper contains a short description of the algorithm.

## 1 Introduction

Let $X$ be a $v$-set (i.e. a set with $v$ elements) whose elements are called *points*. A $t$-$(v,k,\lambda)$ *design* is a collection of $k$-subsets (called *blocks*) of $X$ with the property that any $t$-subset of $X$ is contained in exactly $\lambda$ blocks. A $t$-$(v,k,\lambda)$ design is called *simple* if no blocks are repeated, and *trivial* if every $k$-subset of $X$ is a block and occurs the same number of times in the design.

A straightforward approach to the construction of $t$-$(v,k,\lambda)$ designs is to consider the matrix

$$M^v_{t,k} := (m_{i,j}), \quad i = 1,\dots,\binom{v}{t},\ j = 1,\dots,\binom{v}{k}:$$

The rows of $M^v_{t,k}$ are indexed by the $t$-subsets of $X$ and the columns by the $k$-subsets of $X$. We set $m_{i,j} := 1$ if the $i$-th $t$-subset is contained in the $j$-th $k$-subset, otherwise $m_{i,j} := 0$. Simple $t$-$(v,k,\lambda)$ designs therefore correspond to $\{0,1\}$-solutions $x$ of the system of $\binom{v}{t}$ linear equations:

$$M^v_{t,k} \cdot x = \lambda(1,1,\dots,1)^\top.$$

Unfortunately, for most designs with interesting parameters $v, t, k$ the size of the matrix $M^v_{t,k}$ is prohibitively large. For example in the case of $v = 33$, $t = 7$ and $k = 8$ the matrix $M^{33}_{7,8}$ has 4 272 048 rows and 13 884 156 columns.

But by assuming a group action on the set $X$ the size of $M^v_{t,k}$ can be dramatically reduced. A group $G$ acting on $X$ induces also an action on the set of $t$-subsets and the set of $k$-subsets of $X$. With $A_{t,k} = (a_{i,j})$ we denote the matrix where $a_{ij}$ counts the number of those elements in the $j$-th orbit of $G$ on the $k$-subsets of $X$ which contain a representative of the $i$-th orbit of $t$-subsets of $X$. This matrix was introduced by Kramer and Mesner [7]. They observed:

**Theorem 1** *(see [7]) A simple $t$-$(v, k, \lambda)$ design with $G \leq \mathrm{Sym}(X)$ as an automorphism group exists if and only if there is a $\{0, 1\}$-solution $x$ to the matrix equation*

$$A_{t,k} \cdot x = \lambda(1, 1, \ldots, 1)^\top. \tag{1}$$

Taking the group $\mathrm{P\Gamma L}(2, 2^5)$ the matrix $A_{7,8}$ in the above example has 32 rows and 97 columns. Nevertheless it is still a respectable task to find solutions of (1).

Finding solutions for this problem requires algorithms which do searching in high dimensional spaces. These algorithms can roughly be divided into two classes, depending on whether they search in a systematic manner for all possible solutions or if they just try to find one solution.

For finding just one solution the algorithms are mostly randomized, for example simulated annealing, combinatorial optimization, local search [13] and lattice basis reduction [8, 15, 1]. See [13] for a survey. Recently also algorithms which use Gröbner bases have been proposed [21, 22].

In [8] the authors used the original lattice basis reduction algorithm (LLL) as described in [11] and a lattice like the one proposed in [9]. Meanwhile lattice basis reduction algorithms have been greatly improved. New algorithms were invented by Schnorr, [17, 18, 19]. Also new lattices have been proposed, see [2, 4].

On the contrary, in order to find <u>all</u> $\{0, 1\}$-solutions of (1) until now only exhaustive search techniques based on backtracking have been used, see for example [13, 14]. SCHMALZ [16] used a graph theoretical approach, he enumerates all solutions implicitely via graphs.

The new approach – using lattice basis reduction [11] – is to construct a basis of the kernel of the equation

$$\begin{pmatrix} & & 1 \\ & A_{t,k} & \vdots \\ & & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}, \quad x_i \in \mathbf{Z}, y \in \mathbf{Z} \tag{2}$$

which consists of short integer vectors. But the shortest integer vectors (in the euclidean norm) in the kernel of (2) need not correspond to solutions of our $\{0,1\}$-problem (1). KAIB and RITTER proposed in [5] an algorithm which enumerates all solutions with $y = \pm\lambda$ as linear combination of this short integer basis vectors.

The first step of this algorithm – finding a basis of the kernel – can be done in polynomial time in the number of columns of the Kramer Mesner matrix $A_{tk}$ with the help of lattice basis reduction [11]. But the explicit enumeration in the second step of [5] is still an exponential algorithm whilst in most cases much faster than the brute force enumeration as it was used in the above mentioned algorithms. Thus in some sense this algorithm combines the two classes of algorithms to solve (1).

This is the first announcement of the 4 996 246 7-(33,8,10)-designs with automorphism group PΓL(2,32) together with a short overview of the algorithm. A more detailed description will be submitted to the *Journal of Combinatorial Designs*.

## 2 From linear equations to lattices

As in [1] we transform the Kramer Mesner matrix $A_{tk}$ with $l$ rows and $s$ columns into the matrix

$$\left(\begin{array}{ccc|cc} & & & c_0 1 & 0 \\ & c_0 A_{tk} & & \vdots & \vdots \\ & & & c_0 1 & 0 \\ \hline c_1 2 & & 0 & 0 & c_1 1 \\ & \ddots & & \vdots & \vdots \\ 0 & & c_1 2 & 0 & c_1 1 \\ \hline 0 & \dots & 0 & 1 & 0 \\ 0 & \dots & 0 & 0 & c_1 1 \end{array}\right) \tag{3}$$

containing $s+2$ column vectors with $l+s+2$ rows. The set of all integer linear combinations of these vectors is called a *lattice*. A minimal set of vectors which generates the lattice is called a *basis* of the lattice. Important in our context are bases which contain short vectors. These are called *reduced bases* if they fulfill certain criteria of shortness, see [11].

The lattice $L$ spanned by the columns of the matrix (3) has the column vectors of the matrix itself as a basis. This basis is reduced with the algorithm proposed in [20] to a new basis.

**Definition 1** *Let $L \subset \mathbf{R}^n$ be a lattice. For $1 \leq p < \infty$ the norm defined by the mapping*

$$\|.\|_p : \mathbf{R}^n \to \mathbf{R},\ x \mapsto \|x\|_p := (\sum_{i=1}^{n} |x_i|^p)^{1/p}$$

*is called p-norm. The norm defined by the mapping*

$$\|.\|_\infty : \mathbf{R}^n \to \mathbf{R},\ x \mapsto \|x\|_\infty := \max\{|x_i| \mid 1 \leq i \leq n\}$$

*is called $\infty$-norm.*
*For $1 \leq p \leq \infty$ we call a vector $\in L$ $p$-*shortest* if it is a shortest vector in $L$ in $p$-norm.*

If we set in (3) $c_1 = \lambda$ and $c_0 > \lambda$, the $\infty$-shortest vectors of the lattice are solutions of the equation (1). $\infty$-shortest vectors in $L$ consist of zeros in the first $l$ rows and have only the entries $-1 \cdot c_1$ or $1 \cdot c_1$ in the rows $l+1, \ldots, l+s$. Further, in row $l+s+1$ and $l+s+2$ they contain $\pm\lambda$ and $\pm 1$, respectively.

Until now only reduction techniques for the norm $p = 2$ are working efficiently. To find a $\infty$-shortest vector we have to employ backtracking methods. Since the 2-norm and the $\infty$-norm are equivalent (for all $x \in \mathbf{R}^n$: $\|x\|_\infty \leq \|x\|_2 \leq \sqrt{n}\|x\|_\infty$) it's reasonable to use 2-short vectors to find the $\infty$-shortest vectors.

Let $\langle .,. \rangle$ denote the ordinary inner product in $\mathbf{R}^n$, $n \in \mathbf{N}$. For a sequence of linear independent vectors $b_1, \ldots, b_m \in \mathbf{R}^n$ we let $b_1^*, \ldots, b_m^*$ be the *Gram-Schmidt orthogonalized* sequence. We thus have

$$b_i^* := b_i - \sum_{j=1}^{i-1} \mu_{i,j} b_j^* \quad \text{for } i = 1, \ldots, m, \text{ where } \mu_{i,j} = \frac{\langle b_i, b_j^* \rangle}{\langle b_j^*, b_j^* \rangle} \ . \tag{4}$$

**Definition 2** *For an (ordered) basis $b_1, b_2, \ldots, b_m$ of a lattice $L \subset \mathbf{R}^n$ and $1 \leq i \leq m$, $\pi_i(v)$ is the* orthogonal projection *of $v \in \mathbf{R}^n$ into $\langle b_1, b_2, \ldots, b_i \rangle^\perp$. $L_i := \pi_i(L)$ is the orthogonal projection of the lattice $L$ into $\langle b_1, b_2, \ldots, b_i \rangle^\perp$.*

Since

$$v = \sum_{j=1}^{m} \langle v, b_j^* \rangle b_j^*$$

we see that

$$\pi_i(v) = \sum_{j=i}^{m} \langle v, b_j^* \rangle b_j^*. \tag{5}$$

# 3 Explicit enumeration

For every basis vector $b_t$ and $j \leq t$ we have:

$$\pi_j(b_t) = \sum_{i=j}^{t} \mu_{t,i} b_i^*.$$

With $c_s := \|b_s^*\|_2^2$ for $1 \le s \le k$ it follows

$$\pi_j(\sum_{t=j}^{m} u_t b_t) = (\sum_{i=j}^{m} u_i \mu_{i,j}) b_j^* + \pi_{j+1}(\sum_{t=j+1}^{m} u_t b_t), \tag{6}$$

and

$$\|\pi_j(\sum_{t=j}^{m} u_t b_t)\|_2^2 = (\sum_{i=j}^{m} u_i \mu_{i,j})^2 c_j + \|\pi_{j+1}(\sum_{t=j+1}^{m} u_t b_t)\|_2^2.$$

**Definition 3** *For $u_j, u_{j+1}, \ldots, u_m \in \mathbf{Z}$ we write $w_j := \pi_j(\sum_{t=j}^{m} u_t b_t)$.*

The backtracking algorithm tries all possible integer values for $u_m$, $u_{m-1}$, $\ldots, u_1$. Starting from $t = m$ it computes $w_t$ for $m \ge t \ge 1$ and finally $w_1 = \sum_{i=1}^{m} u_i b_i$.

**Remark 1** *If $u_{j+1}, u_{j+2}, \ldots, u_m \in \mathbf{Z}$ are fixed and $u_j \in \mathbf{Z}$ has to be choosen such that $\|w_j\|_2^2$ is minimal, then $u_j$ has to be set to the nearest integer to $-\sum_{i=j+1}^{m} u_i \mu_{i,j}$, since*

$$\|w_j\|_2^2 = \|\pi_j(\sum_{t=j}^{m} u_t b_t)\|_2^2 = (u_j + \sum_{i=j+1}^{m} u_i \mu_{i,j})^2 c_j + \|\pi_{j+1}(\sum_{t=j+1}^{m} u_t b_t)\|_2^2.$$

The solutions of our system of linear equations (2) are the $\infty$-shortest vectors in the lattice generated by the vectors in (3), but we describe the search for the $p$-shortest vector in $L$ for arbitrary $1 \le p \le \infty$.

Let $F$ be an upper bound of the $p$-shortest vector of $L$. Since all $p$-norms in $\mathbf{R}^n$ are equivalent, there exist constants $r_p, R_p$ such that $r_p\|x\|_p \le \|x\|_2 \le R_p\|x\|_p$ for all $x \in \mathbf{R}^n$. Therefore a $p$-shortest vector $v$ has 2-norm $\|v\|_2 \le R_p F$ and in order to find $p$-shortest vectors we enumerate all vectors with 2-norm not greater than $R_p F$.

Moreover, Kaib, Ritter [5] use Hölder's inequality to combine the search for $p$-shortest vectors with enumeration in 2-norm:

**Theorem 2** *If for fixed $u_j, u_{j+1}, \ldots, u_m \in \mathbf{Z}$ there exist*

$$u_1, u_2, \ldots, u_{j-1} \in \mathbf{Z}$$

*with $\|\sum_{i=1}^{m} w_i\|_p \le F$, then for all $y_j, y_{j+1}, \ldots, y_m \in \mathbf{R}$:*

$$\left|\sum_{i=j}^{m} y_i \|w_i\|_2^2\right| \le F \cdot \|\sum_{i=j}^{m} y_i w_i\|_q \tag{7}$$

*with $1 \le q \le \infty$ such that $1/p + 1/q = 1$.*

It remains to select $y_j, \ldots, y_m$ appropiately to enable an early recognition of enumeration branches which cannot yield solutions. KAIB, RITTER [5] proposed two selections:

1. $(y_j, y_{j+1}, \ldots, y_m) = (1, 0, \ldots, 0)$: Test if $\|w_j\|_2^2 \leq F\|w_j\|_q$.

2. $(y_j, y_{j+1}, \ldots, y_m) = (\eta, 1-\eta, 0, \ldots, 0)$ with $\eta \in \,]0,1[$ .

   Let's say $w_j = xb_j^* + w_{j+1}$ for an $x \in \mathbf{R}$. Then for every successive $w_j'$ in the same direction, that means every $w_j' = (x+r)b_j^* + w_{j+1}$ with $r \in \mathbf{Z}$ and having the same sign as $x$, we have for $\eta := \frac{x}{x+r}$:

   $$w_j = \eta w_j' + (1-\eta)w_{j+1} \quad \text{and} \quad 0 < \eta < 1. \tag{8}$$

   If $w_j'$ can lead to a solution, then from (7) it follows for every $\eta \in \,]0,1[$:

   $$\eta\|w_j'\|_2^2 + (1-\eta)\|w_{j+1}\|_2^2 \leq F\|\eta w_j' + (1-\eta)w_{j+1}\|_q. \tag{9}$$

   With (8) the inequality reduces to

   $$\|w_j\|_2^2 \leq F\|w_j\|_q.$$

   Here $0 \leq \eta \leq 1$ is needed.

   Therefore we can cut the enumeration in the direction of $x$ if $\|w_j\|_2^2 > F\|w_j\|_q$.

This results in the following algorithm:

**Algorithm 1** *1. Compute a LLL-reduced integer basis of the kernel of the linear system (1): Choose $c_0$ large enough such that the number of remaining columns will be equal to $s - l + 2$ and LLL-reduce the matrix (3).*

*2. Remove the columns with nonzero entries in the first $l$ rows. From the remaining columns remove the first $l$ rows (the zero entries).*

*3. Compute for the remaining columns $b_1, \ldots, b_m$ the Gram-Schmidt vectors $b_1^*, b_2^*, \ldots, b_m^*$ with their Gram-Schmidt coefficients $\mu_{i,j}$, see (4).*

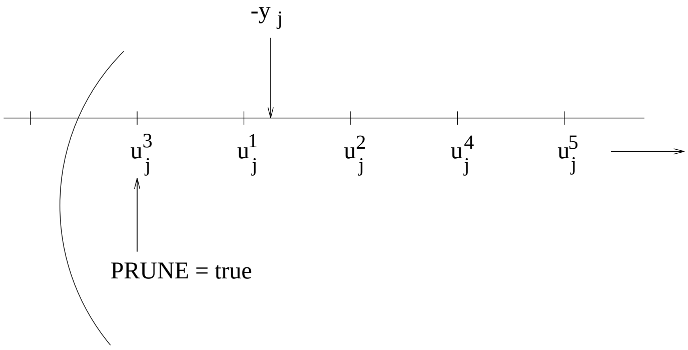
4. *Set $j := 1$;*
   *$F :=$ upper limit to the p-shortest vector in $L$.*
   *Set $\bar{F} := R_p^2 F^2$.*

5. *Do the search loop:*

```
while j ≤ m
        Compute w_j from w_{j+1}.
        if ||w_j||_2^2 > F̄ then
                j := j + 1
                NEXT(u_j)
        else
            if j > 1 then
                if PRUNE(u_j) then
                                if onedirection then
                                        j := j + 1
                                        NEXT(u_j)
                                else
                                        onedirection := true
                                        NEXT(u_j)
                                end if
                else
                                j := j − 1
                                y := Σ_{i=j+1}^{m} u_i μ_{i,j}
                                u_j := round(−y)
                                onedirection := false
                end if
            else /* (j = 1) */
                PRINT u_1, ..., u_m
                NEXT(u_j)
            end if
        end if
end while
```

The procedure NEXT determines the next value of the variable $u_j$. Initially $u_j$ is set to the nearest value of $-y_j := -\sum_{i=j+1}^{m} u_i\mu_{i,j}$, say

$u_j^1$. The next value ($u_j^2$) of $u_j$ is the second nearest integer to $-y_j$ then follows $u_j^3$ and so forth. Therefore the values of $u_j$ alternate around $-y_j$. If PRUNE is true for one value of $u_j$ we do one more jump around $-y_j$, then the enumeration is only proceeded in this remaining direction until it is pruned again.




For arbitrary $p$ with and $q$ such that $1/p + 1/q = 1$ the procedure PRUNE looks like this:

**Algorithm 2** *Choose* $y_j, \ldots, y_m$

*PRUNE*($u_j$)
    **if** $\left|\sum_{i=j}^{m} y_i \|w_i\|_2^2\right| \leq F \cdot \| \sum_{i=j}^{m} y_i w_i \|_q$
        *Return false*
    **else**
        *Return true*
    **end if**

# 4 Results

We used the algorithm to find all 7-(33,8,10) designs with automorphism group $P\Gamma L_2(32)$. The Kramer Mesner matrix was already published in [12]. The algorithm described in [1] produced the following $32 \times 97$ matrix which is a permutation of the rows and columns of the matrix in [12]:

```
222222222222220000000000000000000000000000000000000000000000000000000000000000000000000000000000
210111000000021111211111111111100000000000000000000000000000000000000000000000000000000000000000
111000000100010020000000000010001111121121111111000000000000000000000000000000000000000000000000
001100000000002100100000000000010001100000010001213111111111100000000000000000000000000000000000
000120000000000000000100000100111000000000100100100110001000011211111111100000000000000000000000
000011000000000100000010002000000100100000110011100000010100100000011002111111000000000000000000
000001300011010010000010100100100010002000100010010101001000000100010001001000100000000000000000
000000420000000002000020022000000002000000200000000000200000200000000000002000200000000000000000
000000022000000000200000200000200000200002000000002000000000020000200002202000000000000000000000
000100012100000110010000010100000000010001000101001000010000000100111000001010010111000000000000
000000200120001001101101000011000000110100011001000001000000000010000000010000101001100000000000
000000020002020000000220002000020000002200000002000020000000000000000000002000000000200000000000
000000000101200000000010100002200000000000000001110000000110010110000100100100010200101000000000000
020000000000221002000010100000111100000000010000000000000000100011001010000000100001011100000000
001010000000100100120010000001000100000200000000100101101100010000000000000101000000010211000000000
000011001100001010000102010000000000000010100000010100000000101001000000000010000000010211001110000
001102000000001001000000010010000000100010000000000001010000100000100011000010001010212000001001000
000000022000000000024000200000000000000000000000000000000000000002002200000000220200000000002000000
000001001001100001011001001100111000010000000100000101000100000000100000010110001000000010001100
020000020000020000020000020002000200000000202000000000000020000000000000002000200000000000000020000
000100000101001100000000000010000110001000100001000010000010000110000001000000000200100310101001000
001010000000110001020010010000000001000110000101100100110000000000200000100000000000010000010000300
020000000000000100000100000000000102201000001100000000000100100100110000001000110100210000100000100
001100000000000001000010000000101000110100000010001000000101000000201000000101000000101002000100120
000000020000000000002002000000000000000200000000200000000200000000020020220000000000002000200002000
001000000000000000011000010100000000000001000000100101000020010000011011100110000000011100000002110
000000000001000000100000110100000000000100010200000002001100002000000100100001110001001100010010101
000000000000000000000100101000101100001000001000010101011000000100010100020100110000000000003001100
000000001000000100000000001010001101000100100001110010001000000100100011000100000000011100101100 1
000000000500000000000000000000000000000050000000000005000500000000000000000000000000000000000500000001
000000000000000000000000000000000000000000000000000005000000000500000000500050000000000050000000000000001
000000000000000000000000000000000000000000000000000005000000000000050000000000000000000000500500000500001
```

The method of [1] in it's first version found only 1 solution for $\lambda = 10$. It was Brendan McKay [14] who observed with his general integer backtracking algorithm that there are more than one solutions. In fact he estimated the number of solutions to be between 5 and 7 million. He also estimated that the backtracking algorithm as described in [14] would have taken 100 years to enumerate all these solutions with the computers we had at hand at that time.

For each of the two matrices the above Algorithm 1 found that there are 4 996 426 solutions for $\lambda = 10$ and that there are no solutions for other values of $\lambda$ beside the complementary designs with $\lambda = 16$. All these solutions give nonisomorphic designs: By [10] $\mathrm{P}\Gamma\mathrm{L}(2, 2^5)$ is a maximal subgroup of $S_{33}$. Therefore the full automorphism group could be either $\mathrm{P\Gamma L}(2, 2^5)$ or $S_{33}$. The latter case is impossible, since it would require all 8-subsets to be included into the design because of the transitivity of $S_{33}$ on $X$.

The computing time was about one week on a DEC ALPHA 3000. The newest results of the Bayreuth group on $t$-designs can be found at
`http://www.mathe2.uni-bayreuth.de/betten/DESIGN/d1.html`
and
`http://www.mathe2.uni-bayreuth.de/people/laue.html.`

# References


[1] A. Betten, A. Kerber, A. Kohnert, R. Laue, A. Wassermann: The Discovery of Simple 7-Designs with Automorphism Group $P\Gamma L(2, 32)$. *AAECC 11* in *Lecture Notes in Computer Science* **547** (1995), 281–293.

[2] M. J. Coster, B. A. LaMacchia, A. M. Odlyzko, C. P. Schnorr: An improved low-density subset sum algorithm. *Proceedings EUROCRYPT '91, Brighton, May 1991* in *Springer Lecture Notes in Computer Science* **547** (1991), 54–67.

[3] H. H. Hörner: Verbesserte Gitterbasenreduktion; getestet am Chor-Rivest Kryptosystem und an allgemeinen Rucksack-Problemen. Diplomarbeit, Universität Frankfurt (August 1994).

[4] A. Joux, J. Stern: Improving the Critical Density of the Lagarias-Odlyzko Attack Against Subset Sum Problems. *Proceedings of Fundamentals of Computation Theory 91* in *Lecture Notes in Computer Science* **529** (1991), 258–264.

[5] M. Kaib, H. Ritter: Block Reduction for Arbitrary Norms. Preprint 1995.

[6] R. Kannan: Improved algorithms for integer programming and related lattice problems. *15th Ann. ACM Symb. on Theory of Computing* (1983), 193–206.

[7] E. S. Kramer, D. M. Mesner: $t$-designs on hypergraphs. *Discrete Math.* **15** (1976), 263–296.

[8] D. L. Kreher, S. P. Radziszowski: Finding Simple t-Designs by Using Basis Reduction. *Congressus Numerantium* **55** (1986), 235–244.

[9] J. C. Lagarias, A. M. Odlyzko: Solving low-density subset sum problems. *J. Assoc. Comp. Mach.* **32** (1985), 229–246.

[10] M. W. Liebeck, C. E. Praeger, J. Saxl: The maximal factorizations of the finite simple groups and their automorphism groups, *Memoirs of the Amer. Math. Soc.* **432** (1990), Chapter 9.

[11] A. K. Lenstra, H. W. Lenstra Jr., L. Lovász: Factoring Polynomials with Rational Coefficients, *Math. Ann.* **261** (1982), 515–534.

[12] S. S. Magliveras, D. W. Leavitt: Simple 6-(33,8,36) Designs from $P\Gamma L_2(32)$. *Computational Group Theory* M. D. Atkinson ed., Academic Press (1984) 337–352.

[13] R. Mathon: Computational Methods in Design Theory. *London Math. Soc. Lect. Notes* **166** (1991), 101–117.

[14] B. McKay: Personal communication.
Email-address: `bdm@cs.anu.edu.au`.
WWW-page: `http://cs.anu.edu.au:80/people/bdm/`

[15] S. P. Radziszowski, D. L. Kreher: Solving subset sum problems with the $L^3$ algorithm. *J. Combin. Math. Comput.* **3** (1988), 49–63.

[16] B. Schmalz: The $t$-designs with prescribed automorphism group, new simple 6-designs. *J. Combinatorial Designs* **1** (1993), 125–170.

[17] C. P. Schnorr: A hierachy of polynomial time lattice basis reduction algorithms. *Theoretical Computer Science* **53** (1987), 201–224.

[18] C. P. Schnorr: A More Efficient Algorithm for Lattice Basis Reduction. *J. Algorithms* **9** (1988), 47–62.

[19] C. P. Schnorr, M. Euchner: Lattice basis reduction: Improved practical algorithms and solving subset sum problems. *Proceedings of Fundamentals of Computation Theory 91* in *Lecture Notes in Computer Science* **529** (1991), 68–85.

[20] C. P. Schnorr, H. H. Hörner: Attacking the Chor-Rivest cryptosystem by improved lattice reduction. *Advances in Cryptology – Eurocrypt '51* in *Lecture Notes in Computer Science* **921** (1995), 1–12.

[21] B. Sturmfels, R. Weismantel: Gröbner bases of lattices, corner polyhedra, and integer programming. Preprint (1994)

[22] R. Urbaniak, R. Weismantel, G. M. Ziegler: A Variant of the Buchberger Algorithm for Integer Programming. *SIAM J. Discrete Mathematics*, to appear.

Address of the author:
Alfred Wassermann
Department of Mathematics
University of Bayreuth
D-95440 Bayreuth
Germany
Email: `Alfred.Wassermann@uni-bayreuth.de`
WWW: `http://did.mat.uni-bayreuth.de/wassermann/wassermann.html`