Finding Simple t-Designs with Enumeration Techniques

Alfred Wassermann
Department of Mathematics, University of Bayreuth, D-95440 Bayreuth, Germany

Received May 21, 1996; accepted March 4, 1997

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).

Keywords: *t-designs; multidimensional subset sum problems; constructive combinatorics*

1. INTRODUCTION

Let X be a v-set (i.e., a set with v elements) whose elements are called *points*. A t-(v, k, λ) *design* is a collection of k-subsets (called *blocks*) of X with the property that any t-subset of X is contained in exactly λ blocks. A t-(v, k, λ) 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, λ) designs is to consider the matrix

$$M_{t,k}^{v} := (m_{i,j}), i = 1, \ldots, \left(\begin{array}{c} v \\ t \end{array} \right), j = 1, \ldots, \left(\begin{array}{c} v \\ k \end{array} \right) :$$

The rows of $M_{t,k}^{v}$ 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 ith t-subset is contained in the jth k-subset, otherwise $m_{i,j} := 0$. Simple t-(v, k, λ) designs therefore correspond to $\{0, 1\}$-solutions x of the system of $\binom{v}{t}$ linear equations:

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

e-mail: alfred.wassermann@uni-bayreuth.de; WWW: http://did.mat.uni-bayreuth.de/wassermann/wassermann.html
 CCC 1063 8539/98/020079-12

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 jth orbit of G on the k-subsets of X which contain a representative of the ith orbit of t-subsets of X. This matrix was introduced by Kramer and Mesner [10]. They observed:

Theorem 1 (see [10]). *A simple t-(v, k, λ) design exists with $G \leq \mathrm{Sym}(X)$ as an automorphism group 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 PΓL(2,32) 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).

Solving eq. (1) is a special instance of the multidimensional *subset sum* problem which is known to be NP-complete [5]. This problem is also of interest in other areas such as cryptography [2, 13], number theory [23], and combinatorial optimization [18].

Finding solutions for this problems 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 [17], and lattice basis reduction [1, 11, 12, 19]. See [17] for a survey. Recently also algorithms which use Gröbner bases have been proposed [26, 27].

In [11, 12] the authors used the original lattice basis reduction algorithm (LLL) as described in [14] and a lattice like the one proposed in [13]. Meanwhile lattice basis reduction algorithms have been improved. New algorithms were invented by Schnorr, [21, 22, 24]. Also new lattices have been proposed, see [3, 4, 7].

In [1] the authors implemented these new methods and used a slightly different lattice to find the first simple 7-designs with small parameters.

Meanwhile the lattice basis reduction algorithms are further improved [25] and have been generalized from the euclidean norm to arbitrary norms [8]. These algorithms are now strong enough to find new simple 7-designs.

On the contrary, in order to find *all* $\{0, 1\}$-solutions of (1) until now only exhaustive search techniques based on backtracking have been used. See, for example, [17]. Schmalz [20] used a graph theoretical approach. He enumerates all solutions implicitely via graphs.

The new approach—using lattice basis reduction [14]—is to construct a basis of the kernel of the equation

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

which consists of short integer vectors. But the shortest integer vectors (in the euclidean norm) in the kernel of (2) need not be solutions of our $\{0, 1\}$-problem (1). Kaib and

Ritter proposed in [8] 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 [14]. The explicit enumeration in the second step of [8] is still an exponential algorithm. But in most cases it is much faster than the brute force enumeration which was used in the above mentioned algorithms. Thus in some sense this algorithm combines the two classes of algorithms to solve (1).

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

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

containing $s+2$ column vectors with $l+s+2$ rows. The set of all integer linear combinations of these vectors is called a *lattice*. The maximal number of $(\mathbb{R})$-linear independent vectors in this lattice is called the *rank* of the 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 [14]. See further Section 3.

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 [25] to a new basis.

Definition 1. *Let $L \subset \mathbb{R}^n$ be a lattice of rank m. For $1 \le p < \infty$ the norm defined by the mapping*

$$\|\cdot\|_p : \mathbb{R}^n \to \mathbb{R}, x \mapsto \|x\|_p := \left(\sum_{i=1}^{n} |x_i|^p\right)^{1/p}$$

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

$$\|\cdot\|_\infty : \mathbb{R}^n \to \mathbb{R}, x \mapsto \|x\|_\infty := \max\{|x_i| | 1 \le i \le n\}$$

is called ∞-norm.

(1) *For $1 \le p \le \infty$ we call a vector $\in L$ p-shortest if it is a shortest nonzero vector in L in p-norm.*
(2) *The successive minima $\lambda_1, \ldots, \lambda_m$ of L are defined through: $\lambda_i = \lambda_i(L)$ is the smallest radius r of a ball centered at the origin which contains exactly i linearly*

independent lattice vectors. Thus $\lambda_1(L)$ denotes the (euclidean) length of a 2-shortest vector in L.

If we set in (3) $c_1 = \lambda$ and $c_0 > \lambda$, the ∞-shortest vectors of the lattice with an entry equal to $\pm c_1$ in line $l + s + 1$ are $\{0, 1\}$-solutions of the eq. (1), because

1. Every integer solution $(x_1, \ldots, x_s)^\perp$ of (1) can be regarded as linear combination of the columns of the matrix (3) multiplied by $\pm(x_1, \ldots, x_s, -\lambda, \cdot)$.
2. Every integer linear combination of the columns which does not correspond to a solution of (1) contains at least one nonzero entry in the first l coefficients. Since it is a multiple of c_0 and $c_0 > \lambda$, the ∞-norm of this linear combination is greater than λ.
3. Every integer linear combination which corresponds to a solution of (1) contains only zeros in the first l lines. Only $\{0, 1\}$-linear combinations of the first s columns plus or minus the last column can give a vector which contains only entries with absolute value less or equal to $c_1 = \lambda$ in line $l + 1$ to $l + s$ and $l + s + 2$ and simultaneously only zeros in the lines 1 to l.

A $\{0, 1\}$-solution of the system (1) therefore corresponds to a vector in L which consists of zeros in the first l rows and has 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 $l + s + 2$ it contains $\pm\lambda$ and $\pm c_1$, respectively.

Until now only reduction techniques (see Sec. 3) for the norm $p = 2$ are working efficiently. To find a ∞-shortest vector we have to employ backtracking methods. Since the 2-norm and the ∞-norm are equivalent (for all $x \in \mathbb{R}^n$: $\|x\|_\infty \leq \|x\|_2 \leq \sqrt{n}\|x\|_\infty$) it is reasonable to use 2-short vectors to find the ∞-shortest vectors.

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

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

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

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. IMPROVED LATTICE BASIS REDUCTION

In the first step of the algorithm we compute an integer basis of the kernel of the eq. (2) with lattice basis reduction. The original algorithm (LLL) for lattice basis reduction of

Lenstra, Lenstra, and Lovász [14] reduces a basis $b_1, b_2, \ldots, b_m$ of the lattice $L \subset \mathbb{R}^n$. In this section we only use the euclidean norm:

Algorithm 1 (LLL-algorithm, see [14])**.**
set $k := 1$.
do until $k = m - 1$:

1. *For* $j = 1, \ldots, k-1$ *replace* b_k *by* $b_k - rb_j$, *where* $r = \lceil \mu_{k,j} \rfloor$ *is the nearest integer to* $\mu_{k,j}$.
2. *if* $\delta \|b_k^*\|^2 > \|b_{k+1}^* + \mu_{k+1,k} b_k^*\|^2$ *then interchange* b_{k+1} *and* b_k *and set* $k := \max(k-1, 1)$,
otherwise set $k := k + 1$.

Remark 1. With step 1 of the algorithm we achieve that the LLL-reduced basis vectors are ''as orthogonal as possible''.

In step 2 of the algorithm we compare two vectors: The projection of b_k on the subspace orthogonal to $\langle b_1, b_2, \ldots, b_{k-1}\rangle^\perp$ and the projection of b_{k+1} on the same subspace. Then we take the vector which gives the shorter projection as the new vector b_k.

In [14] the authors give an upper bound on the euclidean length of the reduced basis vectors:

$$\|b_i\|_2^2 \leq \lambda_i^2 \cdot \left(\frac{4}{4\delta - 1}\right)^{m-1} \tag{6}$$

A generalization of this algorithm is *Korkine–Zolotarev reduction* where we not only compare the euclidean length of the projections of b_k and b_{k+1} on the subspace $\langle b_1, b_2, \ldots, b_{k-1}\rangle^\perp$ but we search for the integer linear combination $u_k b_k + u_{k+1} b_{k+1} + \cdots + u_n b_n$ which gives the shortest nontrivial projection on $\langle b_1, b_2, \ldots, b_{k-1}\rangle^\perp$.

Since there is no algorithm to find the shortest nontrivial projection in polynomial time in the number of vectors $(n - k)$, Schnorr restricted the search to blocks of β vectors [22, 24]: The integer linear combination $u_k b_k + u_{k+1} b_{k+1} + \cdots + u_{k+\beta-1} b_{k+\beta-1}$ which gives the shortest nontrivial projection on $\langle b_1, b_2, \ldots, b_{k-1}\rangle^\perp$ is found by explicit enumeration. See further [24] and [25] for improved versions with better estimates for the reduced bases.

We want to compute an integer basis of the kernel of (2). The lattice vectors which correspond to elements of the kernel of (2) contain only zeros in the first z entries. Practical experience shows that it is sufficient to set the constant c_0 equal to $(s + 2)^2$ in order to compute a integer basis of the kernel of (2) with the above lattice basis reduction algorithm.

4. EXPLICIT ENUMERATION

After computing an integer basis of the kernel of the eq. (2) we remove all nonkernel vectors from the lattice and remove the first l lines of the remaining vectors, because they only contain zeros.

Now we search for all integer linear combinations of these remaining kernel basis vectors which correspond to $\{0, 1\}$-solutions of (1), i.e., which only consist of $\pm c_1$ entries. We adapted the algorithm of Kaib and Ritter [8] to find the ∞-shortest vectors in the above lattice. The use of $\langle b_1, b_2, \ldots, b_i\rangle^\perp$ and the lattices L_i is motivated by

1. Korkine–Zolotarev reduction: A basis $b_1, b_2, \ldots, b_m$ is called Korkine–Zolotarev reduced if
$$b_i = \lambda_1(L_i), \qquad \text{for } 1 \leq i \leq m.$$
2. Blockwise Korkine–Zolotarev reduction: Since there is no polynomial algorithm to find a Korkine–Zolotarev reduced basis, Schnorr [24] proposed an algorithm which constructs a basis where only blocks with fixed length of basis vectors are Korkine–Zolotarev reduced. This blocks are reduced by explicit enumeration which has exponential runtime in the block length.
3. It is easy to see that $L_1 = L$. If we can find a shortest vector in L_i we can find a shortest vector in L. We will see that there is an algorithm which computes (in exponential time) the p-shortest vector in L_i. It is a backtracking algorithm which uses information about p-short vectors in L_i to compute p-short vectors in L_{i-1}.

For the first two points see [9, 21, 24]. We will not exploit them any further here. We are interested in the third point, especially for $p = \infty$. For this let us have a closer look on the projection π_i defined in (5):

As above let the *Gram–Schmidt* vectors be $b_1^*, b_2^*, \ldots, b_m^*$ together with their *Gram–Schmidt* coefficients $\mu_{i,j} := \frac{\langle b_i, b_j^* \rangle}{\langle b_j^*, b_j^* \rangle}$ for $i > j$. We set $\mu_{i,i} := 1$. It follows for every basis vector b_t and $j \leq t$:
$$\pi_j(b_t) = \sum_{i=j}^{t} \mu_{t,i} b_i^*.$$
And since $\pi_j(\cdot)$ is a linear mapping it follows for $u_t, u_{t+1}, \ldots, u_m \in \mathbb{Z}$:
$$\pi_j \left(\sum_{t=j}^{m} u_t b_t \right) = \sum_{t=j}^{m} u_t \sum_{i=j}^{t} \mu_{t,i} b_i^*.$$
Reordering of the summands gives
$$\pi_j \left(\sum_{t=j}^{m} u_t b_t \right) = \sum_{s=j}^{m} b_s^* \sum_{i=s}^{m} u_i \mu_{i,s}.$$
With $c_s := \|b_s^*\|_2^2$ for $1 \leq s \leq m$ it follows
$$\left\| \pi_j \left(\sum_{t=j}^{m} u_t b_t \right) \right\|_2^2 = \sum_{s=j}^{m} \left(\sum_{i=s}^{m} u_i \mu_{i,s} \right)^2 c_s.$$
For the speed of the enumeration algorithm the following relation between the projections $\pi_j(\sum_{t=j}^{m} u_t b_t)$ and $\pi_{j+1}(\sum_{t=j+1}^{m} u_t b_t)$ is crucial:
$$\pi_j \left(\sum_{t=j}^{m} u_t b_t \right) = \left(\sum_{i=j}^{m} u_i \mu_{i,j} \right) b_j^* + \pi_{j+1} \left(\sum_{t=j+1}^{m} u_t b_t \right), \tag{7}$$
hence
$$\left\| \pi_j \left(\sum_{t=j}^{m} u_t b_t \right) \right\|_2^2 = \left(\sum_{i=j}^{m} u_i \mu_{i,j} \right)^2 c_j + \left\| \pi_{j+1} \left(\sum_{t=j+1}^{m} u_t b_t \right) \right\|_2^2. \tag{8}$$

It means we have to compute $(\sum_{i=j}^{m} u_i\mu_{i,j})^2 c_j$ to get from stage $j+1$ to stage j of the algorithm.

Definition 3. *For $u_j, u_{j+1}, \ldots, u_m \in \mathbb{Z}$ 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 $m \geq t \geq 1$ and finally $w_1 = \sum_{i=1}^{m} u_i b_i$.

Remark 2. If $u_{j+1}, u_{j+2}, \ldots, u_m \in \mathbb{Z}$ are fixed and $u_j \in \mathbb{Z}$ 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

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

We are searching for the p-shortest vector with $1 \leq p \leq \infty$. The solutions of our system of linear equations (2) are the ∞-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 $1 \leq p \leq \infty$.

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

Moreover, Kaib and Ritter [8] use Hölders inequality to combine the search for p-shortest vectors with enumeration in 2-norm:

Theorem 2. *If for fixed $u_j, u_{j+1}, \ldots, u_m \in \mathbb{Z}$ there exist $u_1, u_2, \ldots, u_{j-1} \in \mathbb{Z}$ with $\|\sum_{i=1}^{m} u_i b_i\|_p \leq F$, then for all $y_j, y_{j+1}, \ldots, y_m \in \mathbb{R}$:*

$$\left|\sum_{i=j}^{m} y_i\|w_i\|_2^2\right| \leq F \cdot \left\|\sum_{i=j}^{m} y_i w_i\right\|_q \tag{9}$$

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

Proof. For $l < i$ we see from (7) that there exist $x_l, x_{l+1}, \ldots, x_{i-1} \in \mathbb{R}$ such that

$$w_l = x_l b_l^* + w_{l+1} = x_l b_l^* + x_{l+1} b_{l+1}^* + w_{l+2} = \cdots = x_l b_l^* + \cdots + x_{i-1} b_{i-1}^* + w_i.$$

It follows

$$\langle w_l - w_i, w_i\rangle = \langle x_l b_l^* + \cdots + x_{i-1} b_{i-1}^*, w_i\rangle = 0,$$

since w_i is a linear combination of $b_i^*, \ldots, b_m^*$. Therefore, if $l < i$,

$$\langle w_l, w_i\rangle = \langle w_i, w_i\rangle.$$

If there exist $u_1, u_2, \ldots, u_m \in \mathbb{Z}$ with $\|w_1\|_p = \|\sum_{i=1}^{m} u_i b_i\|_p \leq F$ then Hölders inequality gives for an arbitrary vector $v \in \mathbb{R}^n$:

$$|\langle w_1, v\rangle| \leq \|w_1\|_p\|v\|_q \leq F \cdot \|v\|_q.$$

And it follows:

$$\left|\sum_{i=j}^{m} y_i\|w_i\|_2^2\right| = \left|\sum_{i=j}^{m} y_i\langle w_i, w_i\rangle\right| = \left|\sum_{i=j}^{m} y_i\langle w_1, w_i\rangle\right| = \left|\left\langle w_1, \sum_{i=j}^{m} y_i w_i\right\rangle\right|$$

$$\leq \|w_1\|_p \left\| \sum_{i=j}^{m} y_i w_i \right\|_q \leq F \cdot \left\| \sum_{i=j}^{m} y_i w_i \right\|_q .$$

□

It remains to select $y_j, \ldots, y_m$ appropriately to enable an early recognition of enumeration branches which cannot yield solutions. Kaib and Ritter [8] proposed the following selection:

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

$$w_j = \eta w_j' + (1-\eta)w_{j+1} \quad \text{and} \quad 0 < \eta < 1. \tag{10}$$

If w_j' can lead to a solution, then from (9) it follows for every $\eta \in]0, 1[$:

$$\eta\|w_j'\|_2^2 + (1-\eta)\|w_{j+1}\|_2^2 \leq F\|\eta w_j' + (1-\eta)w_{j+1}\|_q. \tag{11}$$

With (10) 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 2.

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* $j := 1$; *Set* $F :=$ *upper bound of the* p*-shortest vector in* L. *Set* $\bar{F} := R_p^2 F^2$.
5. *Do the search loop*:

while $j \leq m$
  Compute w_j from w_{j+1}.
  if $\|w_j\|_2^2 > \bar{F}$ **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, see Figure 1. If $p = 2$ the procedure PRUNE(u_j) is equivalent to the first test $\|w_j\|_2^2 > \bar{F}$ and hence the (expensive) computation of the w_j's can be avoided. Actually for $p = 2$ Kannan [9] already proposed this method. And he noticed that in each level we have to test only two values of u_j: u_j^1 and u_j^2. For arbitrary p with and q such that $1/p + 1/q = 1$ the procedure PRUNE looks like this:

Algorithm 3. *Choose* $y_j, \ldots, y_m$

PRUNE(u_j)

if $|\sum_{i=j}^{m} y_i\|w_i\|_2^2| \le F \cdot \|\sum_{i=j}^{m} y_i w_i\|_q$

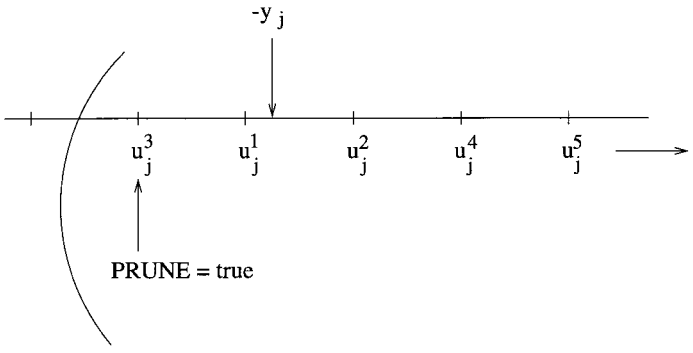

FIG. 1. Enumeration With Pruning.

Return false
else
Return true
end if

Some improvements can be made in order to speed up the algorithm:

1. We do not have to compute w_j in order to test if $\|w_j\|_2^2 > R_pF = \bar{F}$. Since
$$\|w_j\|_2^2 = (u_j + y_j)^2 \cdot \|b_j^*\|_2^2 + \|w_{j+1}\|_2^2$$
we can keep track of $\|w_j\|_2^2$ and test if $\|w_j\|_2^2 > R_pF = \bar{F}$. This test is cheap and we only compute w_j if it succeeds.
2. We can stop the enumeration of a branch if there is a row i where $|w_{j,i}| \neq \lambda$ or if $|w_{j,n}| \neq 1$ and $b_{1,i} = b_{2,i} = \cdots = b_{j-1,i} = 0$. In this case $w_{1,i}$ will stay unchanged for all selections of $u_1, \ldots, u_{j-1}$.
3. Since multiplication of the solution vector with -1 yields the same solution of (1) we always choose the last nonzero coefficient $u_t > 0$.
4. Another choice of $(y_j, y_{j+1}, \ldots, y_m)$ in (9) and Algorithm 3 is $y_j := 1$ and $y_{j+1} := -1$: We know from (7) and (8) that,
$$w_j = \left(\sum_{i=j}^{m} u_i \mu_{i,j} \right) b_j^* + w_{j+1},$$
respectively,
$$\|w_j\|_2^2 = \left(\sum_{i=j}^{m} u_i \mu_{i,j} \right)^2 \|b_j^*\|_2^2 + \|w_{j+1}\|_2^2.$$
Using $y_j = 1$ and $y_{j+1} = -1$ the inequality (9) results in
$$\left| \sum_{i=j}^{m} u_i \mu_{i,j} \right| \leq F \cdot \frac{\|b_j^*\|_1}{\|b_j^*\|_2^2}.$$
If we precompute $F \cdot \frac{\|b_j^*\|_1}{\|b_j^*\|_2^2}$ this test is also very cheap.

5. RESULTS

We used the algorithm to find all 7-(33,8,10) designs with automorphism group PΓL(2,32) acting on the projective line GF(32) $\cup\{\infty\}$. With the lexicographic numbering

$$\begin{array}{llll} 1: & \infty & 5: & x+1 \in \mathbb{F}_{32} \\ 2: & 0 \in \mathbb{F}_{32} & 6: & x^2 \in \mathbb{F}_{32} \\ 3: & 1 \in \mathbb{F}_{32} & \vdots & \vdots \\ 4: & x \in \mathbb{F}_{32} & 33: & x^4 + x^3 + x^2 + x + 1 \in \mathbb{F}_{32} \end{array}$$

the following set of strong generators $\alpha_1, \alpha_2, \alpha_3, \alpha_4$ for a Sims-chain of PΓL(2,32) was used:

$$\begin{aligned}
\alpha_1 &= (4,29,15,18,6)(5,28,14,19,7)(8,27,24,31,22)\\
&\quad (9,26,25,30,23)(10,21,32,16,12)(11,20,33,17,13)\\
\alpha_2 &= (3,33,9,19,6)(4,17,30,25,12)(5,18,29,8,16)\\
&\quad (7,32,26,24,23)(10,15,21,13,31)(11,20,22,28,27)\\
\alpha_3 &= (2,33,26,11,15,14,21,22,7,3)(4,18,6,32,9,16,30,10,20,13)\\
&\quad (5,17)(8,19,29,27,24,12,31,25,23,28)\\
\alpha_4 &= (1,33,7,15,16,14,5,13,24,3)(4,10,32,27,29,28,26,31,17,19)\\
&\quad (6,23,9,30,22,8,21,18,11,12)(20,25).
\end{aligned}$$

The algorithm described in [1] produced a 32×97 matrix which is a permutation of the rows and columns of the matrix in [16], generated by another set of permutations.

The method of [1] in its first version found only 1 solution for $\lambda = 10$. It was Brendan McKay 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 his backtracking algorithm 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 2 found that there are 4 996 426 solutions for $\lambda = 10$ and that there are no solutions for other values of λ beside the complementary designs with $\lambda = 16$. All these solutions give nonisomorphic designs: By [15] PΓL(2,32) is a maximal subgroup of S_{33}. Therefore the full automorphism group could be either PΓL(2,32) 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.

ACKNOWLEDGMENTS

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. Finally, the author wants to thank Anton Betten and Reinhard Laue. It is a pleasure to work with them.

REFERENCES

[1] A. Betten, A. Kerber, A. Kohnert, R. Laue, and A. Wassermann, *The discovery of simple* 7*-designs with automorphism group PΓL*(2,32). AAECC 11 in Lecture Notes in Computer Science **948** (1995), 131–145.

[2] E. F. Brickell, ''*Solving low density knapsacks*,'' Advances in cryptology, Proceedings of Crypto '83, Plenum Press, New York, 1984, 25–37.

[3] M. J. Coster, B. A. LaMacchia, A. M. Odlyzko, and 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.

[4] M. J. Coster, A. Joux, B. A. LaMacchia, A. M. Odlyzko, C. P. Schnorr, and J. Stern, *Improved low-density subset sum algorithms*, Computational Complexity **2** (1992), 111–128.

[5] M. R. Garey and D. S. Johnson, Computers and intractability: A guide to the theory of NP-completeness, W. H. Freeman and Company, 1979.

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

[7] A. Joux and 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.

[8] M. Kaib and H. Ritter, *Block reduction for arbitrary norms*, preprint 1995.

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

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

[11] D. L. Kreher and S. P. Radziszowski, *Finding simple t-designs by using basis reduction*, Congressus Numerantium **55** (1986), 235–244.

[12] D. L. Kreher and S. P. Radziszowski, *Constructing* 6-(14,7,4) *designs*, Contemporary Mathematical, American Mathematical Society, **111** (1990), 137–151.

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

[14] A. K. Lenstra, H. W. Lenstra Jr., and L. Lovász, *Factoring polynomials with rational coefficients*, Math. Ann. **261** (1982), 515–534.

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

[16] S. S. Magliveras and D. W. Leavitt, ''*Simple 6-(33,8,36) designs from $P\Gamma L_2(32)$*,'' Computational group theory, M. D. Atkinson (Editor), Academic Press, 1984, pp. 337–352.

[17] R. Mathon, *Computational methods in design theory*, London Math. Soc. Lect. Notes **166** (1991), 101–117.

[18] G. L. Nemhauser and L. A. Wolsey, Integer and combinatorial optimization, Wiley, 1988.

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

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

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

[22] C. P. Schnorr, *A more efficient algorithm for lattice basis reduction*, J. Algorithms **9** (1988), 47–62.

[23] C. P. Schnorr, *Factoring integers and computing discrete logarithms via diophantine approximation*, Advances in Cryptology—Eurocrypt '91 in Lecture Notes in Computer Science **547** (1991), 281–293.

[24] C. P. Schnorr and 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.

[25] C. P. Schnorr and H. H. Hörner, *Attacking the Chor–Rivest cryptosystem by improved lattice reduction*, Advances in Cryptology—Eurocrypt '95 in Lecture Notes in Computer Science **921** (1995), 1–12.

[26] B. Sturmfels and R. Weismantel, *Gröbner bases of lattices, corner polyhedra, and integer programming*, Preprint, 1994.

[27] R. Urbaniak, R. Weismantel, and G. M. Ziegler, *A variant of the Buchberger algorithm for integer programming*, SIAM J. Discrete Mathematics (to appear).