AUSTRALASIAN JOURNAL OF COMBINATORICS
Volume **40** (2008), Pages 173–186

Primitive sets in a lattice

Spyros S. Magliveras

Department of Mathematical Sciences
Florida Atlantic University
Boca Raton, FL 33431
U.S.A.
spyros@fau.unl.edu

Tran van Trung

Institute for Experimental Mathematics
University of Duisburg-Essen
Essen
Germany
trung@iem.uni-due.de

Wandi Wei

Department of Mathematical Sciences
Florida Atlantic University
Boca Raton, FL 33431
U.S.A
wei@brain.math.fau.edu

Abstract

We study the problem of extending a primitive set of size $k < n$, in a lattice of rank n, to a basis of the lattice. Our approach relies on solving the *matrix completion problem*, and we show that the relevant computed matrices have nicely bounded entries. We present formulas, for the cases $k = 1,\ n - 1$ for an extended basis, and analyze the computational complexity for each of these two cases. The formula for $k = 1$ was already known [6], while the formula for $k = n - 1$ and the complexity analysis for the two cases are new. For the general case, $k < n$, to the authors' knowledge, there are no known formulas for computing an extended basis. However, a polynomial time algorithm to compute an extended basis can be constructed by combining some well known, efficient algorithms.

1 Introduction

Let $\mathbb{Z}$ denote the ring of integers, and $\mathbb{R}^m$ the m-dimensional vector space over the reals. If $\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n \in \mathbb{R}^m$ are n linearly independent vectors (over $\mathbb{R}$), the set

$$\Lambda = \Lambda(\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n) = \{z_1\mathbf{b}_1 + z_2\mathbf{b}_2 + \ldots + z_n\mathbf{b}_n : z_i \in \mathbb{Z}, 1 \leq i \leq n\}$$

is called a **lattice of rank** n **in** $\mathbb{R}^m$, and the set of vectors $\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n$ a **basis** of Λ.

A set C of k lattice vectors $\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_k \in \Lambda$ is said to be **primitive** if these vectors are linearly independent over $\mathbb{R}$ and

$$\Lambda \cap \mathrm{span}_{\mathbb{R}}(\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_k) = \Lambda(\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_k).$$

When $k = 1$, the single vector in C is called a **primitive vector**.

It has been proved (see, for example, [4, 7]) that a lattice vector

$$\mathbf{a} = a_1\mathbf{b}_1 + a_2\mathbf{b}_2 + \ldots + a_n\mathbf{b}_n$$

is primitive if and only if

$$\gcd(a_1, a_2, \ldots, a_n) = 1.$$

It has also been shown that any primitive set of vectors can be extended to a basis for the lattice. The proof in [4] depends on the result that if $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_k$ $(k < n)$ form a primitive set and

$$\mathbf{y} \in \Lambda \setminus \Lambda(\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_k),$$

then the $(k+1)$-dimensional parallelotope P spanned by $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_k, \mathbf{y}$ contains a vector $\mathbf{y}_{k+1} \in P \cap \Lambda$ with a positive minimum distance to $\Lambda(\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_k)$, and the $k+1$ vectors $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_k, \mathbf{y}_{k+1}$ form a primitive set. The proof in [7] makes use of the fact that for any k linearly independent lattice vectors $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_k$, the infimum

$$\inf\{t_k > 0 \ : \ \mathbf{z} = t_1\mathbf{y}_1 + t_2\mathbf{y}_2 + \ldots + t_k\mathbf{y}_k \in \Lambda, \ t_i \in \mathbb{R}, \ t_i \geq 0, \ 1 \leq i \leq k\}$$

is actually attained for some vector $\mathbf{z} \in \Lambda$, and $\mathbf{z}$ is used in the construction of an extended basis. These authors do not give formulas or efficient algorithms for computing their extended bases.

A different way of dealing with this problem is through the **matrix completion** problem: Let A be a $k \times n$ integer matrix, with $k < n$, such that the gcd of the k^{th} order minors of A is $g_k(A)$. Find an $(n-k) \times n$ integer matrix B such that the absolute value of the determinant of the compound matrix $\binom{A}{B}$ is the same as $g_k(A)$. The matrix completion problem is a bit more general than the extended basis problem, since when $g_k(A) = 1$, the former becomes the latter.

In this article we study the problem of extending a primitive set of vectors in a lattice to a basis from the point of view of the matrix completion problem, and present our results in three cases. The first case is when $k = 1$. In this case,

we analyze the computational complexity of a previously known formula [6]. For $k = n - 1$ we present a new formula and analyze its computational complexity. Finally, in the general case, we utilize certain known algorithms for other problems to design polynomial time algorithms for the matrix completion problem.

2 The case $k = 1$

Let $\mathbb{Z}^{k\times n}$ denote the set of $k \times n$ matrices over $\mathbb{Z}$. When $k = 1$, the following result on matrices over a principal ideal domain (PID), presented by M. Newman [6], can be used as a formula for the efficient computation of an extended basis from a given primitive lattice vector.

Theorem 2.1 (Theorem II.1 in [6]) *Let R be a principal ideal domain, α_1, α_2, $\ldots, \alpha_n$ be elements of R, and let d_n be their greatest common divisor. Then there is a matrix U over R with first row $(\alpha_1, \alpha_2, \ldots, \alpha_n)$ and determinant d_n.*

Let $\mathbf{a} := (a_1, a_2, \ldots, a_n) \in \mathbb{Z}^n$ be a nonzero vector, and

$$d := \gcd(\mathbf{a}) := \gcd(a_1, a_2, \ldots, a_n).$$

By permuting the basis vectors, if necessary, we can achieve that $a_1 \neq 0$. Let

$$\begin{aligned} d_1 &= a_1, \\ d_i &= \gcd(a_1, a_2, \ldots, a_i), \quad 2 \le i \le n, \\ d &= d_n. \end{aligned}$$

Since $a_1 \neq 0$, all the d_i are well defined, and

$$d_i = \gcd(d_{i-1}, a_i), \quad 2 \le i \le n.$$

By the Euclidean algorithm, we can determine values $t_i,\ s_i,\ (2 \le i \le n)$ satisfying Bézout's identity :

$$d_i = t_{i-1}\, d_{i-1} + s_{i-1}\, a_i, \quad \text{with} \quad |s_{i-1}| \le d_{i-1}, \quad 2 \le i \le n. \tag{2.1}$$

The corollary below follows immediately from Theorem 2.1 and its proof.

Corollary 2.1 *Let n be a positive integer greater than 1, $(a_1, a_2, \ldots, a_n) \in \mathbb{Z}^n$ a nonzero vector with $a_1 \neq 0$, and $d = \gcd(a_1, a_2, \ldots, a_n)$. Let $U = U_n$ be the $n \times n$ matrix*

$$U := U_n := (u_{i,j}) = \begin{pmatrix} a_1 & a_2 & a_3 & \dots & a_n \\ -s_1 & t_1 & 0 & \dots & 0 \\ -\frac{a_1\, s_2}{d_2} & -\frac{a_2\, s_2}{d_2} & t_2 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ -\frac{a_1\, s_{n-1}}{d_{n-1}} & -\frac{a_2\, s_{n-1}}{d_{n-1}} & -\frac{a_3\, s_{n-1}}{d_{n-1}} & \dots & t_{n-1} \end{pmatrix},$$

i.e.,

$$\begin{aligned} u_{1,j} &= a_j, \quad 1 \le j \le n \\ u_{i,j} &= -\frac{a_j\, s_{i-1}}{d_{i-1}}, \quad 1 \le j \le i-1,\ 2 \le i \le n \\ u_{i,i} &= t_i, \quad 2 \le i \le n \\ u_{i,j} &= 0, \quad i+1 \le j \le n,\ 2 \le i \le n \end{aligned}$$

Then U_n is an integral matrix and

$$det\,(U_n) = d_n = d. \tag{2.2}$$

Corollary 2.1 can be viewed as a formula for the computation of the extended basis when the gcd $d = 1$. The complexity of the computation will be analyzed latter.

By the way, it might be interesting to note that this corollary can be regarded as a variant of Bézout's identity : For integers a, b not both zero, there exist integers t, s such that

$$\gcd(a, b) = at + bs. \tag{2.3}$$

Because (2.3) can be rewritten as

$$\gcd(a, b) = \begin{vmatrix} a & b \\ -s & t \end{vmatrix}, \quad t, s \in \mathbb{Z}. \tag{2.4}$$

Based on the Corollary 2.1 stated above, one can easily compute matrix U as shown in the following algorithm.

Algorithm A.

Input: $A = (a_1, a_2, \dots, a_n) \in \mathbb{Z}^n$ with $a_1 \neq 0$.

Output: $U \in \mathbb{Z}^{n \times n}$ such that the first row of U is A and $\det(U) = \gcd(A)$.

Steps :

Step 1. Invoke the Euclidean Algorithm to compute the d_i $(2 \le i \le n)$ and the values of $s_i,\ t_i$ $(2 \le i \le n)$.

Step 2. Compute the integral values of the entries $\frac{a_i\, s_j}{d_j}$, $2 \le i \le j-1$, $2 \le j \le n$.

To analyze the time complexity of Algorithm A, we need some lemmas.

Lemma 2.1 *Let $u, v \in \mathbb{Z} \setminus \{0\}$. Then, there exist $s, t \in \mathbb{Z}$ such that $gcd(u, v) = su + tv$ and*

$$|s| \le |v|,\ |t| \le |u|.$$

Proof. Let d denote $\gcd(u, v)$, and $s_0,\ t_0 \in \mathbb{Z}$ any numbers such that $d = s_0 u + t_0 v$. Then for any $k \in \mathbb{Z}$,

$$s = s_0 + kv,\ t = t_0 - ku$$

satisfy $su + tv = d$. We have

$$\begin{aligned} d &\ge |s_0 + kv| \cdot |u| - |t_0 - ku| \cdot |v|, \\ d + |t_0 - ku| \cdot |v| &\ge |s_0 + kv| \cdot |u|. \end{aligned}$$

By the division algorithm, we can choose k such that $|t_0 - ku| < |u|$, i.e., $|t_0 - ku| \le |u| - 1$. So

$$d + (|u| - 1) \cdot |v| \ge d + |t_0 - ku| \cdot |v| \ge |s_0 + kv| \cdot |u|,$$

and then

$$|s_0 + kv| \le \frac{d}{|u|} + (1 - \frac{1}{|u|})\,|v| \ = \ |v| - \frac{|v| - d}{|u|} \ \le \ |v|. \qquad \square$$

We thank the anonymous referee for noting that the entries of matrix U, as computed by Algorithm A, are nicely bounded. More precisely, from $u_{i,j} = -a_j s_{i-1}/d_{i-1}$ and $|s_{i-1}| \le d_{i-1}$, we have that

$$|u_{i,j}| \le a_j\ .$$

The following lemmas are well known, and can be found in many books, for example, [1], [3], [5].

Lemma 2.2 *Let $u, v \in \mathbb{Z} \setminus \{0\}$. Then, $gcd(u, v)$ as well as s, t such that $gcd(u, v) = su + tv$ can be computed in $O((\log |u|)(\log |v|))$ bit operations.*

Lemma 2.3 *Let $u, v \in \mathbb{Z} \setminus \{0\}$. Then, each of $u \cdot v$ and $\frac{u}{v}$ can be computed in $O((\log |u|)(\log |v|))$ bit operations.*

We now analyze the time complexity of Algorithm A. Let a_0 and a_0' be the two largest among all the absolute values $|a_i|$. The case where either a_0 or a_0' is 0 is trivial, so we now assume that both are nonzero.

Theorem 2.2 *The worst-case time complexity of Algorithm A is* $O(n^2(\log a_0)(\log a_0'))$ *bit operations.*

Proof. Step 1 of the algorithm can be carried out by invoking the Euclidean Algorithm $n-1$ times. By Lemma 2.2, this can be done in

$$(n-1)\cdot O((\log a_0)(\log a_0')) = O(n(\log a_0)(\log a_0'))$$

bit operations.

The number of divisions and the number of multiplications in Step 2 is the same, that is

$$2+3+\ldots+(n-1) = O(n^2).$$

By Lemma 2.1, the absolute values of all entries of U_n are bounded by a_0 and a_0'. Therefore, by Lemmas 2.2 and 2.3, all integral values of the fractions involved in Step 2 of the algorithm can be computed in $O(n^2(\log a_0)(\log a_0'))$ bit operations. Therefore, the worst-case time complexity of the algorithm is $O(n^2(\log a_0)(\log a_0'))$ bit operations. □

Let us now apply the above results to primitive sets.

Let $\Lambda = \Lambda(\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n)$ be a lattice with basis $\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n$. Let $\mathbf{a} = a_1\mathbf{b}_1 + a_2\mathbf{b}_2 + \ldots + a_n\mathbf{b}_n \in \Lambda$ be a primitive vector. Then we have

$$\gcd(a_1, a_2, \ldots, a_n) = 1.$$

Corollary 2.1 asserts that $\det(U) = 1$, and the row vectors of matrix $U = (u_{i,j})$ with respect to basis $\{\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n\}$ give rise to a new basis of Λ, which is an extension of primitive vector $\mathbf{a}$.

Combining this and Corollary 2.1, we have

Theorem 2.3 *The primitive vector* $\mathbf{a}$ *together with the* $n-1$ *vectors*

$$\sum_{j=0}^{n} u_{i,j}\mathbf{b}_j, \quad 2 \le i \le n$$

form an extended basis of $\mathbf{a}$*, where*

$$u_{i,j} = -\frac{a_j\, s_{i-1}}{d_{i-1}}, \quad 1 \le j \le i-1,\ 2 \le i \le n$$

$$\begin{aligned} u_{i,i} &= t_i, \quad 2 \le i \le n \\ u_{i,j} &= 0, \quad i+1 \le j \le n,\ 2 \le i \le n \end{aligned}$$

The worst-case time complexity (in bit operations) of computing the extended basis is $O(n^2(\log a_0)(\log a_0'))$.

3 The case $k = n - 1$

We continue using the notation for $a_i, d_i, s_i, t_i, \Lambda, \mathbf{b}_i$ as introduced in §2.

We need the following results. The first of them can be proved by induction, and the second can be found in [2].

Lemma 3.1 *Let a_i $(1 \le i \le n)$ be integers, not all zero. Then the* $\gcd(a_1, a_2, \ldots, a_n)$ *can be expressed as an integral linear combination of the a_i as follows:*

$$\gcd(a_1, a_2, \ldots, a_n) = \sum_{i=1}^{n} (s_{i-1} \prod_{j=i}^{n-1} t_j) a_i, \tag{3.1}$$

where $s_0 = 1$.

Lemma 3.2 *Let $1 \le k < n$ and*

$$\mathbf{a}_i = a_{i,1}\mathbf{b}_1 + a_{i,2}\mathbf{b}_2 + \ldots + a_{i,n}\mathbf{b}_n \in \Lambda, \quad 1 \le i \le k.$$

Let M denote the $k \times n$ matrix $(a_{i,j})$ $(1 \le i \le k,\ 1 \le j \le n)$. Then $\{\mathbf{a}_i : 1 \le i \le k\}$ is a primitive set if and only if the gcd of all the k^{th} order minors of M is 1.

As a counterpart of Lemma 2.1 for n integers, not all zero, we have

Theorem 3.1 *Let a_i $(1 \le i \le n)$ be integers, not all zero. Suppose that $|a_n|$ is the smallest among all $|a_i| \ne 0$. Then the* $\gcd(a_1, a_2, \ldots, a_n)$ *can be expressed as an integral linear combination of the a_i*

$$\gcd(a_1, a_2, \ldots, a_n) = \sum_{i=1}^{n} c_i a_i, \tag{3.2}$$

with

$$|c_i| < |a_n|, \quad 1 \le i \le n-1 \tag{3.3}$$

$$|c_n| < |a_1| + \ldots + |a_{n-1}|. \tag{3.4}$$

Proof. Let d denote the $\gcd(a_1, a_2, \ldots, a_n)$. By Lemma 3.1, there are integers e_i such that

$$d = \sum_{i=1}^{n} e_i a_i. \tag{3.5}$$

Writing

$$e_i = r_i + q_i a_n,\ 0 \le r_i < |a_n|, \quad 1 \le i \le n-1, \tag{3.6}$$

we can express (3.5) as

$$\begin{aligned} d &= \sum_{i=1}^{n} e_i a_i \\ &= \sum_{i=1}^{n-1} (r_i + q_i a_n) a_i + e_n a_n \\ &= \sum_{i=1}^{n-1} r_i a_i + (e_n + \sum_{i=1}^{n-1} q_i a_i) a_n. \end{aligned} \tag{3.7}$$

Let $c_i = r_i$ $(1 \le i \le n-1)$ and $c_n = e_n + \sum_{i=1}^{n-1} q_i a_i$. Then (3.3) is satisfied according to (3.6). We now prove (3.4).

By (3.5) and (3.7) we have

$$\begin{aligned} d &\ge |(e_n + \sum_{i=1}^{n-1} q_i a_i) a_n| - |\sum_{i=1}^{n-1} r_i a_i| \\ &\ge |(e_n + \sum_{i=1}^{n-1} q_i a_i) a_n| - \sum_{i=1}^{n-1} |r_i a_i| \\ &> |(e_n + \sum_{i=1}^{n-1} q_i a_i) a_n| - |a_n| \sum_{i=1}^{n-1} |a_i|, \end{aligned}$$

which yields

$$d + |a_n| \sum_{i=1}^{n-1} |a_i| > |(e_n + \sum_{i=1}^{n-1} q_i a_i) a_n|.$$

Therefore,

$$\begin{aligned} & \frac{d}{|a_n|} + \sum_{i=1}^{n-1} |a_i| > |e_n + \sum_{i=1}^{n-1} q_i a_i|, \\ i.e., \quad & \sum_{i=1}^{n-1} |a_i| \ge |(e_n + \sum_{i=1}^{n-1} q_i a_i)| = |c_n|. \end{aligned}$$

□

The following notation will be used. Let M be a $k \times n$ matrix, $\{i_1, \ldots, i_p\} \subseteq [1, k]$ and $\{j_1, \ldots, j_q\} \subseteq [1, n]$. We use $M[i_1, \ldots, i_p \,|\, j_1, \ldots, j_q]$ to denote the submatrix of M formed by the rows $i_1, \ldots, i_p$ and columns $j_1, \ldots, j_q$. We also use $M[\text{-} \,|\, j_1, \ldots, j_q]$ and $M[i_1, \ldots, i_p \,|\, \text{-}]$ to denote $M[1, 2, \ldots, k \,|\, j_1, \ldots, j_q]$ and $M[i_1, \ldots, i_p \,|\, 1, 2, \ldots, n]$, respectively.

Based on Lemmas 3.1 and 3.2 and Theorem 3.1 we develop the following algorithm.

Algorithm B

Input: $A = (a_{i,j}) \in \mathbb{Z}^{(n-1)\times n}$ with $\det(A[1,2,\ldots,n-1 \mid 1,2,\ldots,n-1]) \neq 0$.

Output: $B \in \mathbb{Z}^{n\times n}$ such that A is a submatrix of B and $\det(A)$ is the gcd of the $(n-1)^{th}$-order minors of A.

Steps:

Step 1. Compute

$$A_i := \det(A[1,\ldots,n-1 \mid 1,\ldots,i-1,i+1,\ldots,n]),\ 1 \le i \le n.$$

By permuting A_i $(1 \le i \le n)$ if necessary, we may assume without loss of generality that $A_1 \neq 0$ and $|A_n|$ is the smallest among all positive $|A_j|$ $(1 \le i \le n)$.

Steps 2. Employ the Euclidean algorithm to compute

$$\begin{aligned} d_1' &:= A_1, \\ d_i' &:= \gcd(d_{i-1}, A_i), \quad 2 \le i \le n, \\ d' &:= d_n' \end{aligned}$$

and $t_i',\ s_i',\ (2 \le i \le n)$ such that

$$d_i' = t_{i-1}'\, d_{i-1}' + s_{i-1}'\, A_i, \quad 2 \le i \le n.$$

Step 3. Compute

$$b_i = (-1)^{n-i} s_{i-1}' \prod_{j=i}^{n-1} t_j' \mod A_n, \quad 1 \le i \le n-1 \tag{3.8}$$

and

$$b_n = \frac{d' - \sum_{i=1}^{n-1} (-1)^{n+i} b_i A_i}{A_n}. \tag{3.9}$$

Step 4. Output the compound matrix

$$B = \begin{pmatrix} A \\ (b_1, b_2, \ldots, b_n) \end{pmatrix},$$

and stop.

Theorem 3.2 *Algorithm B is correct, and the integers b_i are bounded as follows:*

$$|b_i| < |A_n|,\ 1 \le i \le n-1 \tag{3.10}$$

$$|b_n| < |A_1| + \ldots + |A_{n-1}| \tag{3.11}$$

Proof. Applying Lemma 3.1 to $A_1, \ldots, A_n$, we have

$$d' = \sum_{i=1}^{n} (s'_{i-1} \prod_{j=i}^{n-1} t'_j) A_i = \sum_{i=1}^{n} ((-1)^{n-i} s'_{i-1} \prod_{j=i}^{n-1} t'_j)((-1)^{n+i} A_i).$$

By (3.8), we can write

$$b_i = (-1)^{n-i} s'_{i-1} \prod_{j=i}^{n-1} t'_j - q_i A_n, \quad 1 \le i \le n-1,$$

where q_i are some integers. So

$$\begin{aligned} d' &= \sum_{i=1}^{n} ((-1)^{n-i} s'_{i-1} \prod_{j=i}^{n-1} t'_j)((-1)^{n+i} A_i) \\ &= \sum_{i=1}^{n-1} (b_i + q_i A_n)((-1)^{n+i} A_i) + s'_{n-1} A_n \\ &= \sum_{i=1}^{n-1} (-1)^{n+i} b_i A_i + (s'_{n-1} + \sum_{i=1}^{n-1} (-1)^{n+i} q_i A_i) A_n. \end{aligned}$$

Therefore,

$$s'_{n-1} + \sum_{i=1}^{n-1} (-1)^{n+i} q_i A_i = \frac{d' - \sum_{i=1}^{n-1} (-1)^{n+i} b_i A_i}{A_n} = b_n,$$

which is an integer. Thus,

$$d' = \sum_{i=1}^{n} (-1)^{n+i} b_i A_i = \det(B).$$

This proves that B is a completion of A. The bounds in (3.10) and (3.11) are immediate from Theorem 3.1. □

Theorem 3.3 *Suppose that the algorithm used for computing A_i has worst-case complexity $c(a_i, a'_i)$ and that the upper bound used for $|A_i|$ is $w(a_i, a'_i)$, where a_i, a'_i are the two largest absolute values among the entries in A_i. Then the worst-case complexity of Algorithm B is $O(n\,(c(a_0, a'_0) + n \log^2 w(a_0, a'_0)))$, where a_0 and a'_0 are the two largest among all the absolute values $|a_i|$.*

Proof. There are many algorithms for computing integral determinants and many propositions establishing upper bounds for the absolute values of determinants. Suppose that the algorithm used for computing A_i has worst-case complexity $c(a_i, a'_i)$ bit operations. Then the worst-case complexity of computing all A_i $(1 \le i \le n)$ is

$$O(n\, c(a_0, a'_0)).$$

Suppose that the upper bound used for $|A_i|$ is $w(a_i, a_i')$. Then all $|A_i|$ $(1 \le i \le n)$ are bounded above by $w(a_0, a_0')$. Note that there are $O(n)$ multiplications and divisions in computing each of b_i and that the magnitudes of all intermediate factors appearing in computing b_i are bounded above by $w(a_0, a_0')$ due to the modulus value A_n. By Lemmas 2.2 and 2.3, the computations of all b_i $(1 \le i \le n)$ need no more than

$$O(n\, c(a_0, a_0')) + O((n \log w(a_0, a_0'))^2) = O(n\,(c(a_0, a_0') + n \log^2 w(a_0, a_0')))$$

bit operations. □

4 The general case $k < n$

From the previous sections we know that there are formulas for the efficient matrix completion of a given $k \times n$ matrix A, in two extreme cases. To the authors' knowledge, no such formulas are known for the general case. Therefore, algorithmic solutions have been sought. For example, Newman [6] has pointed out that one can use the process of reducing a matrix over a PID into its Smith normal form by row and column elementary operations. Unfortunately, even when the principal ideal domain is the special ring of integers, the time complexity of the algorithm Newman proposes does not seem to be polynomial because of the explosive growth of the magnitudes of the entries of intermediate matrices. Our guess is that it would be exponential in k. So we turn our attention to alternate methods.

Let $k < n$ and $A \in \mathbb{Z}^{k\times n}$ of rank k. It was proved, see, for example Newman [6], that there are matrices $V \in \mathbb{Z}^{n\times n}$ and $H \in \mathbb{Z}^{k\times n}$ such that

$$AV = H, \tag{4.1}$$

where V is unimodular, and H lower triangular with the property that its main diagonal entries are positive and the value of any of its off-diagonal entries below the main diagonal is between 0 and the value of the diagonal entry in the same row. Such a matrix H is unique and is the well known **Hermite normal form** of A. Matrix V is called a **transformation matrix of** A into its Hermite normal form.

The following theorem provides a way of employing the Hermite normal form of A in the completion of A.

Theorem 4.1 *Let $k < n$ and $A \in \mathbb{Z}^{k\times n}$ of rank k. Let H be the Hermite normal form of A, and V a transformation matrix of A into H. Let B denote the matrix formed by the last $n-k$ rows of V^{-1}. Then the compound matrix $\binom{A}{B}$ is a completion of A.*

Proof. Write

$$H = (G_k\ 0_{k\times(n-k)}),$$

where G_k is a lower triangular matrix of order k and $0_{s\times t}$ denotes the $s \times t$ zero matrix. Let

$$K = \begin{pmatrix} G_k & 0_{k\times(n-k)} \\ 0_{(n-k)\times k} & I_{(n-k)} \end{pmatrix}.$$

Then

$$KV^{-1} = \begin{pmatrix} A \\ B \end{pmatrix}.$$

Since V is unimodular, we have $V^{-1} \in \mathbb{Z}^{n\times n}$ and $|\det(V)| = |\det(V^{-1})| = 1$. Therefore,

$$|\det(\begin{pmatrix} A \\ B \end{pmatrix})| = |\det(K)| = |\det(G_k)|.$$

Since the gcd of the k^{th} order minors of A does not change after A is multiplied by a unimodular matrix, we conclude that $|\det(G_k)|$ and the gcd of the k^{th} order minors of A are the same. □

There are known polynomial time algorithms for computing both H and V for given integer matrices. Let HNF denote any one of these algorithms, and let $O(f(k, n, ||A||))$ denote the time complexity, in bit operations, of the algorithm on the input matrix $A = (a_{ij}) \in \mathbb{Z}^{k\times n}$, where the norm of A is defined as

$$||A|| := \max_{1\le i\le k, 1\le j\le n} |a_{ij}|.$$

There are also known polynomial time algorithms for computing the rational inverse of nonsingular integer matrices. Let INV denote any one such algorithm, and let $O(g(n, ||C||))$ denote the time complexity in bit operations of the algorithm on the input matrix $C = (c_{ij}) \in \mathbb{Z}^{n\times n}$.

Then HNF and INV can be used to solve the matrix completion problem as shown in the following algorithm.

Algorithm C.

Input: $A \in \mathbb{Z}^{k\times n}$ of rank k.

Output: $B \in \mathbb{Z}^{(n-k)\times n}$ such that the compound matrix $\begin{pmatrix} A \\ B \end{pmatrix}$ is a completion of A.

Step 1. Employ HNF on A to obtain the Hermite normal form H of A and a transformation matrix V such that $AV = H$.

Step 2. Employ INV to compute V^{-1}.

Step 3. Output as B the submatrix of V^{-1} formed by its last $n - k$ rows, then stop.

The correctness of Algorithm C is immediate from Theorem 4.1. As for its time complexity, it is stated in the following theorem which is also immediate from Theorem 4.1.

Theorem 4.2 *Suppose that for the matrix V produced by algorithm HNF on A we have $||V|| = O(h(||A||))$. Then the time complexity in bit operations of algorithm C is*

$$O(f(k, n, ||A||) + g(n, h(||A||))).$$

The generality of Theorems 4.1 and 4.2 allows us to produce a number of concrete algorithms. As an example, we state one of them below.

We need some notation. Let $O(\mu(t))$ denote the time complexity in bit operations of multiplying two integers, each of t bits, and let $O(k^\theta)$ denote the time complexity in integer operations of multiplying two integer matrices of order k. It is well known that θ can be 2.38 and $\mu(t) = O(t^2)$. Let

$$O^\sim(f(x)) := O(f(x)\log^c f(x)),$$

where $c > 0$ is a constant.

Let us take as HNF the algorithm described in Storjohann [8] or Storjohann and Labahn [9], and let INV be the algorithm described in Villard [10]. Let $A \in \mathbb{Z}^{k\times n}$. The former has the time complexity, in bit operations, equal to

$$O^\sim(k^{\theta-1}n \cdot \mu(k \log ||A||)),$$

and produces V with $||V|| \leq (\sqrt{k}\,||A||)^k$. The latter has time complexity, in bit operations, equal to

$$O^\sim(n^{\theta+1} \log ||V||).$$

In this case, the time complexity, in bit operations, for completing A is

$$\begin{aligned} & O^\sim(k^{\theta-1}n \cdot \mu(k \log ||A||)) + O^\sim(n^{\theta+1} \log ||V||) \\ = \; & O^\sim(k^{\theta-1}n \cdot \mu(k \log ||A||)) + O^\sim(n^{\theta+1} k \log(\sqrt{k}\,||A||)) \\ = \; & O^\sim(kn^{\theta+1}(\log^2 ||A|| + \log k)). \end{aligned}$$

Consequently we see that by combining appropriate HNF and INV algorithms, solving the general *matrix completion problem* can be achieved in time polynomial in n, k, and $\log ||A||$. Thus, extending a primitive set of lattice vectors to a basis can be achieved in polynomial time in the same parameters.

Acknowledgments

The authors would like to thank the anonymous referee for her/his valuable comments.

References

[1] Eric Bach and Jeffrey Shallit, *Algorithmic Number Theory, Volume 1: Efficient Algorithms*, The MIT Press, Cambridge, Massachusetts, 1997.

[2] J. W. S. Cassels, *An Introduction to the Geometry of Numbers* Springer, Berlin, 1997.

[3] Neal Koblitz, *Algebraic Aspects of Cryptography*, Springer, Berlin, 1998.

[4] C. G. Lekkerkerker and P. M. Gruber, *Geometry of Numbers*, Second edition North-Holland Mathematical Library, 37, North-Holland Publishing Co., 1987.

[5] Daniele Micciancio and Shafi Goldwasser, *Complexity of Lattice Problems, A Cryptographic Perspective*, Kluwer Academic Publishers, Boston, 2002

[6] Morris Newman, *Integral Matrices*, Academic Press, New York, 1972.

[7] Carl Ludwig Siegel, *Lectures on the Geometry of Numbers*, Springer-Verlag, 1989.

[8] Arne Storjohann, Algorithms for Matrix Canonical Forms, 2000.

[9] Arne Storjohann and George Labahn, Asymptotically Fast Computation of the Hermite Normal Form of an Integer Matrix, in *Proc. Int'l Symp. on Symbolic and Algebraic Computation: ISSAC '96 (ed. Y. N. Lakshman)*, ACM Press, 1996.

[10] Gilles Villard, Exact computations on polynomial and integer matrices, `http://www.ens-lyon.fr/gvillard`

(Received 2 Feb 2007; revised 30 Aug 2007)