Faster Algorithms for Integer Lattice Basis Reduction

Arne Storjohann
Eidgenössische Technische Hochschule
CH-8092 Zürich
storjoha@inf.ethz.ch

July 1996

Abstract

The well known L^3-reduction algorithm of Lovász transforms a given integer lattice basis $b_1, b_2, \ldots, b_n \in \mathbb{Z}^n$ into a reduced basis. The cost of L^3-reduction is $O(n^4 \log B_o)$ arithmetic operations with integers bounded in length by $O(n \log B_o)$ bits. Here, B_o bounds the Euclidean length of the input vectors, that is, $B_o \geq |b_1|^2, |b_2|^2, \ldots, |b_n|^2$. We present a simple modification of the L^3-reduction algorithm that requires only $O(n^3 \log B_o)$ arithmetic operations with integers of the same length. We gain a further speedup by combining our new approach with Schönhage's modification of the L^3-reduction algorithm and incorporating fast matrix mutliplication techniques. The result is an algorithm for semi-reduction that requires $O(n^{2.381} \log B_o)$ arithmetic operations with integers of the same length.

1 Introduction

A subset L of row vectors in $\mathbb{Z}^{n'}$ is an n-dimensional *integer lattice* precisely if there exists some rank n integer matrix $A \in \mathbb{Z}^{n \times n'}$ such that L is equal to the set of all integer linear combinations of rows of A. Conversely, the row vectors of a rank n matrix $A \in \mathbb{Z}^{n \times n'}$ comprise a *basis* for some n-dimensional lattice L. The rows of a second matrix $B \in \mathbb{Z}^{n \times n'}$ form a basis for the same lattice as that of A if and only if A and B are *left equivalent*, that is, if and only if A and B are related by a unimodular matrix U with $A = UB$ and $B = U^{-1}A$. (Recall that a square matrix U over $\mathbb{Z}$ is unimodular if $\det(U) = \pm 1$; such a matrix has the property that U^{-1} is over $\mathbb{Z}$.) An integer lattice basis reduction algorithm takes as input a full row rank integer matrix such as

$$A = \begin{bmatrix} 33554516 & 3750842 & -8343524 & 21489465 & 13970499 \\ 25456939 & 2845665 & -6330013 & 16303498 & 10599055 \\ 10552673 & 1179613 & -2623983 & 6758294 & 4393630 \\ 10628092 & 1188047 & -2642738 & 6806596 & 4425031 \end{bmatrix}$$

and returns a new integer matrix that is left equivalent to A but consists of (typically) *shorter* basis vectors. Here, the length $|b|$ of an integer vector b is the Euclidean length, that is, $|b|^2 = \langle b, b \rangle$ where $\langle \cdot, \cdot \rangle$ denotes the usual inner product. The L^3-reduction algorithm presented in [12] guarantees to return a basis with initial vector at most $2^{n/2}$ times the length of the shortest vector in the lattice. In practice, the performance of the L^3-reduction algorithm is much better. For example, the L^3-reduction algorithm returns, with the above input matrix A, the left equivalent matrix

$$R = \begin{bmatrix} -22 & 35 & -64 & -6 & -67 \\ -57 & 59 & -45 & 8 & 93 \\ 28 & 114 & -8 & 43 & -4 \\ -42 & 36 & 118 & -28 & -3 \end{bmatrix}$$

Integer lattice basis reduction has many applications including, for example, diophantine approximation [12], finding integer relations among real numbers [9], computing other bases for integer lattices such as the Hermite and Smith normal form [10].

In this paper we present algorithms for the reduction and semi-reduction of integer lattice basis. We prove asymptotic running time bounds for our algorithms that improve on those of the original L^3-reduction algorithm presented in Lenstra, Lenstra & Lovász [12] and the semi-reduction algorithm of Schönhage [14]. Before summarizing our complexity results we define precisely what we mean by *reduction* and *semi-reduction* — to do this we need to recall the Gram-Schmidt orthogonalization process. Let $b_1, b_2, \ldots, b_n \in \mathbf{Z}^{n'}$ be an integer lattice basis. Gram-Schmidt orthogonalization determines the associated orthogonal basis $b_1^*, b_2^*, \ldots, b_n^*$ together with elimination factors μ_{ij} $(1 \le j < i \le n)$ defined inductively by

$$b_i^* = b_i - \sum_{j=1}^{i-1} \mu_{ij} b_j^* \tag{1}$$

and

$$\mu_{ij} = \langle b_i, b_j^* \rangle / |b_j^*|^2. \tag{2}$$

Following Schönhage in [14] we define some different meanings of reduced. A vector b_k (or the whole basis) is called *size-reduced* if $|\mu_{ki}| \le 1/2$ holds for all $i < k$ (so for all k). A basis is called *2-reduced* if $|b_k^*|^2 \le 2|b_{k+1}^*|^2$ for $1 \le k < n$. As in [12], "reduced" is used to mean 2-reduced and reduced in size. Our first reduction algorithm is a modification of the L^3-reduction algorithm. Let $b_1, b_2, \ldots, b_n \in \mathbf{Z}^{n'}$ be an integer lattice basis with $n' = O(n)$ and $|b_i|^2 \le B_o$ for $1 \le i \le n$. Our algorithm requires $O(n^{4+\epsilon}(\log B_o)^{2+\epsilon})$ bit operations (where $\epsilon > 0$) to produce a reduced bases. This complexity result improves by a factor of $O(n)$ the running time proven for the L^3-reduction algorithm in [12]. Schnorr [13] has proposed a modification of the L^3-reduction algorithm which requires $O(n^{5+\epsilon} \log B_o + n^4(\log B_o)^{2+\epsilon})$ bit operations.

For some applications a complete 2-reduction of the input bases is not required. A bases $b_1, b_2, \ldots, b_n$ is called *semi-reduced* if

$$|b_r^*|^2 \le 2^{n+s-r} |b_s^*|^2 \quad \text{for} \quad 1 \le r < s \le n. \tag{3}$$

Schönhage has given a modification of the L^3-reduction algorithm that requires $O(n^{4+\epsilon}(\log B)^{2+\epsilon})$ bit operations to return a semi-reduced bases. By combining our modification of the L^3-reduction algorithm with Schönhage's and incorporating fast matrix multiplication techniques we are able to prove a running time of $O(n^{3+1/(5-\theta)+\epsilon}(\log B_o)^{2+\epsilon})$ bit operations for semi-reduction. This complexity result assumes a pre- and post-conditioning step which requires $O(n^{\theta+1+\epsilon}(\log B_0)^{1+\epsilon})$ bit operations — about the same time as required to compute the determinant of the input lattice. Here θ is the exponent for matrix multiplication over rings: two $n \times n$ matrices over a ring R can be multiplied in $O(n^\theta)$ ring operations. Using standard matrix multiplication $\theta = 3$ whereas $\theta < 2.376$ using the algorithm of Coppersmith and Winograd [5]. In any case, we achieve a speedup for semi-reduction ranging from $O(n^{.5})$ for the case $\theta = 3$ to $O(n^{0.618})$ for the case $\theta = 2.376$. We also present a version of our semi-reduction algorithm which uses standard integer and matrix multiplication. Under these assumptions we prove a running time of about $O(n^{4+1/3}(\log B_0)^3)$ bit operations (ignoring logarithmic terms).

The rest of this paper is organized as follows. In Section 2 we define some notation and discuss the complexity model used throughout the paper. In Section 3 we present an asymptotically fast algorithm for fraction free Gaussian elimination over $\mathbf{Z}$. We require this to compute the initial Gram-Schmidt orthogonalization of the input basis. In Section 4 we present our modification of the L^3-reduction algorithm and in Section 5 our asymptotically fast algorithm for semi-reduction.

2 Preliminaries

In this section we discuss the complexity model used throughout the paper. Our main goal is to rigorously prove asymptotically fast complexity results for the algorithms in question — to this end we call upon pseudo-linear integer arithmetic and frequently incorporate fast matrix multiplication techniques in our algorithms. However, our secondary goal is always the development of fast practical algorithms — to this end we don't assume, for example, that integer multiplication will always be pseudo-linear. Whenever possible, we develop our algorithms to use homomorphic imaging schemes which greatly reduce the number of operations on large integers.

Integer Arithmetic Using an algorithm of Schönhage & Strassen [15], two $\lceil t \rceil$-bit integers can be multiplied in $O(t(\log t)(\log\log t))$ bit operations. For brevity, we prefer to write this complexity result as $O(t^{1+\epsilon})$ where ϵ is understood to be some positive constant. For example, algorithms of Karatsuba & Ofman [11] allow $\epsilon = \log_2 3 < 1.585$ or $\epsilon = \log_3 5 < 1.465$. In the case of standard integer arithmetic we have $\epsilon = 1$. Each of the following operations can also be performed in $O(t^{1+\epsilon})$ bit operations: multiplying two $\lceil t \rceil$-bit integers; computing the division with remainder of two $\lceil t \rceil$-bit integers; computing the the greatest common divisor of two $\lceil t \rceil$-bit integers; and computing either direction of the homomorphism implied by the Chinese remainder algorithm where the product of the modulii is bounded by $\lceil t \rceil$ bits (see, for example, Aho, Hopcroft & Ullman [1]). In particular, a single field operation (including division) from $\mathbf{Z}_p$, the field of integers modulo a prime p, costs $O((\log p)^{1+\epsilon})$ bit operations.

Note that we will not "sweep under the rug" other logarithmic terms in complexity results with an ϵ; the parameter ϵ is reserved as the exponent for integer multiplication. This will lead to some inelegancies in the statements and proofs of some complexity results, but the extra effort will prove worthwhile. For example, the algorithms we give for Gram-Schmidt orthogonalization and size-reduction of integer lattices basis are the currently fastest known, not only asymptotically assuming $\epsilon \to 0$ and $\theta = 2.37$, but also in the case of standard integer and matrix multiplication.

Matrix Arithmetic Let $\mathsf{M}(n,k,m)$ be a bound on the number of ring operations required to multiply an $n \times k$ by a $k \times m$ matrix over a principal ideal ring R. We assume that

$$\mathsf{M}(n,n,n) \ll n^\theta$$

where $2 < \theta \le 3$. The current record on θ is $\theta < 2.376$ using the algorithm of Coppersmith & Winograd [5]. Using the obvious block decomposition we also have

$$\mathsf{M}(n,m,n) \ll \begin{cases} n^2 m^{\theta-2} & \text{for} \quad n \ge m \\ n m^{\theta-1} & \text{for} \quad n < m \end{cases}$$

We will also use the fact that the problem of computing the determinant of an $n \times n$ matrix, computing the inverse of a nonsingular $n \times n$ matrix, and multiplying two $n \times n$ matrices over a field have been shown to be computationally equivalent (see, for example, Winograd [17] or Aho, Hopcroft & Ullman [1]).

For a real matrix A we write A_{ij} to denote the entry in row i column j, $||A||$ to denote the absolute value of the largest magnitude entry, and A^t to denote the transpose. Given two integers a and M with M positive, we write $\mathrm{mods}(a, M)$ to mean the unique integer r congruent to a modulo M in the symmetric range, that is, with $-\lfloor (M-1)/2 \rfloor \le r \le \lfloor M/2 \rfloor$. The closest integer to a real number x is denoted by $[x]$ (in case of ambiguity choose the smallest).

We will make free use of the following results throughout the paper.

Lemma 1 (Giesbrecht [7]) *Let $x \ge 3$ and $l = 6 + \log\log x$. Then there exist at least $2\lceil\lceil \log_2(2x)\rceil/(l-1)\rceil$ primes p such that $2^{l-1} < p < 2^l$.* ■

Theorem 2 *Let $A \in \mathbf{Z}^{n\times k}$ and $B \in \mathbf{Z}^{k\times n}$ and let $(\beta-1)/2$ be a bound on the magnitudes of entries in A, B and AB. Then the matrix product $C = AB$ can be computed in $O(\mathsf{M}(n,k,m)(\log\beta)(\log\log\beta)^\epsilon + (nm+kn+km))(\log\beta)^{1+\epsilon})$ bit operations.*

Proof. For p a prime, let $A_p = A \bmod p$ be the matrix obtained from A by replacing each integer entry with its image mod p; we consider A_p to be over the field $\mathbf{Z}_p$ of integers modulo p. The algorithm computes $C = AB$ using the standard homomorphic imaging scheme: Compute $C_p = (AB)_p$ over $\mathbf{Z}_p$ for sufficiently many primes p to allow recovery of the integer coefficients appearing in C via the Chinese remainder algorithm. Since entries in C are bounded by β, an application of Lemma 1 shows that we can choose all our primes to be $l = 6 + \log\log\beta$ bits in length. It follows that we can choose a list of $s = 2\lceil\lceil(\log 2\beta)\rceil/(l-1)\rceil = \Theta((\log\beta)/l)$

distinct primes $(p_i)_{1\le i\le s}$ that are bounded in length by l bits and that satisfy $\prod_{1\le i\le s} p_i > \beta$. The algorithm can now be described as follows: (1) Find the images $(A_{p_i}, B_{p_i})_{1\le i\le s}$ at a cost of $O((nk+km)(\log B)^{1+\epsilon})$ bit operations; (2) For $1 \le i \le s$, compute $C_p = A_pB_p$ over $\mathbb{Z}_p$ at a cost of $O(s\cdot \mathsf{M}(n,k,m)\cdot l^{1+\epsilon}) = O(\mathsf{M}(n,k,m)(\log\beta)(\log\log\beta)^{\epsilon})$ bit operations; (3) Apply Chinese remaindering to recover modulo $\prod_{1\le i\le s} p_i$ in the symmetric range the $O(nm)$ integer coefficients of C a cost of $O(nm(\log\beta)^{1+\epsilon})$ bit operations. ∎

Corollary 3 *Let $A \in \mathbb{Z}^{n\times k}$ and $B \in \mathbb{Z}^{k\times n}$ and let $(\beta-1)/2$ be a bound on the magnitudes of entries in A, B and AB. Then the matrix product $C = AB$ can be computed in $O(\mathsf{M}(n,k,m))$ arithmetic operations with integers bounded in length by $O(\log\beta)$ bits.*

Proof. Let A_d, B_d and C_d be the the matrices A, B and C respectively with entries considered embedded in the principal ideal ring $\mathbb{Z}_d$ where $d = \lceil\beta\rceil$. The matrix C is recovered from C_d by reducing modulo d (symmetric range) all entries. C_d is computed in $\mathsf{M}(n,k,m)$ ring operations from $\mathbb{Z}_d$. ∎

3 Asymptotically Fast Fraction Free Gaussian Elimination

Let R be a principal ideal domain. A key step in the algorithm of the next section is to compute, for a given $A \in \mathsf{R}^{n\times n}$, a lower triangular matrix $F \in \mathsf{R}^{n\times n}$ such that the matrix $T = FA$ is upper triangular with i-th diagonal entry the determinant of the principal i-th submatrix of A for $1 \le i \le n$. If A is definite (all principal submatraces of A are nonsingular) then F is unique and T is precisely the matrix obtained by applying fraction free Gaussian elimination *without* row pivoting to A. (For a thorough discussion of fraction free Gaussian elimination see, for example, Geddes, Czapor & Labahn [6] or the original articles by Bareiss [2, 3].) In this section we define explicitly the entries of the matrices F and T and give an asymptotically fast algorithm for their computation for the case $\mathsf{R} = \mathbb{Z}$. First recall some basic definitions and facts from linear algebra. For a matrix $A \in \mathsf{R}^{n\times n}$, the minor M_{ij} of entry a_{ij} is defined to be the determinant of the submatrix obtained by deleting the i-th row and j-th column of A and the cofactor C_{ij} is given by $C_{ij} = (-1)^{i+j}M_{ij}$. In general, an $i\times i$ minor of A is the determinant of an $i\times i$ submatrix of A. For brevity, we will sometimes write $A_{[1\cdots i,1\cdots j]}$ to indicate the submatrix comprised of the first i rows and first j columns of A, and $\mathrm{row}(A,k)$ (respectively $\mathrm{col}(A,k)$) to denote the k-th row (respectively column) of A.

Definition 1 *For an $n\times n$ matrix A over a principal ideal domain R, we define $F = \mathrm{FF}(A)$ to be the $n\times n$ lower triangular matrix over R with F_{ij} equal to the cofactor of the element in the j-th row, i-th column of the i-th principal submatrix of A for $1 \le j \le i \le n$.*

Fact 1 *Let A be an $n\times n$ matrix over R with adjoint A^{adj}. Entries of A^{adj} are given by $A^{\mathrm{adj}}_{ij} = C_{ji}$ for $1\le i\le n$ and $1\le j\le n$ and the determinant of A can be written according to the j-th column expansion as*

$$\det(A) = a_{1j}C_{1j} + a_{2j}C_{2j} + \cdots + a_{nj}C_{nj}.$$

Lemma 4 *Given an $A \in \mathsf{R}^{n\times n}$ let $F = \mathrm{FF}(A)$. Then, the matrix $T = FA$ will be upper triangular with T_{ij} equal to the minor formed from rows $1,2,\ldots,i$ and columns $1,2,\ldots,i-1,j$ of A for $1\le i\le j\le n$.*

Proof. Follows from Fact 1 by noting that the entries of FA are the claimed entries for T — which are minors of A — written according to their cofactor column expansion. ∎

Remark 1 *The first k entries in $\mathrm{row}(\mathrm{FF}(A),k)$ are precisely those in the last row of the adjoint of the principal k-th submatrix of A (for $1\le k\le n$).*

When A is definite, we can relate the matrices F and T to the well known LU decomposition which expresses A as the product of a unit lower triangular L and an upper triangular U. (To do this we need to work over the quotient field of R.) In particular, $F = DL^{-1}$ and $T = DU$ where D is the $n\times n$ diagonal matrix with i-th diagonal entry the determinant d_{i-1} of the $(i-1)$-th principal submatrix of A (with $d_0 = 1$). Note that d_i $(1\le i\le n)$ can also be recovered as the product of the first i diagonal entries in U. However, while the LU decomposition is defined only when A is definite, the matrices F and T of Lemma 4 are well defined even when A is non-definite. This is an important observation since our algorithm for computing $\mathrm{FF}(A)$ for an integer matrices uses a homomorphic imaging scheme which may map a definite A to a nondefinite image. (If we knew in advance that A was definite, we could recover $\mathrm{FF}(A)$ from the LU decomposition of A which can be computed using the asymptotically fast triangularization algorithm of Bunch & Hopcroft [4]). The main purpose of this section is to give an algorithm for computing $\mathrm{FF}(A)$ that is robust even when A is nondefinite. Our algorithm hinges on the following two somewhat technical results.

Lemma 5 *Let A be an $n \times n$ matrix over a principal ideal ring R and let P be an $n \times n$ matrix over R with $\det(P) = 1$. If P can be written in block diagonal form as $\mathrm{diag}(P', I_{n-m})$ for some m with $0 \le m \le n$, then $\mathrm{row}(\mathrm{FF}(A), m+i) = \mathrm{row}(\mathrm{FF}(PA)P, m+i)$ for $0 \le i \le n-m$.*

Proof. Fix some index i with $0 \le i \le n-m$ and let A_1 and P_1 be the principal $(m+i)\times(m+i)$ submatrices of A and P respectively. Because of the special structure of P, and since by definition entries in $\mathrm{row}(\mathrm{FF}(A), m+i)$ are determined entirely from entries in the principal $(m+i)$-th submatrix of A, the first $m+i$ entries in $\mathrm{row}(\mathrm{FF}(A), m+i)$ and $\mathrm{row}(\mathrm{FF}(PA)P, m+i)$ will be given by $\mathrm{row}(\mathrm{FF}(A_1), m+i)$ and $\mathrm{row}(\mathrm{FF}(P_1A_1)P_1, m+i)$ respectively. Thus it will suffice to show that $\mathrm{row}(\mathrm{FF}(A_1), m+i) = \mathrm{row}(\mathrm{FF}(P_1A_1)P_1, m+i)$. Keeping in mind Remark (1), we get

$$\begin{aligned}
\mathrm{row}(\mathrm{FF}(P_1A_1)P_1, m+i) &= \mathrm{row}(\mathrm{FF}(P_1A_1), m+i)P_1 \\
&= \mathrm{row}((P_1A_1)^{\mathrm{adj}}, m+i)P_1 \\
&= \mathrm{row}((A_1^{\mathrm{adj}}P_1^{\mathrm{adj}}, m+i)P_1 \\
&= \mathrm{row}(A_1^{\mathrm{adj}}, m+i)P_1^{\mathrm{adj}}P_1 \\
&= \mathrm{row}(A_1^{\mathrm{adj}}, m+i) \\
&= \mathrm{row}(\mathrm{FF}(A_1), m+i)
\end{aligned}$$

as required. ∎

Lemma 6 *Let A be an $n \times n$ matrix over a field $\mathbf{F}$ and let $F = \mathrm{FF}(A)$. Fix indices m and k with $1 \le m \le n$ and $0 \le k \le n-m$. If $A_{[1\cdots m+k, 1\cdots m]}$ has rank less than m, then entries in row($F, m+i$) are all zero for $1 \le i \le k$.*

Proof. If $k = 0$ then the claim is vacuously true so assume that the condition of the lemma holds for some $k \ge 1$. Fix some i with $1 \le i \le k$. By definition, entries in $\mathrm{row}(F, m+i)$ are multiples of $(m+i-1)\times(m+i-1)$ minors of $A_{[1\cdots m+i, 1\cdots m+i-1]}$. Thus, it suffices to show that the rank of $A_{[1\cdots m+i, 1\cdots m+i-1]}$ is less than $m+i-1$. Now, $A_{[1\cdots m+i, 1\cdots m+i-1]}$ can have full column rank only if all columns in $A_{[1\cdots m+i, 1\cdots m+i-1]}$ are linearly independent. But the submatrix comprised of the first m columns of $A_{[1\cdots m+i, 1\cdots m+i-1]}$ is a submatrix of $A_{[1\cdots m+k, 1\cdots m]}$, which, by assumption, has rank less than m. This shows that the rank of $A_{[1\cdots m+i, 1\cdots m+i-1]}$ must be less than $m+i-1$. ∎

Lemma 7 *There exists a deterministic algorithm that takes as input an $n \times m$ matrix A over a field $\mathbf{F}$, with $m \le n$ and with principal m-th minor singular, and returns as output the largest index k ($0 \le k \le n-m$) such that the submatrix comprised of the first $m+k$ rows of A has rank less than m. The cost of the algorithm is $O(nm^{\theta-1}\log n)$ field operations from $\mathbf{F}$. In the case of standard matrix multiplication, the cost of the algorithm is $O(m^3 + nm)$ field operations.*

Proof. By augmenting A with at most $n-m$ rows of zeros, we may assume, without loss of generality, that $n - m = 2^p$ for some integer $p \ge 1$. We show how to determine k by computing the rank of at most $\log_2(n-m)$ submatrices of A. Each of these rank computations requires at most $O(nm^{\theta-1})$ field operations so the claimed complexity bound will follow. To begin, we have $1 \le k \le n-m$; a range with width $n-m$. Our approach is to do a binary search of this range — each step will reduce the width of the possible range for k by half so that the total number of steps is bounded by $p = \log_2(n-m)$. The entire procedure can be understood by considering the first step, which proceeds as follows. Let $n_1 = (n-m)/2$ and compute the rank r of the submatrix comprised of the first $m+n_1$ rows of A. If $r = m$ then k must satisfy $1 \le k \le n_1$.

If $r < m$ then k must satisfy $n_1 + 1 \leq k \leq n$. In both cases we have halved the width of the possible range for k.

The result for standard matrix matrix multiplication is obtained more easily. Triangularize using Gaussian elimination, for $i = 0, 1, 2, \ldots, n - m$ in succession, the submatrix $A_{[1\cdots m+i,1\cdots m]}$ and let k be the largest i for which $A_{[1\cdots m+i,1\cdots m]}$ has rank less that m. The triangularization of $A_{[1\cdots m,1\cdots m]}$ requires at most $O(m^3)$ field operations. Each additional submatrix $A_{[1\cdots m+i,1\cdots m]}$ for $1 \leq i \leq n - m$ can be computed within $O(m)$ field operations for a maximum of $O((n - m)m) = O(nm)$ field operations. ∎

Theorem 8 *There exists a deterministic algorithm that takes as input an $n \times n$ matrix A over a principal ideal domain R and returns as output the matrices $F = \mathrm{FF}(A)$ and $T = FA$. The cost of the algorithm is $O(n^\theta \log n)$ ring operations from R. In the case of standard matrix multiplication, the cost of the algorithm is $O(n^3)$ ring operations.*

Proof. The presentation of the algorithm is greatly simplified if we assume we are working over a field. We can do this without loss of generality by working over the quotient field of R, the definition of which we recall now. Define an equivalence relation $\sim$ on the set of 2-tuples (a, b) of elements of R with $b \neq 0$ by

$$(a_1, b_1) \sim (a_2, b_2) \quad \text{if and only if} \quad a_1 b_2 = a_2 b_1.$$

The quotient field $\bar{\mathsf{R}}$ of R is this set of equivalence classes, with addition and multiplication defined by

$$\begin{aligned} [(a_1, b_1)] + (a_2, b_2)] &= [(a_1 b_2 + a_2 b_1, b_1 b_2)] \\ [(a_1, b_1)] \cdot [(a_2, b_2)] &= [(a_1 a_2, b_1 b_2)] \end{aligned}$$

so that a single field operation in $\bar{\mathsf{R}}$ requires a constant number of ring operations from R. The entries in $\mathrm{FF}(A)$ are defined uniquely in terms of minors of A and so can be recovered from the entries in $\mathrm{FF}(\bar{A})$ where $\bar{A}$ is the matrix A considered embedded into $\bar{\mathsf{R}}^{n\times n}$. This shows that without loss of generality we may assume our input matrix A to be over some field $\mathbf{F}$.

Let $C(n)$ be the number of field operations from $\mathbf{F}$ required to compute, for an $A \in \mathbf{F}^{n\times n}$, the matrix $F = \mathrm{FF}(A)$. By augmenting A with at most $n - 1$ rows and columns of zeros we may assume, without loss of generality, that n is a power of 2. We claim that

$$C(n) \leq 3C(n/2) + cn^\theta \log n \tag{4}$$

for some absolute constant c. To prove (4) we give a conquer and divide algorithm which requires at most three recursive calls on $(n/2) \times (n/2)$ matrices and additional work at most one call to the algorithm of Lemma 6 plus a constant number of matrix operations (determinant, inversion, multiplication and addition) involving matrices bounded in dimension by n.

To begin, write A using a block decomposition as

$$A = \left[\begin{array}{c|c} A_1 & A_3 \\ \hline A_2 & A_4 \end{array}\right] \tag{5}$$

where each block is $m \times m$ with $m = n/2$. The principal m-th submatrix of F is given by $F_1 = \mathrm{FF}(A_1)$ which we compute recursively. To compute the remaining rows of F we have two two cases depending on whether or not $d = \det(A_1)$ is zero. Note that d can be recovered as the m-th diagonal entry of $F_1 A_1$.

Case 1 ($d \neq 0$): Compute $A_4' \leftarrow -A_2 A_1^{-1} A_3 + A_4$ so that

$$\left[\begin{array}{c|c} I_m & \\ \hline -A_2 A_1^{-1} & I_m \end{array}\right] \left[\begin{array}{c|c} A_1 & A_3 \\ \hline A_2 & A_4 \end{array}\right] = \left[\begin{array}{c|c} A_1 & * \\ \hline & A_4' \end{array}\right] \tag{6}$$

with each block $m \times m$. Compute $F_4' = \mathrm{FF}(A_4')$ recursively. Multiply equation (6) on the left by $\mathrm{diag}(F_1, dF_4')$ to obtain

$$\left[\begin{array}{c|c} F_1 & \\ \hline -dF_4' A_2 A_1^{-1} & dF_4' \end{array}\right] \left[\begin{array}{c|c} A_1 & A_3 \\ \hline A_2 & A_4 \end{array}\right] = \left[\begin{array}{c|c} T_1 & * \\ \hline & dT_4' \end{array}\right] \tag{7}$$

Since the premultiplier matrix on the left of equation (6) was unit lower triangular, the matrix on the left of (7) is $\mathrm{FF}(A)$. This completes case 1.

Case 2 ($d = 0$): Using the algorithm of Lemma 7, compute the largest index k ($0 \le k \le n - m = m$) such that $A_{[1\cdots m+k,1\cdots m]}$ has rank less than m. By Lemma 6, $\mathrm{row}(F, i)$ contains only zeros for $1 \le i \le k$. If $k = m$ we are finished so assume $k < m$. It remains to compute the last $n-m-k$ rows of F. Using the asymptotically fast triangularization algorithm of Hafner & McCurley [8], compute, at a cost of $O(n^\theta)$ field operations, an $(m + k + 1) \times (m + k + 1)$ unimodular (i.e. nonsingular) matrix P_1 over $\mathbf{F}$ such that $P_1 A_{[1\cdots m+k+1,1\cdots m]}$ is upper triangular. By the assumption on k, the principal m-th submatrix of $P_1 A_{[1\cdots m+k+1,1\cdots m]}$ will be nonsingular. Let P_1' be the matrix obtained from P_1 by multiplying row 1 of P_1 by $1/\det(P_1)$, and set

$$P = \left[\begin{array}{c|c} P_1' & \\ \hline & I_{n-m-k-1} \end{array}\right]$$

so that $\det(P) = 1$ and PA has nonsingular principal m-th submatrix. By Lemma 5 the last $n - m - k$ rows of F can be recovered as the last $n - m - k$ rows of $\mathrm{FF}(PA)P$. Since the principal m-th submatrix of PA is nonsingular we can now proceed as in case 1. Let $B = PA$ and write B using a block decomposition as in (5). Compute $B_4' \leftarrow -B_2 B_1^{-1} B_3 + B_4$ so that

$$\left[\begin{array}{c|c} I_m & \\ \hline -B_2 B_1^{-1} & I_m \end{array}\right] \left[\begin{array}{c|c} B_1 & B_3 \\ \hline B_2 & B_4 \end{array}\right] = \left[\begin{array}{c|c} B_1 & * \\ \hline & B_4' \end{array}\right]. \tag{8}$$

Compute $F_4' = \mathrm{FF}(B_4')$ recursively. The last $n - m - k$ rows of F are given by the last $n - m - k$ rows of

$$\left[\begin{array}{c|c} -d' F_4' B_2 B_1^{-1} & d' F_4' \end{array}\right] P$$

where $d' = \det(B_1)$. T is recovered by computing the matrix product FA. This completes case 2, and proves (4). Iterating (4) yields

$$\begin{aligned} C(n) &\le 3C(n/2) + cn^\theta \log n \\ &\le 4C(n/2) + c(\log n)(n^\theta) \\ &\le 8C(n/4) + c(\log n)(n^\theta + 4(n/2)^\theta) \\ &\vdots \\ &\le 4^{\log_2 n} C(1) + cn^\theta (\log n) \sum_{i=0}^{\log_2 n} (4/2^\theta)^i \\ &\ll n^2 C(1) + n^\theta \log n. \end{aligned}$$

The result now follows by noting that $C(1)$ can be computed in $O(1)$ field operations.

The result in terms of standard matrix multiplication uses exactly the same algorithm. From Lemma 6 we now get

$$C(n) \le 3C(n/2) + cn^3 \tag{9}$$

for some absolute constant c. Iterating (9) now yields $C(n) = O(n^3)$ field operations. ■

Theorem 9 *There exists a deterministic algorithm that takes as input an $n \times n$ integral matrix A and returns as output the matrices $F = \mathrm{FF}(A)$ and $T = FA$. The cost of the algorithm is $O(n^2(\log\beta)^{1+\epsilon} + n^\theta(\log n)(\log\beta)(\log\log\beta)^\epsilon)$ or the simpler bound of $O(n^\theta(\log n)(\log\beta)^{1+\epsilon})$ bit operations where β is a bound on the magnitudes of entries in A, F and T. In the case of standard matrix multiplication, the cost of the algorithm is $O(n^2(\log\beta)^{1+\epsilon} + n^3(\log\beta)(\log\log\beta)^\epsilon)$ bit operations.*

Proof. For p a prime, let $A_p = A \bmod p$ be the matrix obtained from A by replacing each integer coefficient with its image mod p; we consider A_p to be over the field $\mathbf{Z}_p$ of integers modulo p. The algorithm exploits the standard homomorphic imaging scheme: Compute F_p and $T_p = F_p A_p$ over $\mathbf{Z}_p$ for sufficiently many primes p to allow recovery of the integer coefficients appearing in F and T via the Chinese remainder algorithm. Since entries in F and T are bounded by β, an application of Lemma 1 shows that we can choose all our primes to be $l = 6 + \log\log\beta$ bits in length. It follows we can choose a list of $s = 2\lceil\lceil(\log 2\beta)\rceil/(l-1)\rceil = \Theta((\log\beta)/l)$ distinct primes $(p_i)_{1\le i\le s}$ that are bounded in length by l bits and that satisfy $\prod_{1\le i\le s} p_i > \beta$. The algorithm can now be described as follows: (1.) Find the images $(A_{p_i})_{1\le i\le s}$; (2.) For $1 \le i \le s$, compute (F_{p_i}, T_{p_i}) at a cost of $O(s \cdot n^\theta \log n \cdot (\log l)^{1+\epsilon}) = O(n^\theta(\log n)(\log\beta)(\log\log\beta)^\epsilon)$ bit operations using the algorithm of Theorem 8; (3.) Apply Chinese remaindering to recover the $O(n^2)$ integer coefficients of F and T at a cost of $O(n^2 \cdot (\log\beta)^{1+\epsilon})$ bit operations. Note that the complexity of step (1.) will be bounded by that of step (3.), which, in turn, is bounded by that of step (2). Assuming standard matrix multiplication, step (2.) requires $O(n^3(\log\beta)(\log\log\beta)^\epsilon)$ bit operations. ∎

Corollary 10 *There exists a deterministic algorithm that takes as input an n-dimensional lattice of integer vectors $b_1, b_2, \ldots, b_n \in \mathbf{Z}^{n\times n'}$, and returns as output the corresponding Gram Schmidt orthogonalized bases $b_1^*, b_2^*, \ldots, b_n^*$. The cost of the algorithm is $O(n^{3+\epsilon}(\log B_o)^{1+\epsilon} + n^{\theta+1+\epsilon}(\log n)(\log B_0)(\log\log B_0)^\epsilon)$ or the simpler bound of $O(n^{\theta+1}(\log n)(\log B_o)^{1+\epsilon})$ bit operations where $B_o \ge |b_i|^2$ for $1 \le i \le n$. In the case of standard matrix multiplication, the cost of the algorithm is $O(n^{3+\epsilon}(\log B_o)^{1+\epsilon} + n^{\theta+1+\epsilon}(\log B_0)(\log\log B_0)^\epsilon)$ bit operations.*

Proof. Let A be the $n \times n'$ integral matrix with i-th row b_i for $1 \le i \le n$. Then the corresponding Gram Schmidt orthogonalized basis is given by $A^* = D^{-1}FA$ where $F = \mathrm{FF}(AA^t)AA^t$ and D is the $n \times n$ diagonal matrix with $D_{11} = 1$ and $D_{ii} = F_{i-1\,i-1}$ for $2 \le i \le n$. (Note that $D^{-1}F$ will be unit lower triangular.) By Hadamard's inequality, entries in F and D will be bounded in length by $O(n\log B_0)$ bits — this will bound the bit length of entries in FA. The result follows by noting that the matrix product $D^{-1}(FA)$ can be computed within the stated complexity. ∎

4 An Improved L^3-Reduction Algorithm

In this section we assume some familiarity with the L^3-reduction algorithm as presented in [12]. Let an integer lattice basis of row vectors $b_1, b_2, \ldots, b_n \in \mathbf{Z}^{n'}$ be given in the form of an $n \times n'$ matrix A, that is, with $\mathrm{row}(A, i) = b_i$ for $1 \le i \le n$. Before beginning with the basis reduction we require the quantities $d_i = |b_1^*|^2 |b_2^*|^2 \cdots |b_i^*|^2$ and the elimination factors μ_{ij} corresponding to the associated orthogonal basis $b_1^*, b_2^*, \ldots, b_n^*$ — these can be computed directly from the definitions (1) and (2). In practice, we simply triangularize the positive definite symmetric matrix AA^t (where A^t denotes the transpose of A) using fraction free Gaussian elimination without row pivoting to obtain the triangular integer matrix

$$T = \mathrm{FF}(AA^t)AA^t = \begin{bmatrix} d_1 & \bar\mu_{21} & \bar\mu_{31} & & \bar\mu_{n1} \\ & d_2 & \bar\mu_{32} & \cdots & \bar\mu_{n2} \\ & & d_3 & & \bar\mu_{n3} \\ & & & \ddots & \vdots \\ & & & & d_n \end{bmatrix} \tag{10}$$

where $\bar\mu_{ij} = d_j\mu_{ij}$ for $1 \le i < j \le n$. Throughout the algorithm we work with the integer valued $\bar\mu_{ij}$'s and d_i's. The L^3 algorithm does not keep track of the orthogonal basis vectors b_i^* of (1); all that is required is the quantities $B_i = \langle b_i^*, b_i^*\rangle$ and these are given by $B_i = d_i/d_{i-1}$ (for $1 \le i \le n$ and where $d_0 = 1$).

The key to our approach is to consider the L^3 algorithm as a *matrix* algorithm, that is, we consider steps in the L^3 algorithm as corresponding to certain operations on the matrices T and A. At each stage in the algorithm the upper triangular matrix T (i.e. the d_i's and $\bar\mu_{ij}$'s) is related to the current A as in (10). The L^3 algorithm works by applying unimodular row operations to the basis matrix A and then updating the entries in T. There are two types of operations — 2-reduction and size-reduction — with size-reduction being the simpler of the two.

The unimodular row operation corresponding to size-reduction consists of adding an integer multiple of row r to row k of A for some choice of r and k with $1 \le r < k \le n$. The $\bar\mu_{ij}$'s and d_i's are updated by adding the same multiple of column r to column k of T. We get the following (trivial) subroutine which requires $O(n)$ arithmetic operations.

SubtractRow(A, T, k, r, q)
Subtract q times row r from row k of A and update T.
$\mathrm{row}(A, k) \leftarrow \mathrm{row}(A, k) - q\,\mathrm{row}(A, r)$;
$\mathrm{col}(T, k) \leftarrow \mathrm{col}(T, k) - q\,\mathrm{col}(T, r)$;

The unimodular row operation corresponding to 2-reduction consists of switching rows $k-1$ and k in A for some choice of k with $2 \leq k \leq n$. It is easiest to explain with a picture what needs to be updated in the matrix T after a row switch in A. We can write T in block form as

$$T = \left[\begin{array}{ccc|cc|ccc} d_1 & \cdots & \bar{\mu}_{k-2\,1} & \bar{\mu}_{k-1\,1} & \bar{\mu}_{k\,1} & \bar{\mu}_{k+1\,1} & \cdots & \bar{\mu}_{n\,1} \\ & \ddots & \vdots & \vdots & \vdots & \vdots & & \vdots \\ & & d_{k-2} & \bar{\mu}_{k-1\,k-2} & \bar{\mu}_{k\,k-2} & \bar{\mu}_{k+1\,k-2} & \cdots & \bar{\mu}_{n\,k-2} \\ \hline & & & d_{k-1} & \bar{\mu}_{k\,k-1} & \bar{\mu}_{k+1\,k-1} & \cdots & \bar{\mu}_{n\,k-1} \\ & & & & d_k & \bar{\mu}_{k+1\,k} & \cdots & \bar{\mu}_{n\,k} \\ \hline & & & & & d_{k+1} & \cdots & \bar{\mu}_{n\,k+1} \\ & & & & & & \ddots & \vdots \\ & & & & & & & d_n \end{array}\right] \tag{11}$$

where the centre block is 2×2. It is precisely the entries inside, above and to the right of this centre block that need to be updated; all other entries in T remain unchanged. We recast the update formulas for these entries, which are given in detail in [12], in terms of row and column operations on the matrix T. There are three steps: (i) "backtrack" fraction free Gaussian elimination at row $k-1$; (ii) switch rows/columns $k-1$ and k; (iii) "forwardtrack" fraction free Gaussian elimination at row $k-1$. The following subroutine, which requires $O(n)$ arithmetic operations, gives precisely this procedure.

SwitchRow(A, T, k)
Switch rows $k-1$ and k in A and update T.
switch rows $k-1$ and k in A;
$\mathrm{row}(T, k) \leftarrow (1/T_{k-1\,k-1})(T_{k-2\,k-2}\,\mathrm{row}(T, k) + T_{k-1\,k}\,\mathrm{row}(T, k-1))$;
switch rows $k-1$ and k of T;
switch columns $k-1$ and k of T;
$\mathrm{row}(T, k) \leftarrow (1/T_{k-2\,k-2})(T_{k-1\,k-1}\,\mathrm{row}(T, k) - T_{k-1\,k}\,\mathrm{row}(T, k-1))$;

We now give the L^3 reduction algorithm in terms of our subroutines SubtractRow and SwitchRow. In what follows, we write $[x]$ to mean the nearest integer to a real number x (if nonunique then choose the candidate with smallest magnitude), and as in [12], we write B_i to denote the quantity $\langle b_i^*, b_i^* \rangle$. The quantities B_i and μ_{ij} are recovered from the d_i's and $\bar{\mu}_{ij}$'s, which are entries in the matrix T. Recall that $B_i = d_i/d_{i-1}$ and $\mu_{ij} = \bar{\mu}_{ij}/d_j$ for $1 \leq i < j \leq n$.

```
ReductionL³(A)
# Inplace 2-reduce and size-reduce the n-dimensional integer lattice A.
(1.)[Fraction free Gaussian elimination:]
    T ← FF(AA^t)AA^t;
(2.)[2-reduce and size-reduce:]
    k ← 2;
    do
        SubtractRow(A, T, k, k − 1, [μ_{k k−1}]);
        if B_k < (3/4 − μ²_{k k−1})B_{k−1} then
            # Case 1.
            SwitchRow(A, T, k);
            if k > 2 then k ← k − 1 fi;
        else
            # Case 2.
          ♠ for j from k − 2 by −1 to 1 do SubtractRow(A, T, k, j, [μ_{kj}]) od;
            if k = n then terminate else k ← k + 1 fi;
        fi;
    od;
```

On termination of algorithm $\mathtt{ReductionL}^3$ the output basis $b_1, \ldots, b_n$ (where $b_i = \mathrm{row}(A, i)$) is both 2-reduced and size-reduced, that is, satisfies the properties

(i) $|b_i^* + \mu_{i\,i-1} b_{i-1}^*|^2 < \frac{3}{4}|b_{i-1}^*|^2$ for $2 \le i \le n$.

(ii) $\mu_{ij} \le \frac{1}{2}$ for $1 \le j < i \le n$.

Theorem 11 (A. K. Lenstra, H. W. Lenstra, Jr. and L. Lovász [12]) *For an input basis $b_1, b_2, \ldots, b_n \in \mathbf{Z}^{n'}$ with $n' = O(n)$ and bound $B_o \ge |b_1|^2, |b_2|^2, \ldots, |b_n|^n$, algorithm* $\mathtt{ReductionL}^3$ *requires $O(n^4 \log B_o)$ arithmetic operations with integers bounded in length by $O(n \log B_o)$ bits.* ∎

It is useful at this point to give a sketch of the proof of Theorem 11 — at least the bound on the number of *arithmetic* operations required by step (2.). For convenience, let B be a bound on the magnitudes of initial diagonal entries in T. At the start of the algorithm $|b_i^*|^2 \le |b_i|^2 \le B_o$. The i-th diagonal entry d_i of T is given by $d_i = |b_1^*|^2 \, |b_2^*|^2 \cdots |b_i^*|^2 \le B_o^i$, so so we can take $B = B_o^n$. Consider the two cases of the conditional inside the do loop in step (2.). Case 1 corresponds to a 2-reduction step which costs $O(n)$ operations, and case 2 to a vector size-reduction (line ♠) which costs $O(n^2)$ operations. Because k is always incremented in case 2 (except when $k = n$ which implies termination), the number of occurrences of case 2 will be at most $n-1$ more than the number of occurrences of case 1, so it will be sufficient to bound the number of 2-reduction steps; to this end, the quantity $D = \prod_{i=1}^{n-1} d_i$ is considered. Initially, $\log D = O(n \log B)$ and throughout the algorithm $D \ge 1$ (see [12] for details). D changes only when a 2-reduction step is performed. 2-reduction is performed at index k only if $B_k < (\frac{3}{4} - \mu_{k\,k-1}^2)B_{k-1}$ and, in this case, 2-reduction has the effect of decreasing d_{k-1} by a factor of $< 3/4$ while the other d_i's remain unchanged (again, see [12] for details). This shows that the number of iterations is bounded by $O(\log D) = O(n \log B)$, and hence step (2.) admits a complexity bound of $O(n^3 \log B)$ arithmetic operations. Since $B = O(B_o^n)$, this becomes $O(n^4 \log B_o)$ arithmetic operations as claimed. Note that the dominant step as far as the arithmetic complexity is concerned is line ♠.

We now give a simple modification of step (2.) of algorithm $\mathtt{ReductionL}^3$ that requires only $O(n^2 \log B)$ arithmetic operations. Our idea hinges on the following observation: The vector size-reduction step in line ♠ of case 2 is *not* integral to the correctness of the algorithm. In particular, a close inspection of the proof given in [12] (which we have sketched above) reveals that algorithm $\mathtt{ReductionL}^3$ terminates within $O(n \log B)$ iterations of the loop in step (2.) even if line ♠ is omitted. By omitting line ♠ we get a mathematically correct algorithm for 2-reduction (but not size reduction) that by our complexity analysis above now requires only $O(n^2 \log B)$ arithmetic operations. To complete the basis reduction we include a third step after step (2.) finishes which size reduces the basis — this third step essentially executes line ♠ for $k = 2, 3, \ldots, n$. We remark (without proof) that the modified reduction algorithm just described will return exactly the same reduced basis as algorithm $\mathtt{ReductionL}^3$.

Now that we have reduced by a factor of $O(n)$ the number of arithmetic operations we turn our attention to bounding the bit complexity. A careful analysis in [12] shows that intermediate expressions in algorithm `ReductionL`^{3} (namely the entries in A and T) remain bounded in length by $O(n \log B_o)$ bits. Crucial to the derivation of this size bound is that row k is size-reduced before incrementing k in the loop in step (2.). Certainly, this size bound does not carry over to our modified algorithm described above which omits line ♠. Our method of bounding the length of intermediate integers is completely different. We need the following result.

Lemma 12 *Let $b_1, b_2, \ldots, b_n$ be an input basis to algorithm* `ReductionL`^{3}. *If at any point in the algorithm we have numbers B_o and B such that the basis satisfies*

$$B_i \leq B_o \quad \textit{and} \quad d_i \leq B \quad \textit{for} \quad 1 \leq i \leq n \tag{12}$$

then (12) will remain satisfied for the remainder of the algorithm, and upon termination

$$|b_i|^2 \leq nB_o \quad \textit{for} \quad 1 \leq i \leq n.$$

Proof. See [12]. ∎

Let $A \in \mathbf{Z}^{n \times n'}$ be an input matrix to algorithm `ReductionL`^{3} with $B_o \geq |b_1|^2, |b_2|^2, \ldots, |b_n|^2$. To avoid confusion, we write A_1 to mean the original input matrix (since A is modified inplace during the course of the algorithm). After step (1.), and for the remainder of the algorithm, entries in the work matrix A are never used as intermediate values for subsequent computations. In particular, instead of applying unimodular row operations to A, we could record all row operations in an $n \times n$ unimodular matrix U, initially set to be the identity matrix at the start of step (2.). At the end of the algorithm, the reduced basis can be computed as UA_1. Let $M = 2\lceil (nB_o)^{1/2} \rceil + 1$ so that by Lemma 12 entries in the reduced basis matrix returned by the algorithm will be bounded in magnitude by $(M-1)/2$. Our modular approach hinges on the following simple idea: If $\bar{U}$ is the matrix obtained by reducing modulo M all entries in U, then UA_1 will be equal to the matrix obtained by reducing modulo M (symmetric range) all entries in $\bar{U}A_1$. Equivalently, after step (1.) and for the remainder of the algorithm, we can arbitrarily reduce modulo M entries in the the work matrix A. At the end of the algorithm, the reduced matrix is recovered by reducing modulo M (symmetric range) all entries in A.

So far, we have shown how to keep all intermediate entries in the work matrix A bounded by $M = O(B_o)$. Lemma 12 has already bounded the diagonal entries d_i of T throughout the algorithm. The following lemma shows how we bound the offdiagonal entries $\bar{\mu}_{ij}$ of T.

Lemma 13 *Let T be the matrix of (10), M a positive integer, and i and j indices with $1 \leq i < j \leq n$. There exists a unit upper triangular integer matrix V such that TV is identical to T except with the entry in the i-th row j-th column reduced modulo $d_i d_{i-1} M$. Furthermore, V can be chosen so that $\bar{V}$, the matrix obtained by reducing modulo M the entries in V, will be the identity matrix.*

Proof. The proof rests on some elementary facts from linear algebra. First, recall that $T = FA$ where $F = \mathrm{FF}(AA^t)$ is a lower triangular integer matrix. Since AA^t is a positive definite symmetric matrix, we have

$$TF^t = FAA^tF^t = \begin{bmatrix} d_1 & & & & \\ & d_2 d_1 & & & \\ & & d_3 d_2 & & \\ & & & \ddots & \\ & & & & d_n d_{n-1} \end{bmatrix}.$$

(To see this, note that FAA^tF^t must be symmetric and upper triangular at the same time.) Take V_o be the $n \times n$ strictly upper triangular matrix with column j equal to column i of F^t and all other entries zero. Now, let $q = [T_{ij}/(d_i d_{i-1} M)]$ so that $T_{ij} - q\, d_i d_{i-1} M = \mathrm{mods}(T_{ij}, d_i d_{i-1} M)$. The matrix qMV_o will be strictly upper triangular with $TqMV_o$ equal to the zero matrix except for the entry in the i-th row j-th column which is $qd_i d_{i-1} M$. The matrix $V = -qMV_o + I_n$ has the desired properties. ∎

It follows from Lemma 13 that during the reduction of the lattice basis A we can arbitrarily reduce modulo $d_i d_{i-1} M$ entries to the right of the diagonal in row i of T for $1 \leq i \leq n-1$. For example, consider the entry $\bar{\mu}_{ji}$ in row i column j of T for some choice of i and j with $1 \leq i < j \leq n$. Let V and $\bar{V}$ be the unit upper triangular integer matrices of Lemma 13. Premultiplying A by V^t and reducing entries modulo M will have no effect since $\bar{V}^t$ is the identity matrix. Nonetheless, we are still "required" to update the entries in T corresponding to this row transformation on A — this update proceeds by postmultiplying T by V, but this has precisely the desired effect of reducing modulo $d_i d_{i-1} M$ the entry in row i column j of T. We get the following modified lattice basis reduction algorithm, the correctness of which follows from the previous discussion.

`ModSubtractRow`(A, T, M, k, r, q)
\# Subtract q times row r from row k of A and update T.
`SubtractRow`(A, T, k, r, q);
for i to $k-1$ do $T_{i,k} \leftarrow \text{mods}(T_{i,k}, d_i d_{i-1} M)$ od;
for j to n' do $A_{k,j} \leftarrow \text{mods}(A_{k,j}, M)$ od;

`ModSwitchRow`(A, T, M, k)
\# Switch rows $k-1$ and k in A and update T.
`SwitchRow`(A, T, k);
for i to $k-2$ do $T_{i\,k-1} \leftarrow \text{mods}(T_{i\,k-1}, d_i d_{i-1} M)$ od;
for i to $k-1$ do $T_{i\,k} \leftarrow \text{mods}(T_{i\,k}, d_i d_{i-1} M)$ od;
for j from k to n do $T_{k-1\,j} \leftarrow \text{mods}(T_{k-1\,j}, d_{k-1} d_{k-2} M)$ od;
for j from k+1 to n do $T_{k\,j} \leftarrow \text{mods}(T_{k\,j}, d_k d_{k-1} M)$ od;

`ModReductionL`$^3(A)$
\# Inplace 2-reduce and size-reduce the n-dimensional integer lattice A.
(1.)[Fraction free Gaussian elimination:]
  $T \leftarrow \text{FF}(AA^T)AA^T$;
(2.)[2-reduction:]
  $M \leftarrow 2\lceil (n \max(|b_1|^2, \ldots, |b_n|^2))^{1/2} \rceil + 1$;
  $k \leftarrow 2$;
  do
    `ModSubtractRow`$(A, T, k, k-1, [\mu_{k\,k-1}])$;
    if $B_k < (\frac{3}{4} - \mu_{k\,k-1}^2) B_{k-1}$ then
      `ModSwitchRow`(A, T, k);
      if $k > 2$ then $k \leftarrow k-1$ fi;
    else
      if $k = n$ then break else $k \leftarrow k+1$ fi;
    fi;
  od;
(3.)[Size-reduction:]
  for k from 2 to n do
    for j from $k-1$ by -1 to 1 do `ModSubtractRow`$(A, T, M, k, j, [\mu_{kj}])$ od;
  od;

In analysing the bit complexity of our new algorithm `ModReductionL`^{3} we prefer to follow the approach of Schönhage and assume we have a bound B such that the input basis satisfies

$$|b_i|^2 \leq B, \quad d_i \leq B \ \text{ for } \ 1 \leq i \leq n, \quad \text{and} \quad 2^n \leq B. \tag{13}$$

Note that $B \geq B_o$. Since the initial basis satisfies $d_i \leq B_o^i$, we can take $B = B_o^n$ in case B is not explicitly given. The point of assuming a bound B satisfying (13) is that for some basis reduction problems the input basis admits a certain structure and satisfies $|b_i| = O(B_o)$ rather than the worst case bound $|b_i| = O(B_o^i)$.

Theorem 14 *Algorithm* `ModReductionL`^{3} *is correct. For an input basis* $b_1, b_2, \ldots, b_n \in \mathbf{Z}^{n \times n'}$ *with* $n' = O(n)$ *and bound* B *satisfying (13), the algorithm requires* $O(n^2 \log B)$ *arithmetic operations with integers* *bounded in length by $O(\log B)$ bits. If only a bound $B_o \geq |b_1|^2, |b_2|^2, \ldots, |b_n|^2$ is known, then the running time of the algorithm is $O(n^3 \log B_o)$ arithmetic operations with integers bounded in length by $O(n \log B_o)$ bits.*

Proof. The correctness of the algorithm follows from the earlier discussion. To determine the complexity of computing $T = \mathrm{FF}(AA^t)AA^t$ in step (1.), we need to bound the magnitudes of entries in $C = AA^t$ and T. For C, we have $C_{ij} = \langle b_i, b_j \rangle \leq \max(\langle b_i, b_i \rangle, \langle b_j, b_j \rangle) \leq B$. By assumption, diagonal entries of T (the d_i's) are bounded by B. For $1 \leq j < i \leq n$ we have

$$T_{ji} = d_j \mu_{ij} = d_j \langle b_i, b_j^* \rangle / |b_j^*|^2 \leq d_j B \leq B^2.$$

Thus, T can be computed within $O(n^3 (\log B)^{1+\epsilon})$ bit operations using the algorithm of Theorem 9. By assumption $B \geq 2^n$ so that $O(n^3(\log B)^{1+\epsilon}) = O(n^2 (\log B)^{2+\epsilon})$. We now bound the complexity of steps (2.) and (3.). We have already shown that the number of arithmetic operations required for step (2.) is bounded by $O(n^2 \log B)$. Step (3.) requires $O(n^2)$ calls to subroutine **`ModSubtractRow`** and so will require $O(n^3)$ arithmetic operations. By assumption $B \geq 2^n$ so that $O(n^3) = O(n^2 \log B)$. Thus, it will suffice to show that all intermediate entries of T and A during step (2.) and (3.) remain bounded by $O(\log B)$ bits in length. Entries in A are kept reduced modulo M where $M \leq 2B_o^{1/2} + 1 \leq 3B$, and entries in row i of T are kept reduced modulo $d_i d_{i-1} M \leq d_i d_{i-1}(3B)$, which by Lemma 12 will remain bounded by $3B^3$. The complexity result in terms of B_o follows by noting that the choice $B = B_o^n$ satisfies (13). ■

4.1 Practical Considerations

For clarity, algorithm **`ModReductionL`**^{3} was presented in a form which differed from the orignial L^3-reduction algorithm as little as possible. In particular, **`ModReductionL`**^{3} will return *exactly* the same reduced basis as the original L^3-reduction algorithm. In practice, it is more efficient to implement the algorithm differently. We mention here three heuristics that can have significant impact on the running time.

Heuristic 1: The goal of reduction is to transform the basis so as to satisfy $|\mu_{ij}| \leq 1/2$ for $1 \leq j < i \leq n$ and

$$\tfrac{3}{4}|b_{i-1}^*|^2 \leq |b_i^* + \mu_{i\,i-1} b_{i-1}^*|^2 \quad \text{for} \quad 1 < i \leq n.$$

Since $\mu_{i\,i-1}$ may be as large as $1/2$, the best that can be concluded from the above conditions is that $|b_{i-1}^*|^2 \leq 2|b_i^*|^2$, which is equivalent to $d_{i-1}^2 \leq 2 d_i d_{i-2}$. Thus, we can replace the conditional $B_k < (\frac{3}{4} - \mu_{k\,k-1}^2) B_{k-1}$ in the loop in step (2.) with the simpler check $d_{k-1}^2 > 2 d_k d_{k-2}$. This change also dispenses with the need to call **`ModSubtractRow`**$(A, T, k, k-1, [\mu_{k\,k-1}])$ at the start of the loop.

Heuristic 2: The original L^3 algorithm (and **`ModReductionL`**^{3}) always picks the smallest possible row index k at which to perform a 2-reduction. In the L^3 algorithm this choice was required to properly bound the size of intermediate integers but in our case we are at liberty to take a more global approach. The rate of progress of 2-reduction is related to the quantity $d_1 d_2 \cdots d_{n-1}$. If $d_{k-1}^2 > 2 d_k d_{k-2}$ for some $2 \leq k \leq n$, then a 2-reduction step performed at row k will reduce d_{k-1} by reduction factor of $\alpha_k < \frac{3}{4}$ while leaving the other d_i's unchanged. Since 2-reduction at row k replaces b_{k-1}^* with $b_k^* + \mu_{k\,k-1} b_{k-1}^*$ (where $\mu_{k\,k-1}$ has first been reduced to satisfy $\mu_{k\,k-1} \leq 1/2$), the actual reduction factor α_k is given by

$$\begin{aligned}
\alpha_k &= |b_k^* + (\mu_{k\,k-1} - [\mu_{k\,k-1}]) b_{k-1}^*|^2 / |b_{k-1}^*|^2 \\
&= (B_k + (\mu_{k\,k-1} - [\mu_{k\,k-1}])^2 B_{k-1}) / B_{k-1} \\
&= B_k / B_{k-1} + (\mu_{k\,k-1} - [\mu_{k\,k-1}])^2 \qquad (14) \\
&\leq B_k / B_{k-1} + \tfrac{1}{4} \qquad (15)
\end{aligned}$$

The idea is to apply the greedy algorithm: at each step through the loop, choose the row index k which guarantees the most progress. The reduction factors α_k for $2 \leq k$ can be computed exactly using (14) or approximated using (15). In many cases, we have observed this greedy approach to approximately halve the number of 2-reduction steps required to 2-reduced the basis.

Heuristic 3: By Lemma 12, at any point in the algorithm we have the bound $\sqrt{n} \max(|b_1^*|^2, \ldots, |b_n^*|^2)^{1/2}$ on the magnitudes of entries in the final basis matrix. In practice, we have observed that this bound decreases rapidly after only a few iterations of the loop in step (2.). This leads to the idea of recomputing the bound each time through the loop and resetting the modulus M accordingly. Of course, we must ensure that the new M is an integer factor of the old M. This condition is easily met by choosing M to be a power of 10, for example.

For completeness, we give a modification of step (2.) which incorporates all three of the heuristics discussed in this section.

(2.)[2-reduction with heuristics:]
    do
        $e \leftarrow \lceil \frac{1}{2}\log_{10}(n \max(|b_1^*|^2, \ldots, |b_n^*|^2)) \rceil + 1$;
        $M \leftarrow 10^e$;
        $k \leftarrow 2$;
        for i from 3 to n do
            if $B_i/B_{i-1} < B_k/B_{k-1}$ then $k \leftarrow i$ fi;
        od;
        if $B_k/B_{k-1} \geq \frac{1}{2}$ then break fi;
        `ModSubtractRow`$(A, T, k, k-1, [\mu_{k\,k-1}])$;
        `ModSwitchRow`(A, T, k);
    od;

5 Asymptotically Fast Semi-Reduction

In this section we combine our modification `ModReductionL`^{3} of Section 4 with Schönhage's [14] modification of the L^3-reduction algorithm. The result is an asymptotically fast algorithm for semi-reduction. A bases $b_1, b_2, \ldots, b_n$ is called *semi-reduced* if

$$|b_r^*|^2 \leq 2^{n+s-r}|b_s^*|^2 \quad \text{for} \quad 1 \leq r < s \leq n. \tag{16}$$

Schönhage's algorithm for semi-reduction requires $O(n^2(\log B)^{2+\epsilon})$ bit operations where B is a bound satisfying (13). In Subsection 5.2 we present an asymptotically fast three step algorithm for semi-reduction:

(1.) Compute the Gram-Schmidt orthogonalization of the input basis.

(2.) Semi-reduce the basis.

(3.) Size-reduce the basis.

We consider steps (1.) and (3.) to be pre- and post-conditioning steps since we prove the running time for these steps is bounded by $O(n^\theta(\log B)^{1+\epsilon})$ bit operations — we consider this to be near optimal since this is about the same time required to compute the determinant of the input lattice. For step (2.) we prove a running time of $O(n^{1+1/(5-\theta)}(\log B)^{2+\epsilon})$ bit operations. If we take $\theta = 2.37$ than this becomes $O(n^{2.381}(\log B)^{2+\epsilon})$ bit operations and this bounds the cost of steps (1.) and (2.) as well. We follow this with an analysis of our semi-reduction algorithm which assumes standard integer and matrix multiplication. Under these assumptions we are able to prove a running time of $O(n^{1+1/3}(\log B)^3(\log\log B))$ bit operations for steps (1.), (2.) and (3.).

Our algorithm requires a number of results and subroutines for performing operations on the upper triangular matrix T of d_i's and $\bar{\mu}_{ij}$'s; We present these separately in Subsection 5.1. Presenting the results of Subsection 5.1 separately has a dual purpose. First, the presentation of the semi-reduction algorithm is greatly simplified. Secondly, the the semantics of the entries in T (namely the d_i's and $\bar{\mu}_{ij}$'s) are an orthogonal issue.

5.1 Subroutines for Positive Definite Symmetric Matrices

In this section we give a number of asymptotically fast algorithms for effecting certain transformations on an $n \times n$ upper triangular integral matrix

$$T = \mathrm{FF}(AA^t)AA^t = \begin{bmatrix} t_{11} & t_{12} & t_{13} & & t_{1n} \\ & t_{22} & t_{23} & \cdots & t_{2n} \\ & & t_{33} & & t_{3n} \\ & & & \ddots & \vdots \\ & & & & t_{nn} \end{bmatrix} \quad \text{where} \quad A \in \mathbf{Z}^{n \times n'}, \ \mathrm{rank}(A) = n \tag{17}$$

Note that AA^t is both symmetric and positive definite, that is, with principal i'th minor positive for $1 \leq i \leq n$. By Lemma 4, the i-th diagonal entry of T will be the i-th principal minor of A.

For the statment of the following theorem we need some definitions. For a positive integer M, an offdiagonal entry t_{ij} in the matrix T is said to be *size-reduced* modulo M if and only if $t_{ij} \bmod (t_{ii}t_{i-1\,i-1}M) \leq t_{ii}/2$. The entire matrix T is said to be size-reduced modulo M if and only if all offdiagonal entries are size-reduced modulo M. It will be convenient to consider a matrix of the form T in (17) to have a zeroeth diagonal entry, which, unless otherwise stated will be assumed to be 1.

Theorem 15 *There exists a deterministic algorithm that takes as input an $n \times n$ upper triangular matrix T as in (17) together with a positive integer M, and produces as output an $n \times n$ unit upper triangular integer matrix V such that TV will be size-reduced modulo M. If diagonal entries in T are bounded in magnitude by B, offdiagonal entries by B^2M, and $B \geq n, M$, then entries in V will be bounded in magnitude by BM and the cost of the algorithm is $O(n^\theta(\log B)(\log\log B)^\epsilon + n^2(\log n)(\log B)^{1+\epsilon})$ or the simpler bound of $O(n^\theta(\log B)^{1+\epsilon})$ bit operations.*

Proof. By working with the matrix $\mathrm{diag}(T, I_p)$, were p is at most $n-1$, we may assume without loss of generality that n is a power of two. We prove the theorem for a special type of input matrix which can be written in block form as

$$T = \left[\begin{array}{ccc|ccc} t_{1\,1} & \cdots & t_{1\,k} & t_{1\,k-1} & \cdots & t_{1\,n} \\ & \ddots & \vdots & \vdots & & \vdots \\ & & t_{j\,j} & t_{p\,p+1} & \cdots & t_{p\,n} \\ \hline & & & 1 & & \\ & & & & \ddots & \\ & & & & & 1 \end{array}\right] \tag{18}$$

where j and n are powers of 2 and where the trailing $(n-j) \times (n-j)$ submatrix is the identity. The general case will follow by taking $j = n$. We first summarize the algorithm before making the argument precise: (i) Compute a V_1 such that rows $j/2+1, j/2+2, \ldots, j$ of TV_1 are size-reduced modulo M; (ii) Compute V_2 such that rows $1, 2, \ldots, j/2$ of TV_1V_2 are size-reduced modulo M; (iii) Paste V_1 and V_2 together to get a V such that TV_1V_2 is size-reduced modulo M. We get the following diagram

$$\begin{array}{c} T \\ \left[\begin{array}{ccccccccc} * & \cdots & * & * & \cdots & * & * & \cdots & * \\ & \ddots & \vdots & \vdots & & \vdots & \vdots & & \vdots \\ & & * & * & \cdots & * & * & \cdots & * \\ \hline & & & * & \cdots & * & * & \cdots & * \\ & & & & \ddots & \vdots & \vdots & & \vdots \\ & & & & & * & * & \cdots & * \\ \hline & & & & & & 1 & & \\ & & & & & & & \ddots & \\ & & & & & & & & 1 \end{array}\right] \end{array} \begin{array}{c} V_1 \\ \left[\begin{array}{ccccccccc} 1 & & & & & & & & \\ & \ddots & & & & & & & \\ & & 1 & & & & & & \\ \hline & & & 1 & \cdots & * & * & \cdots & * \\ & & & & \ddots & \vdots & \vdots & & \vdots \\ & & & & & 1 & * & \cdots & * \\ \hline & & & & & & 1 & & \\ & & & & & & & \ddots & \\ & & & & & & & & 1 \end{array}\right] \end{array} \begin{array}{c} V_2 \\ \left[\begin{array}{ccccccccc} 1 & \cdots & * & * & \cdots & * & * & \cdots & * \\ & \ddots & \vdots & \vdots & & \vdots & \vdots & & \vdots \\ & & 1 & * & \cdots & * & * & \cdots & * \\ \hline & & & 1 & & & & & \\ & & & & \ddots & & & & \\ & & & & & 1 & & & \\ \hline & & & & & & 1 & & \\ & & & & & & & \ddots & \\ & & & & & & & & 1 \end{array}\right] \end{array}$$

where each of the first two blocks in each matrix is comprised of $j/2$ rows. For k and n powers of 2, let $C(n, k)$ be the number of bit operations required to compute, for an input matrix which can be written as in

(18) with $j = k$, a matrix V which satisfies the requirements of the theorem and has trailing $(n-j)\times(n-j)$ trailing submatrix the identity. We claim that

$$C(n,k) \le 2C(n,k/2) + cnk^{\theta-1}(\log\beta)(\log\log\beta)^{\epsilon} + cnk(\log\beta)^{1+\epsilon} \tag{19}$$

for some absolute constant c and where $\beta = 2nB^3m^2 + 1$. To prove (19) let T be an $n\times n$ input matrix which can be written as in (18) with $j = k$. We give a conquer and divide algorithm that computes a V satisfying the requirements theorem with input T that requires two recursive calls on matrices which can be written as in (18) with $j = k/2$ and additional work at most $O(nk^{\theta-1}(\log\beta)(\log\log\beta)^{\epsilon} + nk(\log\beta)^{1+\epsilon})$ bit operations. Let T_1 be the trailing $(n-k/2)\times(n-k/2)$ submatrix of T so that the block diagonal matrix $\mathrm{diag}(T_1, I_{k/2})$ can be written as in (18) with $j = k/2$ and zeroeth diagonal entry now $d_{k/2-1}$. At a cost of $C(n,k/2)$ bit operations, recursively compute a V_1' satisfying the requirements of the theorem with input matrix $\mathrm{diag}(T_1', I_{k/2})$, and set V_1 to be the $n\times n$ block diagonal matrix with principal $(k/2)\times(k/2)$ submatrix the identity and trailing $(n-k/2)\times(n-k/2)$ trailing submatrix the principal $(n-k/2)\times(n-k/2)$ submatrix of V_1'. By assumption, entries in T and V_1 will be bounded by B^2M and BM respectively so that entries in TV_1 will be bounded by nB^3M^2. Because of the block structure of T and V_1, we can compute the matrix product TV_1 at a cost of $O(nk^{\theta-1}(\log\beta)(\log\log\beta)^{1+\epsilon} + nk(\log\beta)^{1+\epsilon})$ bit operations. At a cost of $O(nk(\log\beta)^{1+\epsilon})$ bit operations, compute that matrix $\bar{T}$ from TV_1 by reducing modulo $t_{ii}t_{i-1\,i-1}M$ entries in row i of TV_1 for $1\le i\le k$. At this point, the trailing $(n-k/2)\times(n-k/2)$ trailing submatrix of $\bar{T}$ has the correct form and all offdiagonal entries in $\bar{T}$ are bounded in magnitude by B^2M. Let T_2 be the matrix obtained from $\bar{T}$ by replacing the trailing $(n-k/2)\times(n-k/2)$ submatrix of $\bar{T}$ with the identity matrix. At a cost of $C(n,k/2)$ bit operations compute a matrix V_2 which satisfies the requirements of the theorem with input matrix T_2. Set V to be the matrix with first $k/2$ rows those of T_1, rows $k/2+1,\ldots,k$ those of T_2, and trailing $(n-k)\times(n-k)$ submatrix the identity. Then V satisfies the requirements of the theorem for input matrix T. This proves (19). Iterating (19) with $k = n$ yields

$$\begin{aligned}
C(n,n) &\le 2C(n,n/2) + cnn^{\theta-1}(\log\beta)(\log\log\beta)^{\epsilon} + nn(\log\beta)^{1+\epsilon}\\
&\le 4C(n,n/2) + cn(\log\beta)(\log\log\beta)^{\epsilon}(n^{\theta-1} + 2(n/2)^{\theta-1}) + n(\log\beta)^{1+\epsilon}(n + 2(n/2))\\
&\ \ \vdots\\
&\le nC(n,1) + cn^{\theta}(\log\beta)(\log\log\beta)^{\epsilon}\sum_{i=0}^{\log_2 n}(2/2^{\theta-1})^i + n(\log\beta)^{1+\epsilon}\sum_{i=0}^{\log_2 n} n\\
&\ll nC(n,1) + n^{\theta}(\log\beta)(\log\log\beta)^{\epsilon} + n^2(\log n)(\log\beta)^{1+\epsilon}
\end{aligned}$$

For an input matrix T which can be written as in (18) with $j = 1$, the matrix V, set to be the $n\times n$ identity except with the entry in the first row j-th column equal to $-[t_{1j}/(t_{1\,1}t_{0\,0}M)]$ for $2\le j\le n$, will satisfy the requirements of the theorem. This shows $C(n,1)\le n(\log\beta)^{1+\epsilon}$ bit operations. The result now follows by noting that $\beta = O(B)$ since $B\ge n, M$. ■

In Subsection 5.2 we will need to "backtrack" fraction free Gaussian elimination on the trailing $m\times m$ submatrix of the matrix T of (17) for some $1\le m\le n$. To make this idea precise, let $C = AA^t$ so that $T = \mathrm{FF}(C)C$ and, for $0\le i\le n$ let $\mathrm{TT}(C,i)$ denote the matrix obtained by applying fraction free Gaussian elimination on C up to and including column i. Note that $\mathrm{TT}(C,0) = C$ and $\mathrm{TT}(C,n-1) = \mathrm{TT}(C,n) = \mathrm{FF}(C)C$. To see what $\mathrm{TT}(C,n-m)$ looks like, write C, T and $F = \mathrm{FF}(C)$ using a block decomposition as

$$\overset{F}{\left[\begin{array}{c|c} F_1 & \\ \hline F_2 & F_4\end{array}\right]}\overset{C}{\left[\begin{array}{c|c} C_1 & C_3 \\ \hline C_2 & C_4\end{array}\right]} = \overset{T}{\left[\begin{array}{c|c} T_1 & T_3 \\ \hline & T_4\end{array}\right]} \tag{20}$$

where F_4, C_4 and T_4 are $m\times m$. Then

$$\mathrm{TT}(C,n-m) = \left[\begin{array}{c|c} T_1 & T_3 \\ \hline & -C_2C_1^{\mathrm{adj}}C_3 + dC_4\end{array}\right] \quad \text{with } \ d = t_{n-m\,n-m}.$$

So, what we want is an $m \times m$ matrix U such that

$$\left[\begin{array}{c|c} I_{n-m} & \\ \hline & U \end{array}\right] \overset{T}{\left[\begin{array}{c|c} T_1 & T_3 \\ \hline & T_4 \end{array}\right]} = \overset{\mathrm{TT}(C, n-m)}{\left[\begin{array}{c|c} T_1 & T_3 \\ \hline & -C_2 C_1^{\mathrm{adj}} C_3 + dC_4 \end{array}\right]}. \tag{21}$$

For a nonsymmetric input matrix (but with $(n-m)$-th principal minor nonsingular) we would have to compute U as

$$U = (-C_2 C_1^{\mathrm{adj}} C_3 + dC_4) T_4^{-1} \tag{22}$$

but the following lemma shows that this is much too expensive in the special case where the matrix C is symmetric. Recall that we assume our input matrix T to have a zeroeth diagonal entry with the value 1.

Lemma 16 *For an index m with $0 \leq m \leq n$ let the matrix $C = AA^t$ and $T = \mathrm{FF}(C)C$ of (17) be written as in (20) and let D be the $m \times m$ diagonal matrix with i-th diagonal entry $t_{n-m+i,n-m+i} t_{n-m-1+i,n-m-1+i}$. Then the matrix $U = dT_4^t D^{-1}$ will satisfy $UT_4 = -C_2 C_1^{\mathrm{adj}} C_3 + dC_4$ where $d = t_{n-m,n-m}$.*

Proof. From (22) we have

$$(T_4^{-1})^t U = (T_4^{-1})^t (-C_2 C_1^{\mathrm{adj}} C_3 + dC_4) T_4^{-1}. \tag{23}$$

Note that we will be finished if we show that the matrix on the right hand side of (23) is equal to dD^{-1}. Since $C_2 = C_3^t$ and C_4 is symmetric we must have $-C_2 C_1^{\mathrm{adj}} C_3 + dC_4$ symmetric and hence also the matrix on the right hand side of (23) is symmetric. Now note that U is lower triangular since

$$\left(\left[\begin{array}{c|c} I_{n-m} & \\ \hline & U \end{array}\right] \left[\begin{array}{c|c} I_{n-m} & \\ \hline F_2 & F_4 \end{array}\right] \right) \left[\begin{array}{c|c} C_1 & C_3 \\ \hline C_2 & C_4 \end{array}\right] = \left[\begin{array}{c|c} C_1 & C_3 \\ \hline O & -C_2 C_1^{\mathrm{adj}} C_3 + dC_4 \end{array}\right]$$

together with

$$\left[\begin{array}{c|c} I_{n-m} & \\ \hline -C_2 C_1^{\mathrm{adj}} & dI_m \end{array}\right] \left[\begin{array}{c|c} C_1 & C_3 \\ \hline C_2 & C_4 \end{array}\right] = \left[\begin{array}{c|c} C_1 & C_3 \\ \hline & -C_2 C_1^{\mathrm{adj}} C_3 + dC_4 \end{array}\right] \tag{24}$$

implies $UF_4 = dI_m$ so that we must have $U = dF_4^{-1}$ where F_4^{-1} is lower triangular. But then the matrix on the left hand side of (23), the product of two lower triangular matrices, must be lower triangular. But then the matrix on the right hand side of (23) is symmetric and lower triangular whence diagonal. The result now follows by noting that the principal minors of the matrix $d(T_4^{-1})^t F_4^{-1}$ on the left hand side (and hence also right hand side) of (23) are the same as those of dD^{-1}, whence $(T_4^{-1})^t U = dD^{-1}$. ■

We will also need to "forwardtrack" fraction free Gaussian elimination on the trailing $m \times m$ submatrix of T for some $0 \leq m \leq n$. In other words, we want to compute an $m \times m$ matrix G such that

$$\left[\begin{array}{c|c} I_{n-m} & \\ \hline & G \end{array}\right] \overset{\mathrm{TT}(C, n-m)}{\left[\begin{array}{c|c} T_1 & T_2 \\ \hline & -C_2 C_1^{\mathrm{adj}} C_3 + dC_4 \end{array}\right]} = \overset{T}{\left[\begin{array}{c|c} T_1 & T_3 \\ \hline & T_4 \end{array}\right]}$$

where C and T are written as in (20). Comparing the above with (21) shows that G is given by U^{-1} where U is the $m \times m$ backtrack matrix for the trailing $m \times m$ submatrix of T. We get the following.

Corollary 17 *For an index m with $0 \leq m \leq n$ let the matrix $C = AA^t$ and $T = \mathrm{FF}(C)C$ of (17) be written as in (20) and let D be the $m \times m$ diagonal matrix with i-th diagonal entry $t_{n-m+i,n-m+i} t_{n-m-1+i,n-m-1+i}$. Then the matrix $G = (1/d)D(T_4^{-1})^t$ will satisfy $G(-C_2 C_1^{\mathrm{adj}} C_3 + dC_4) = T_4$ where $d = t_{n-m,n-m}$.*

5.2 The Semi-Reduction Algorithm

Recall the essential idea of step of the L^3-reduction algorithm. We have the basis matrix A being reduced together with the corresponding upper triangular matrix $T = \mathrm{FF}(AA^t)AA^t$ which contains the d_i's and $\bar{\mu}_{ij}$'s. The algorithm determines from the d_i's and $\bar{\mu}_{ij}$'s exactly what unimodular row operation should be performed on the basis matrix A; this row operation is then performed on A and the quantities in T are updated to coincide with the new A. This process repeats until the basis is 2-reduced. Schönhage's idea is to apply L^3-reduction to certain *blocks* of rows of the matrix A being reduced. For example, in algorithm `ModReductionL`^{3} we always applied 2-reduction to a block of 2 rows, namely rows b_{k-1}, b_k of A for some k with $2 \le k \le n$. Schönhage's semi-reduction algorithm applies 2-reduction to a block of $m+1$ rows b_{p-m}, b_{p-m+1}, b_p of A for some choice of p and m with $1 \le m < p \le n$. (How this choice is made will be discussed later.) We can best see with a picture exactly what happens during a block-reduction. Write T using a block decomposition as

$$T = \left[\begin{array}{ccc|ccc|ccc} d_1 & \cdots & \bar{\mu}_{p-m-1\,1} & \bar{\mu}_{p-m\,1} & \cdots & \bar{\mu}_{p\,1} & \bar{\mu}_{p+1\,1} & \cdots & \bar{\mu}_{n\,1} \\ & \ddots & \vdots & \vdots & & \vdots & \vdots & & \vdots \\ & & d_{p-m-1} & \bar{\mu}_{p-m\,p-m-1} & \cdots & \bar{\mu}_{p\,p-m-1} & \bar{\mu}_{p+1\,p-m-1} & \cdots & \bar{\mu}_{n\,p-m-1} \\ \hline & & & d_{p-m} & \cdots & \bar{\mu}_{p,p-m} & \bar{\mu}_{p+1\,p-m} & \cdots & \bar{\mu}_{n\,p-m} \\ & & & & \ddots & \vdots & \vdots & & \vdots \\ & & & & & d_p & \bar{\mu}_{p+1\,p} & \cdots & \bar{\mu}_{n\,p} \\ \hline & & & & & & d_{p+1} & \cdots & \bar{\mu}_{n\,p+1} \\ & & & & & & & \ddots & \vdots \\ & & & & & & & & d_n \end{array}\right] \tag{25}$$

A *block-reduction* at row p of size m is a unimodular transformation of rows $p-m, p-m+1, \ldots, p$ of A such that the matrix T, after being updated corresponding to the transformation on A, will have the centre block of (25) both 2-reduced and size reduced. In particular, T will satisfy

$$|b_i^* + \mu_{i\,i-1} b_{i-1}^*|^2 < \tfrac{3}{4}|b_{i-1}^*|^2 \quad \text{for} \quad p-m+1 \le i \le p \tag{26}$$

and

$$\mu_{ij} \le \tfrac{1}{2} \quad \text{for} \quad p-m \le j < i \le p. \tag{27}$$

Following Schönhage, the total cost in bit operations required to compute a block-reduction can be partitioned into two parts. First, there is the cost of 2-reducing the $(p-m+1) \times (p-m+1)$ centre block of the matrix in (25). Secondly, there is the overhead cost of updating the entries above and to the right of the centre block in T and in rows $p-m, p-m+1, \ldots, p$ of A. Schönhage has shown that a block reduction at row p of size m can be accomplished using $O(tm^2(\log B)^{1+\epsilon})$ bit operations where t is the number of 2-reduction steps, plus an overhead cost of $O(n^2 m(\log B)^{1+\epsilon})$ bit operations (Proposition 3.2 in [14]). In our case, we get the following.

Theorem 18 *Let T be the matrix of (25) and m and p indices with $1 \le m < p \le n$. If diagonal entries of T are bounded by B, offdiagonal entries are bounded by B^2M where $M \le B$, and entries in the basis matrix A are being kept reduced modulo M, then a block-reduction at row p of size m can be performed at cost of $O(tm(\log B)^{1+\epsilon})$ bit operations where t is the number of 2-reduction steps, plus an overhead cost of $O(nm^{\theta-1}(\log B)(\log\log B)^{\epsilon} + (nm + m^2(\log m))(\log B)^{1+\epsilon})$ bit operations.*

Proof. We divide our algorithm for block-reduction into three parts. For each part we prove a bit complexity bounded by that claimed by the theorem. Throughout, the submatrix of T comprised of rows and columns $p-m, p-m+1, \ldots, p$ will be referred to as the "centre block of T".

Part I: The first step is to size-reduce the entries in the centre block of T, that is, we want to satisfy (27). To do this, we compute an $(m+1) \times (m+1)$ upper triangular integer matrix V such that the matrix

$$T \left[\begin{array}{c|c|c} I_{p-m-1} & & \\ \hline & V & \\ \hline & & I_{n-p} \end{array}\right] \tag{28}$$

will have entries in the centre block size-reduced modulo M. By Theorem 15, we can compute a suitable V with $||V|| \le BM$ within $O(m^\theta(\log B)(\log\log)^\epsilon + m^2(\log m)(\log B)^{1+\epsilon})$ bit operations. Then we have $||TV|| < n||T||\,||V|| \le nB^3M^2 = O(B)$ so that we can compute the matrix product of (28) at a cost of $O(nm^{\theta-1}(\log B)(\log\log B)^\epsilon + nm(\log)^{1+\epsilon})$ bit operations. Replace T with the new matrix obtained from the matrix in (28) by reducing modulo $d_i d_{i-1} M$ entries in row i column j for $1 \le i \le p$ and $p-m \le j \le p$. This costs $O(nm(\log B)^{1+\epsilon})$ bit operations. Finally, at a cost of $O(nm^{\theta-1}(\log B)(\log\log B)^{1+\epsilon} + nm(\log)^{1+\epsilon})$ bit operations, replace A with the matrix obtained from

$$\left[\begin{array}{c|c|c} I_{p-m-1} & & \\ \hline & V^t & \\ \hline & & I_{n-p} \end{array}\right] A$$

by reducing modulo M entries in row j for $p-m \le j \le p$. Up to this point we have used $O(nm(\log B)^{1+\epsilon} + nm^{\theta-1}(\log B)(\log\log)^\epsilon)$ bit operations to effect a transformation of the work matrices A and T so that T will satisfy (27).

Part II: The next step is to reduce the centre block

$$T_1 = \begin{bmatrix} d_{p-m} & \bar\mu_{p-m+1\,p-m} & \cdots & \bar\mu_{p\,p-m} \\ & d_{p-m+1} & & \vdots \\ & & \ddots & \bar\mu_{p\,p-1} \\ & & & d_p \end{bmatrix}$$

of T. For clarity, let T_2 be a copy of T_1 so that T_1 will always refer to the size-reduced (but not 2-reduced) centre block of T. In other words, T_1 denotes the centre block of T after part I has completed. The reduction of T_2 proceeds in two steps:

`ModBlockReductionL`$^3(T_2)$

(1.) 2-reduce T_2 using a variation of step (2.) of algorithm `ModReductionL`^{3}, which, instead of applying unimodular row operations to the appropriate block of $m+1$ rows of A, records row operations in a matrix U, initially set to be the $(m+1)\times(m+1)$ identity.

(2.) Size reduce T_2 by post-multiplying by the $(m+1)\times(m+1)$ unit upper triangular V computed using the asymptotically fast size-reduction algorithm of Theorem 15. The transformation matrix U is updated by pre-multiplying by V^t.

The following lemma from Schönhage [14] shows that during the reduction of T_2 we can record all row transformation in $U \bmod \beta$ (symmetric range) where $\beta = 2(m+1)^2(3/2)^m B + 1 = O(B)$. We remark that in order to apply the following result the input matrix T_2 to `ModBlockReductionL`^{3} must be size-reduced on input and on output. In particular, the whole purpose of part I was to satisfy this condition.

Lemma 19 (Lemma 3.1 in [14]) *Upon termination of* `ModBlockReductionL`^{3}*, entries in U will be bounded in magnitude by $(m+1)^2(3/2)^m B$.* ■

Note that during the course of `ModBlockReductionL`^{3} we use $\beta = 2(m+1)^2(3/2)^m B + 1$ in place of the modulus M. Then, entries of T_2 during step (1.) of `ModBlockReductionL`^{3} are kept bounded by $B^2\beta$, and the V computed in step (2.) will satisfy $||V|| \le B\beta$. Since $\beta = O(B)$, the total cost of `ModBlockReductionL`^{3} is bounded by $O(t(\log B)^{1+\epsilon})$ bit operations for step (1.) and $O(m^\theta(\log B)(\log\log B)^\epsilon + m^2(\log m)(\log\log B)^{1+\epsilon})$ bit operations for step (2.).

Part III: We already have the correct form of the centre block T_1 of T, namely T_2. The last step is to update the entries above and to the right of the centre block in T and in rows $p-m, p-m+1, \ldots, p$ of A. For A, this is accomplished by pre-multiplying the centre block comprised of rows $p-m, p-m+1, \ldots, p$ of A by U followed by reduction modulo M. For the entries above the centre block in T, this is accomplished by post-multiplying this $(p-m-1)\times(m+1)$ block by U^t and reducing modulo $d_i d_{i-1} M$ entries in row i column j for $1 \le i \le p-m-1$ and $p-m \le j \le p$. By the bounds established on $||U||$, we can accomplish this update of A and first $p-m-1$ rows of T within $O(nm^{\theta-1}(\log B)(\log\log b)^{1+\epsilon}+nm(\log B)^{1+\epsilon})$ bit operations. Finally, we give the procedure to update the entries to the right of the centre block in T. For convenience, let E denote this $(m+1)\times(n-p)$ block. Also, let $d = d_{p-m-1}$ and let D_1 be the $(m+1)\times(m+1)$ diagonal matrix with i-th diagonal entry $d_{p-m-1+i}d_{p-m-2+i}$ for $1\le i\le m+1$, and D_2 the $(m+1)\times(m+1)$ diagonal matrix with $(D_2)_{11} = (T_2)_{11}d_{p-m-1}$ and $(D_2)_{ii} = (T_2)_{ii}(T_2)_{i-1\,i-1}$ for $2\le i\le m+1$. (Note that the diagonal entries of T_2 will give the new d_i's for $p-m\le i\le p$.) Then we need to perform the following three transformation on E: (i) bactrack fraction free Gaussian by pre-multiplying by $dT_1^tD_1^{-1}$ (see Lemma 16); (ii) pre-multiply by the basis transformation matrix U; (iii) forwardtrack fraction free Gaussian elimination by pre-multiplying by $(1/d)D_2(T_2^{-1})^t$ (see Corollary 17). Thus, the transformed block $\bar{E}$ is given by

$$\bar{E} = QE \quad \text{where} \quad Q = D_2(T_2^{-1})^tUT_1^tD_1^{-1} \tag{29}$$

We need to establish a bound on the magnitudes of integers in $\bar{E}$. To bound $||T_2^{-1}||$, let S be the $(m+1)\times(m+1)$ diagonal matrix with i-th diagonal entry $(T_2)_{ii}$ for $1\le i\le m+1$ so that $S^{-1}T_2$ is unit upper triangular with all offdiagonal entries $\le 1/2$. (Recall that T_2 is size-reduced.) In particular, entries in $(S^{-1}T_2)^{-1}$ are $m\times m$ minors of $(S^{-1}T_2)^{-1}$ which, by Hadamard's inequality, will be bounded in magnitude by $m^{m/2}$. It follows that entries in $T_2^{-1} = (S^{-1}T_2)^{-1}S$ will be bounded by $Bm^{m/2}$. We get

$$\begin{aligned}||\bar{E}|| = ||QE|| &\le (m+1)||Q||\,||E||\\ &\le (m+1)^3||D_2||\,||T_2^{-1}||\,||U||\,||T_1^t||B^2M\\ &\le (m+1)^3B\,Bm^{m/2}\,(m+1)^2(3/2)^mB\,B\,B^2M\\ &\le (m+1)^5m^{m/2}(3/2)^mB^6M\\ &\le O(B)\end{aligned}$$

where the last inequality follows from the assumptions on B, namely $B\ge 2^n, M$. We now compute the matrix $\bar{E}$ using the homomorphic imaging scheme of Theorem 2. We need to choose a basis of s primes $(p_i)_{1\le i\le s}$ with $\prod_{1\le i\le s}p_i\ge\beta$ where $\beta = 2(m+1)^5m^{m/2}(3/2)mB^6M+1$. Care needs to be taken so as not to choose primes which divide diagonal entries of T_2 or D_1 since the formula for Q involves the inverse of these matrices. The product of all diagonal entries in T_2 and D_1 will be bounded by $B^{2(m+1)}$ so that by Lemma 1 we can choose our primes to be bounded in length by $l = 6+\log\log(\beta B^{2(m+1)}) = O(\log\log B)$ bits in length. Since entries in D_1, D_2, T_1, T_2 and U are bounded by β as well, we can compute $\bar{E}$ within $O(nm^{\theta-1}(\log B)(\log\log B)^{\epsilon}+nm(\log B)^{1+\epsilon})$ bit operations. Finally, replace the block to the right of the centre block in T by $\bar{E}$ and reduce modulo $d_id_{i-1}M$ (now using the new d_i's) all entries in row i column j of T for $p-m\le i\le p$ and $p+1\le j\le n$. This ends the proof of Theorem 18. ■

We now discuss how to choose the parameters p and m which decide exactly what block of T to reduce. We follow closely the presentation in Schönhage [14], which we sketch now. Progress of the 2-reduction of T is related to the quantity

$$\prod_{i=1}^{n-1}d_i = \prod_{j=1}^{n-1}|b_j^*|^{2(n-j)} \quad \text{resp.} \quad \lambda = \sum_{j=1}^{n-1}(n-j)\log|b_j^*|^2.$$

The quantity λ is initially satisfies $0\le\lambda\le(n-1)\log B$ and never increases during the algorithm. On the other hand, each 2-reduction step decreases λ by at least $\log(4/3)$ thus bounding the total number of 2-reduction steps by $O(n\log B)$. All 2-reduction steps will therefore not require more than $O(nm_0(\log B)^{1+\epsilon})$ bit operations. We need the following pair of lemmas from Schönhage [14].

Lemma 20 (Lemma 4.1 in [14]) *A block reduction at row p of size m will decrease the value of λ to some λ' such that*

$$\lambda-\lambda' \ge S_{p,m} = \sum_{i=0}^{m}(i-m/2)(p-i+\log d_{p-i}-\log d_{p-i-1}) \tag{30}$$

■

Lemma 21 (Lemma 4.2 in [14]) *For given $n \geq 4$, let $\tau > 0$, $m_0 \leq n-2$ be such that*

$$\tau \left(4 + \frac{12(n-m_0-2)}{(m_0+1)(m_0+2)}\right) \leq 1 \tag{31}$$

holds. Then any basis $b_1, b_2, \ldots, b_n \in \mathbf{Z}^{n\times n'}$ satisfying $S_{p,m} \leq \tau mn$ for all admissible pairs p, m with $m \leq m_0$ is semi-reduced. ∎

The semi-reduction algorithm is now easily described. The algorithm takes as input an index m_0 which bounds the maximum block size m throughout the algorithm. The idea is to restrict block-reductions to those cases with $S_{p,m} > \tau mn$ as per Lemma 21. We present here only the semi-reduction phase.

```
ModSemiReductionL(A, T, M, n, m0)
# Inplace semi-reduce the n-dimensional integer lattice A and F = FF(AA^t)AA^t.
# The input should satisfy n >= 4 and 1 <= m0 <= n - 2.
τ <- (4 + 12(n-m0-2)/((m0+1)(m0+2)))^(-1) ;
p <- 2;
while p < n do
    flag := true;
    for m to min(m0, p - 1) do
        if S_{p,m} > τmn then
            perform block reduction at row p of size m;
            p <- p - m;
            flag := false;
            break;
        fi;
    od;
    if flag then p <- p + 1 fi;
od;
```

The main idea of Schönhage's algorithm is to balance the overhead cost of all block reductions with the cost of all 2-reductions. In the case of the algorithm presented in [14], it was optimal to have τ constant and choose m_0 such that $m_0^2 = O(n)$. In our case, we get the following.

Lemma 22 *With a choice of $m_0 = \lceil n^{1/(5-\theta)} \rceil$,* `ModSemiReduction` *requires $O(n^{1+1/(5-\theta)}(\log B)^{1+\epsilon})$ bit operations.*

Proof. First note that for any n and θ with $n \geq 4$ and $2 < \theta \leq 3$, the choice of the lemma for m_0 satisfies the conditions of Lemma 21, that is, m_0 will satisfy $1 \leq m_0 \leq n-2$ for any θ with $2 < \theta \leq 3$. The total cost for all 2-reduction steps will be bounded by $O(nm_0(\log B)^{1+\epsilon}) = O(n^{1+1/(5-\theta)}(\log B)^{1+\epsilon})$ bit operations. Since $m \leq m_0 \leq n^5$, we have $m^2(\log m) \leq nm$. From Theorem 18 the overhead cost of a block reduction of size m is seen to be bounded by

$$cnm^{\theta-1}(\log B)(\log\log B)^{\epsilon} + cnm(\log B)^{1+\epsilon} \tag{32}$$

bit operations for absolute constant c. A block reduction of size m reduces the quantity $\lambda = \sum_{j=1}^{n-1}(n-j)\log|b_j^*|^2$ by at least τnm so that the overhead cost $C(m)$ per unit reduction for a block of size m is given by dividing the quantity (32) by τnm. We get

$$\begin{aligned} C(m) &\leq (cm^{\theta-2}(\log B)(\log\log B)^{\epsilon} + c(\log B)^{1+\epsilon})\left(4 + \frac{12(n-m_0-2)}{(m_0+1)(m_0+2)}\right) && (33)\\ &\leq (cm_0^{\theta-2}(\log B)(\log\log B)^{\epsilon} + c(\log B)^{1+\epsilon})(4 + 12n/m_0^2) \\ &\leq (c(n^{(\theta-2)/(5-\theta)} + 1)(\log B)(\log\log B)^{\epsilon} + c(\log B)^{1+\epsilon})(4 + 12n^{1-2/(5-\theta)}) \\ &\ll n^{1/(5-\theta)}(\log B)(\log\log B)^{\epsilon} + n^{(3-\theta)/(5-\theta)}(\log B)^{1+\epsilon} && (34) \end{aligned}$$

Since $\lambda \leq O(n \log B)$ initially, the total cost for all block reductions is bounded by

$$O((n \log B) \cdot (n^{1/(5-\theta)}(\log B)(\log\log B)^{\epsilon} + n^{(3-\theta)/(5-\theta)}(\log B)^{1+\epsilon}) = O(n^{1+1/(5-\theta)}(\log B)^{1+\epsilon})$$

bit operations as required. ■

Under the assumption of standard integer and matrix multiplication, we get the following.

Lemma 23 *With a choice of $m_0 = \lceil n^{1/3} \rceil$,* `ModSemiReduction` *requires $O(n^{1+1/3}(\log B)^3(\log\log B))$ bit operations using standard integer and matrix multiplication.*

Proof. First note that for any n with $n \geq 4$, the choice of the lemma for m_0 satisfied $1 \leq m_0 \leq n-2$. By Theorem 18, the total cost of all 2-reduction steps will be bounded by $O((n \log B) \cdot (m_0(\log B)^2)) = O(n^{1+1/3}(\log B)^3)$ bit operations. From (33), the overhead cost per unit reduction for a block of size m is given by

$$\begin{aligned} C(m) &\leq (cm(\log B)(\log\log B) + c(\log B)^2)\left(4 + \frac{12(n - m_0 - 2)}{(m_0+1)(m_0+2)}\right) \\ &\leq (cm_0(\log B)(\log\log B) + c(\log B)^2)(4 + 12n/m_0^2) \\ &\leq (c(n^{1/3}+1)(\log B)(\log\log B) + c(\log B)^2)(4 + 12n^{1/3}) \\ &\ll n^{1/3}(\log B)^2(\log\log B)) \end{aligned}$$

so that the total overhead cost becomes $O((n \log B) \cdot (n^{1/3}(\log B)^2(\log\log B))) =$
$O(n^{1+1/3}(\log B)^3(\log\log B))$ bit operations as required. ■

Theorem 24 *There exists a deterministic algorithm that takes as input a integer lattice L in the form of a basis $b_1, b_2, \ldots, b_n \in \mathbf{Z}^{n'}$ and returns as output a semi-reduced and size-reduced basis for L. If $n' = O(n)$ then the cost of the algorithm is $O(n^{1+1/(5-\theta)}(\log B)^{2+\epsilon})$ bit operations where B is a bound satisfying (13). If only a bound $B_o \geq |b_1|^2, |b_2|^2, \ldots |b_n|^2$ is known, then the running time of the algorithm is $O(n^{3+1/(5-\theta)+\epsilon}(\log B_o)^{2+\epsilon})$ bit operations. In the case of standard integer and matrix multiplication, the cost of the algorithm is $O(n^{1+1/3}(\log B)^3(\log\log B))$ or $O(n^{1+1/3}(\log B_o)^3(\log n + \log\log B_o))$ bit operations.*

Proof. There are three steps: (1.) Compute $T = \mathrm{FF}(AA^t)AA^t$ at a costs of $O(n^{\theta}(\log n)(\log B)^{1+\theta})$ bit operations using the algorithm of Theorem 9; (2.) 2-reduce the lattice using the algorithm `ModSemiReductionL` — by Lemma 22 this costs $O(n^{1+1/(5-\theta)}(\log B)^{2+\epsilon})$ bit operations; (3.) Size-reduce the lattice at a cost of $O(n^{\theta}(\log B)^{1+\epsilon})$ using the algorithm of Theorem 15. Since $\theta - 1 < (1 + 1/(5-\theta))$ for $\theta < 2.381$ and $\log B \geq 2^n$, the complexity of steps (1.) and (3.) simplifies to $O(n^{1+1/(5-\theta)}(\log B)^{2+\epsilon})$ bit operations as required. The complexity result in terms of B_o follows by noting that the choice $B = B_o^n$ will satisfy (13).

In the case of standard integer and matrix multiplication, steps (1.), (2.) and (3.) cost $O(n^2(\log B)^2 + n^3(\log B)(\log\log B))$, $O(n^{1+1/3}(\log B)^3(\log\log B))$ and $O(n^3(\log B)(\log\log B) + n^2(\log n)(\log B)^2)$ bit operations respectively. (This follows from Theorem 9, 18 and Lemma 23 respectively.) Since $B \geq 2^n$ and $\log n = O(\log\log B)$ the complexity of steps (1.) and (3.) will be bounded by that of (2.). ■

6 Conclusions

The key to our approach in this paper was to consider the L^3-reduction algorithm as a *matrix* algorthm. In particular, we consider the input basis $b_1, b_2, \ldots, b_n \in \mathbf{Z}^{n'}$ to be given in the form of an $n \times n'$ integer matrix A. The L^3-reduction process can then be stated as: Find a unimodular transformation of A such that A is 2-reduced and size-reduced. This approach led to the discovery of a simple modification of the L^3-reduction algorithm which returns exactly the same reduced basis but requires a factor of $O(n)$ fewer arithmetic (and bit) operations than that of the origninal algorithm presented in [12].

We have also presented a modification of the semi-reduction algorithm of Schönhage [14] which improves the asymptotic running time bound by a factor of $O(n^{0.618})$ bit operations. Under the assumption of standard integer and matrix multiplication, we have improved the running time bound for semi-reduction by a factor of $O(n^{0.666})$ bit operations.

For the following problems:

1. Gram-Schmidt orthonormalizing A.
2. Size-reducing A.

we have given algorithms with asymptotic running times that are within a logarithmic factor of that required to compute the determinant of A (when A is square). In the case of standard integer and matrix multipliation, we give algorithms for these problems that use a homomorphic imaging scheme to avoid much of the large integer arithemetic — their cost is comparable to that of computing the adjoint of the input matrix. Finally, we mention that the asymptotically fast fraction-free Gaussian elimination algorithm presented in Section 3 should prove useful for many other applications, for example, computing Smith normal forms of matrices over polynomial domains [16].

References

[1] Alfred V. Aho, John E. Hopcroft, and Jeffrey D. Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley, 1974.

[2] E. H. Bareiss. Sylvester's identity and multistep integer-preserving Gaussian elimination. *Mathematics of Computation*, 22(103):565–578, 1968.

[3] E. H. Bareiss. Computational solution of matrix problems over an integral domain. *Phil. Trans. Roy. Soc. London*, 10:68–104, 1972.

[4] J. Bunch and J. Hopcroft. Triangular factorization and inversion by fast matrix multiplication. *Mathematics of Computation*, pages 231–236, 1974.

[5] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. *Journal of Symbolic Computation*, 9:251–280, 1990.

[6] Keith O. Geddes, S. R. Czapor, and George Labahn. *Algorithms for Computer Algebra*. Kluwer, Boston, MA, 1992.

[7] Mark Giesbrecht. *Nearly Optimal Algorithms for Canonical Matrix Forms*. PhD thesis, University of Toronto, 1993.

[8] James L. Hafner and Kevin S. McCurley. Asymptotically fast triangularization of matrices over rings. *SIAM Journal of Computing*, 20(6):1068–1083, December 1991.

[9] J. Hastad, B. Just, J. C. Lagarias, and C. P. Schnorr. Polynomial time algorithms for finding integer relations among real numbers. *SIAM Journal of Computing*, 18(5):859—881, October 1989.

[10] George Havas, Derek F. Holt, and Sarah Rees. Recognizing badly presented Z-modules. *Linear Algebra and its Applications*, 192:137–163, 1993.

[11] A. Karatsuba and Y. Ofman. Multiplication of multidigit numbers on automata. *Soviet Physics-Doklady*, 7:595–596, 1963.

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

[13] C. P. Schnorr. A more efficient algorithm for lattice basis reduction. *Journal of Algorithms*, 9:47–62, 1988.

[14] A. Schönhage. Factorization of univariate integer polynomials by diophantine approximation and an improved basis reduction algorithm. In *Lecture Notes in Computer Science 127*, volume 7, pages 436–447. Springer-Verlag, 1984.

[15] A. Schönhage and V. Strassen. Schnelle Multiplikation grosser Zahlen. *Computing*, 7:281–292, 1971.

[16] Arne Storjohann and George Labahn. A fast Las Vegas algorithm for computing the Smith normal form of a polynomial matrix. *Linear Algebra and its Applications*, 253:155—173, 1997.

[17] S. Winograd. The algebraic complexity of functions. In *Proc. International Congress of Mathematicians, Vol. 3*, pages 283–288, 1970.