On the Complexity of Finding Short Vectors in Integer Lattices*

by

Erich Kaltofen

University of Toronto
Department of Computer Science
Toronto, Ontario M5S 1A4, Canada

Abstract. In [Lenstra, A., et al. 82] an algorithm is presented which, given n linearly independent n-dimensional integer vectors, calculates a vector in the integer lattice spanned by these vectors whose Euclidean length is within a factor of $2^{(n-1)/2}$ of the length of the shortest vector in this lattice. If B denotes the maximum length of the basis vectors, the algorithm is shown to run in $O(n^6(\log B)^3)$ binary steps. We prove that this algorithm can actually be executed in $O(n^6(\log B)^2 + n^5(\log B)^3)$ binary steps by analyzing a modified version of the algorithm which also performs better in practice.

1. Introduction

Various diophantine problems can be reduced to the problem of finding a short or a shortest vector in an integer lattice. Among them are integer linear programming with a fixed number of variables [Lenstra, H. 81], integer polynomial factorization [Lenstra, A., et al. 82], [Kaltofen 82], and the breaking of some variants of the Merkle-Hellman knapsack encryption scheme [Lagarias 82], [Odlyzko 82]. A. Lenstra, H. Lenstra and L. Lovász [Lenstra, A., et al. 82] present an algorithm which, given a set of n linearly independent n-dimensional integer vectors whose Euclidean lengths are bounded by B, finds a vector in the $\mathbf{Z}$-span of these vectors the length of which is within a factor of $2^{(n-1)/2}$ of the length of the shortest vector in this lattice. Their algorithm is shown to run in $O(n^6(\log B)^3)$ binary steps provided that classical integer arithmetic is used. In this paper we present a modified version of this algorithm which not only seems to perform better in practice, but we also prove that this version runs in $O(n^6(\log B)^2 + n^5(\log B)^3)$ binary steps when using classical integer arithmetic. With our result one can prove that factoring an integer polynomial f of degree n can be accomplished in $O(n^{11} + n^8(\log|f|)^3)$ binary steps, where $|f|$ is the length of the coefficient vector of f (cf. [Lenstra, A., et al. 82]).

---

* The research for this paper has been partially supported by the Connaught Fund, Grant # 3-370-126-80.
This paper appeared in: Proc. EUROCAL '83, *Springer Lec. Notes Comp. Sci.* **162**, 236-244 (1983).

If one employs fast integer multiplication the running times compare as follows.

Using a $O(K^{1+\varepsilon})$ complex K-digit times K-digit integer multiplication procedure where ε is a small constant > 0 (cf. [Knuth 81, p. 280]): $O(n^5(\log B)^{2+\varepsilon} + n^{5+\varepsilon}(\log B)^2)$ (ours) vs. $O(n^{5+\varepsilon}(\log B)^{2+\varepsilon})$ ([Lenstra, A., et al. 82]).

Using a $O(K \log K \log\log K)$ complex K-digit times K-digit integer multiplication procedure (cf. [Schönhage and Strassen 71]): $O(n^5(\log B)^2 \log K \log\log K)$ with $K = n + \log B$ (ours) vs. $K = n \log B$ ([Lenstra, A., et al. 82]). Notice that in this case both bounds are asymptotically equal and our improvement only lowers the constant multiplier for the given complexity.

In order to make this paper sufficiently self-contained we shall repeat some of the arguments presented in [Lenstra, A., et al. 82] as well as adopt most of the notation used there. Let $b_1,\ldots,b_n \in \mathbf{Z}^n$ be linearly independent (over $\mathbf{Q}$, the rationals). By $b_1^*,\ldots,b_n^*$ we denote the orthogonalization of this basis, namely

$$b_i^* = b_i - \sum_{j=1}^{i-1} \mu_{ij} b_j^*, \; 1 \le i \le n \tag{1.1}$$

$$\mu_{ij} = \frac{(b_i, b_j^*)}{|\,b_j^*|^2}, \; 1 \le j < i \le n \tag{1.2}$$

where (,) denotes the scalar product and $|\;\;|$ the square norm. The basis $b_1,\ldots,b_n$ is called *reduced* if

$$|\,b_k^*|^2 \ge \tfrac{1}{2} |\,b_{k-1}^*|^2 \text{ for } 1 < k \le n. \tag{1.3}$$

Notice that our notion of reduced basis is weaker than the one given in [Lenstra, A., et al. 82].

Lemma 1 (cf. [Lenstra, A., et al. 82, Proposition 1.11]): Let $b_1,\ldots,b_n \in \mathbf{Z}^n$ form a reduced basis. Then for any non-zero vector $x \in \mathbf{Z}b_1 + \cdots + \mathbf{Z}b_n$

$$2^{n-1} |\,x\,|^2 \ge |\,b_1|^2.$$

Hence $|\,b_1|$ is within a factor of $2^{(n-1)/2}$ of the norm of the shortest vector in the lattice spanned by $b_1,\ldots,b_n$.

Proof: Let $x = \sum_{i=1}^{l} r_i b_i = \sum_{i=1}^{l} r_i^* b_i^*$ with $r_i \in \mathbf{Z}$, $r_i^* \in \mathbf{Q}$ and $r_l \ne 0$, $1 \le l \le n$. Since b_l^* is orthogonal to b_i for $i = 1,\ldots, l-1$, $(x, b_l^*) = r_l(b_l, b_l^*) = r_l^*(b_l^*, b_l^*)$. But $(b_l, b_l^*) = (b_l^*, b_l^*) \ne 0$. Therefore, $r_l^* = r_l$, which is an integer $\ne 0$. It follows that $|\,x\,|^2 = \sum_{i=1}^{l} (r_i^*)^2 |\,b_i^*|^2 \ge (r_l^*)^2 |\,b_l^*|^2 \ge |\,b_l^*|^2 \ge |\,b_1^*|^2/2^{l-1}$, the last inequality by using (1.3) inductively. Since $b_1^* = b_1$ the lemma is proven. □

2. The Basis Reduction Algorithm

Algorithm R
[Given n linearly independent vectors $b_1,\dots,b_n$ with integer entries this algorithm transforms this basis into a reduced one.]

(I) [Initialize the arrays μ and β such that μ_{ij}, $1 \le j < i \le n$, and $\beta_l = |b_l^*|^2$, $1 \le l \le n$, satisfy (1.1) and (1.2):]

```
FOR $i \leftarrow 1$ TO $n$ DO

    [The following loop is skipped for $i = 1$.]
    FOR $j \leftarrow 1$ TO $i-1$ DO
        $\mu_{ij} \leftarrow \frac{1}{\beta_j}\left((b_i,b_j) - \sum_{l=1}^{j-1}\mu_{jl}\mu_{il}\beta_l\right)$.
    $\beta_i \leftarrow |b_i|^2 - \sum_{l=1}^{i-1}\mu_{il}^2\beta_l$.

$k \leftarrow 2$.
```

(L) [At this point μ and β correspond to the orthogonalization of $b_1,\dots,b_n$. Moreover $b_1,\dots,b_{k-1}$ is reduced, i.e.

$$|b_l^*|^2 \ge \tfrac{1}{2}|b_{l-1}^*|^2,\ 1 < l \le k-1, \text{ and } |\mu_{ij}| \le \tfrac{1}{2},\ 1 \le j < i \le k-1. \qquad (2.1)]$$

```
IF $k = n+1$ THEN RETURN $(b_1,\dots,b_n)$.

IF $k = 1$ THEN $k \leftarrow 2$; GOTO step (L).

Call algorithm S given below.

[Now (2.1) is also valid for $i = k$, i.e.
    $|\mu_{kj}| \le \frac{1}{2}$ for $1 \le j < k$.        (2.2)]

IF $\beta_k \ge \frac{1}{2}\beta_{k-1}$ THEN $k \leftarrow k+1$; GOTO step (L).

[At this point
    $|b_k^*|^2 < \frac{1}{2}|b_{k-1}^*|^2 \le \left(\frac{3}{4} - \mu_{k,k-1}^2\right)|b_{k-1}^*|^2$        (2.3)
the last inequality because of (2.2) for $j = k-1$.]

Interchange $b_{k-1}$ and $b_k$ and update the arrays $\mu$ and $\beta$ such that (1.1) and (1.2) is
satisfied for the new order of basis vectors.
```

[As we will explain below, the only entries which change are β_{k-1}, β_k and μ_{ij}, $i = k-1, k$, $1 \le j < i$, as well as $\mu_{i,k-1}$, μ_{ik}, $k < i \le n$. Let γ and ν denote the updated contents of β and μ, then, according to [Lenstra, A., et al. 82, 1.22]

$$\gamma_{k-1} = \beta_k + \mu_{k,k-1}^2\beta_{k-1},\ \nu_{k,k-1} = \frac{\mu_{k,k-1}\beta_{k-1}}{\gamma_{k-1}}, \tag{2.4}$$

$$\gamma_k = \beta_{k-1} - \nu_{k,k-1}^2\gamma_{k-1} = \frac{\beta_{k-1}\beta_k}{\gamma_{k-1}}, \tag{2.5}$$

$$\left.\begin{array}{l} \nu_{i,k-1} = \mu_{i,k-1}\nu_{k,k-1} + \mu_{ik}\beta_k/\gamma_{k-1} \\ \nu_{i,k} = \mu_{i,k-1} - \mu_{ik}\mu_{k,k-1} \end{array}\right\} \text{ for } k<i\le n, \tag{2.6}$$

$$\nu_{k-1,j} = \mu_{kj},\ \nu_{k,j} = \mu_{k-1,j} \text{ for } 1\le j<k-1. \tag{2.7)]}$$

$k \leftarrow k-1$; GOTO step (L). □

Algorithm S
[This algorithm replaces b_k by $b_k - \sum_{l=1}^{k-1}\lambda_l b_l$, $\lambda_l \in \mathbf{Z}$, such that the new μ_{kj} satisfy (2.2):]

```
(M) [Initialize the modulus:]
    $B_k \leftarrow \max\{\beta_1,\ldots,\beta_k\}$; $M \leftarrow \left\lceil \sqrt{(k+3)B_k} \right\rceil$.  [M can also be chosen the least power of
    the radix larger than that.]
(L1) FOR $m \leftarrow k-1$ DOWNTO 1 DO step (L2)
(L2) [Make $\mu_{km}$ absolutely smaller than $\frac{1}{2}$:]
    IF $|\mu_{km}| \ge \frac{1}{2}$ THEN
        $r \leftarrow$ ROUNDED($\mu_{km}$).  [ROUNDED($x$) denotes largest integer $z$ s.t. $|z-x| \le \frac{1}{2}$.]
        Replace $b_k$ by $(b_k - rb_m)$ mod $M$.                                  (2.8)
        [Notice that in this case $\lambda_m = r$, otherwise $\lambda_m = 0$.]
        [Readjust $\mu_{ki}$ for $i \le j < m$:]
        FOR $j \leftarrow 1$ TO $m-1$ DO  $\mu_{kj} \leftarrow \mu_{kj} - r\mu_{mj}$.      (2.9)
        $\mu_{km} \leftarrow \mu_{km} - r$.
    ENDIF
(B) Balance the residual entries of $b_k$ mod $M$ such that each modulus $> \frac{1}{2}M$ is replaced by
    its negative equivalent $\le \frac{1}{2}M$. □
```

Remark: If k is decremented in the main loop (L), (2.2) and (2.7) imply that $|\mu_{kj}| \le \frac{1}{2}$ for the new index k. Hence the subsequent call to algorithm S has no effect and therefore can be omitted. In practice, one should keep a switch which is set/reset as k is in-/decremented, and only call the subprocedure S if this switch is set.

3. Partial Correctness of the Reduction Algorithm

That step (I) computes the correct μ's and β's follows by substituting $b_j^* = b_j + \sum_{l=1}^{j-1} \mu_{jl} b_l^*$ into (1.2). We first focus on the subprocedure S. Since the vector $\sum_{m=1}^{k-1} \lambda_m b_m$ in the subspace spanned by $b_1,\ldots,b_{k-1}$ is subtracted from b_k, neither b_i^*, $1 \le i \le n$, nor μ_{ij}, $i \ne k$, $1 \le j < i$, change. By (1.2), the new μ_{kj} can be computed as

$$\mu_{kj} = \frac{1}{|b_j^*|^2}(b_k - \sum_{m=1}^{k-1} \lambda_m b_m,\, b_j^*) = \frac{(b_k, b_j^*)}{|b_j^*|^2} - \lambda_j - \sum_{m=j+1}^{k-1} \lambda_m \mu_{mj},$$

observing that $(b_j, b_j^*) = |b_j^*|^2$. But this is exactly the value computed within the loops (L1) and (L2). After these loops are executed

$$|\mu_{kj}| \le \tfrac{1}{2} \text{ for } 1 \le j < k.$$

Therefore the new vector b_k satisfies

$$|b_k|^2 = |b_k^*|^2 + \sum_{j=1}^{k-1} \mu_{kj}^2 |b_j^*|^2 \le B_k + (k-1)\frac{B_k}{4} = \frac{k+3}{4} B_k. \tag{3.1}$$

Thus the updated entries of b_k will be absolutely bounded by $\frac{1}{2}\sqrt{(k+3)B_k}$ and it suffices to compute these entries modulo an integer of twice this bound to obtain their true values in step (B).

The partial correctness of algorithm R can now be immediately concluded by the use of the invariant given at the label (L). The invariance of this statement is also easily established. The counter k is incremented only if $|b_k^*|^2 \ge \frac{1}{2}|b_{k-1}^*|^2$. Subprocedure S does not change any b_i^* or μ_{ij} for $i < k$. Hence with (2.2) the statement is true including the additional subscript. If the counter is decremented, β_i and μ_{ij}, $1 \le i \le k-2$, $1 \le j < i$, are not at all affected by the updates (2.4) - (2.7), which implies the invariance in this case as well.

4. Complexity Analysis

Following the ingenious argument laid out in [Lenstra, A., et al. 82] we shall first show that algorithm R must terminate by proving that k can be decremented at most $O(n^2 \log B)$ times, where

$$B = \max\{|b_1|^2,\ldots,|b_n|^2\}.$$

Of course, k can be incremented only an additional n times more which shows that the main loop (L) is executed $O(n^2 \log B)$ times.

Let d_l denote the Gramian of the vectors $\{b_1,\ldots,b_l\}$ that is

$$d_l = \det\Big((b_i,b_j)\Big)_{1\le i,j\le l} = \prod_{i=1}^{l} |b_i^*|^2 \text{ for } 1 \le i \le n. \tag{4.1}$$

(cf. [Gantmacher 59, Sec. 9.3].) From (1.1) it follows that all b_l^* initially satisfy $|b_l^*|^2 \le |b_l|^2 \le B$. Hence at the beginning

$$d_l \le B^l \text{ for } 1 \le l \le n. \tag{4.2}$$

The only time throughout the algorithm any of the Gramians changes is when some $|b_i^*|^2$ are updated. This only happens when k is decremented, that is, when b_{k-1} and b_k are interchanged. By (4.1) it follows that the Gramians d_l are invariant to the order of the vectors $b_1,\ldots,b_l$, hence the exchange does not affect any d_l with $l \ne k-1$. For $d_{k-1} = \prod_{i=1}^{k-1} |b_i^*|^2$, the only of the $|b_i^*|^2$, $1 \le i \le k-1$, being updated is $|b_{k-1}^*|^2$, which is, by (2.4) and (2.3), replaced by

$$|b_k^*|^2 + \mu_{k,k-1}^2 |b_{k-1}^*|^2 < \tfrac{3}{4} |b_{k-1}^*|^2. \tag{4.3}$$

Therefore the new value of d_{k-1} is at least $\frac{3}{4}$ times smaller than the old one, and since no other d_l changes, the same is true for the product $\prod_{l=1}^{n-1} d_l$. However, throughout the algorithm, this product remains a positive integer, which, by (4.2), is initially not greater than $B^{(n-1)n/2}$. Thus, the number of times k can be decremented is at most the number of times $\prod_{l=1}^{n-1} d_l$ can be reduced by the factor $\frac{3}{4}$, which is bounded by

$$\left\lfloor \frac{(n-1)n}{2} \log_{4/3} B \right\rfloor = O(n^2 \log B).$$

This establishes our initial claim.

Before we can derive the binary complexity of algorithm R, we need to estimate the lengths of intermediately computed integers. Since the entries in β and μ are rationals this includes predetermining their denominators. From (4.1) it follows that

$$|b_l^*|^2 = \frac{d_l}{d_{l-1}} \text{ for } 2 \le l \le n. \tag{4.4}$$

It can be easily established (cf. [Lenstra, A., et al. 82, 1.29]) that

$$d_j \mu_{ij} \in \mathbf{Z} \quad \text{for} \quad 1 \le j < i \le n.$$

Therefore all calculated denominators have, by (4.2), $O(n \log B)$ digits.

We now estimate $|b_i|^2$, $|b_i^*|^2$ and μ_{ij} throughout algorithm R. Initially

$$|b_i^*|^2 \le B \quad \text{for} \quad 1 \le i \le n, \tag{4.5}$$

and it is easy to see that this condition remains true during the algorithm. The only changes to $|b_i^*|^2$ occur in (2.4) and (2.5). But

$$0 < \gamma_{k-1} < \tfrac{3}{4}\beta_{k-1} \leq \tfrac{3}{4}B < B,$$

by (4.3), and by (2.5),

$$\gamma_k = \beta_{k-1} - \nu_{k,k-1}^2\gamma_{k-1} < \beta_{k-1} \leq B.$$

Therefore, the new values of $|b_{k-1}^*|^2$ and $|b_k^*|^2$ are not greater than B. This also implies that during algorithm R

$$|b_i|^2 \leq nB \quad \text{for} \quad 1 \leq i \leq n. \tag{4.6}$$

The set $\{b_1,\ldots,b_n\}$ is only changed by the subprocedure S, where b_k is updated. From (3.1) we conclude that the new value of $|b_k|^2$ satisfies

$$|b_k|^2 \leq \frac{k+3}{4}B_k < nB,$$

the later inequality by (4.5). Finally, we conclude that at the label (L)

$$\mu_{ij}^2 \leq \frac{|b_i|^2}{|b_j^*|^2} = \frac{d_{j-1}|b_i|^2}{d_j} \leq d_{j-1}|b_i|^2 \leq nB^j \text{ for } 1 \leq j < i \leq n,$$

using Schwarz' inequality on (1.2), then (4.4), (4.2) and (4.6) (cf. [Lenstra, A. et al. 82, 1.35]). It now follows from specification of algorithm R and (2.4) - (2.7) that any intermediate integer or any rational numerator has at most $O(n \log B)$ digits. Therefore, each arithmetic operation in algorithm R takes $O(n^2(\log B)^2)$ binary steps, using classical integer arithmetic routines.

Now let T_S denote the binary complexity of the subprocedure S. The arithmetic complexity of step (I) is $O(n^3)$, that of step (L), excluding the call to algorithm S, $O(n^3 \log B)$. Therefore, the binary complexity of the whole reduction is

$$O(n^5(\log B)^3 + T_S\, n^2 \log B) \tag{4.7}$$

with classical arithmetic. It remains to determine T_S.

Lemma 2: Let $0 < K \leq N$ be integers. To multiply a K-digit integer with an N-digit integer or to compute the integer quotient and remainder of a $K+N$-digit integer divided by an N-digit integer has binary complexity $O(M(K)N/K)$ where $O(M(K))$ is the binary complexity of multiplying two K digit integers.

Proof: The classical complexity as well as the one for fast multiplication can be found in [Knuth 82, Sec. 4.3, esp. Sec. 4.3.3, Exercise 13]. To establish the fast division complexity, we observe that by grouping the digits of the dividend into integers of K digits, we can determine the quotient by just considering the first few groups [Knuth 82, loc. cit.]. This amounts to dividing integers of size $O(K)$ and then multiplying $O(N/K)$ integers of that size. Division and multiplication can be accomplished in $O(M(K))$ binary steps. □

It is clear that the arithmetic complexity of algorithm S is $O(n^2)$. Each coordinate update of b_k in (2.8) as well as assignment (2.9) is executed that many times. We first focus on the binary complexity of the later, assignment (2.9). Upon entry to algorithm S,

$$|\mu_{kj}| < \sqrt{n 2^n B} \equiv C \text{ for } 1 \le j < k. \tag{4.9}$$

To prove this, we use the invariant at label (L) and the fact that $|b_1^*| = |b_1| \ge 1$. For, by (4.6)

$$nB \ge |b_k|^2 = |b_k^*|^2 + \sum_{l=1}^{k-1} \mu_{kl}^2 |b_l^*|^2 > \mu_{kj}^2 |b_j^*|^2 \ge \mu_{kj}^2 \frac{|b_1^*|^2}{2^{j-1}} > \frac{\mu_{kj}^2}{2^n},$$

which proves (4.9). Let $\mu_{kj}^{(m)}$ denote μ_{kj} after (2.9) has been executed for the indices m and j, $j < m$. We can prove by induction that

$$\left| \mu_{kj}^{(m)} \right| < 2^{k-m} C. \tag{4.10}$$

For $m = k$, the $\mu_{kj}^{(k)}$ are the original μ_{kj} and (4.10) reduces to (4.9). We now deduce (4.10) for $m-1$ from (4.10) for m. By (2.9), $\mu_{kj}^{(m-1)} = \mu_{kj}^{(m)} - r\mu_{mj}$, $1 \le j \le m-2$, where r is the rounded value of $\mu_{k,m-1}^{(m)}$. Hence,

$$|r| < 2\left| \mu_{k,m-1}^{(m)} \right| \le 2^{k-m+1} C \tag{4.11}$$

and $|\mu_{mj}| \le \frac{1}{2}$ by the invariant at label (L). Therefore, $\left| \mu_{kj}^{(m-1)} \right| \le \left| \mu_{kj}^{(m)} \right| + |r|\,|\mu_{mj}| \le 2^{k-m} C + \frac{1}{2} 2^{k-m+1} C = 2^{k-(m-1)} C$, as was to be shown. Thus, throughout algorithm S

$$|\mu_{kj}| \le 2^{k-1} C < \sqrt{n 2^{3n} B} \text{ for } 1 \le j < k. \tag{4.12}$$

This means that r is of size $O(n + \log B)$ and by the above lemma ROUNDED(μ_{km}) and the assignment (2.9) can be computed in $O((n+\log B) n \log B)$ binary steps. Obviously, the update of the coordinates of b_k in (2.8) can be achieved in fewer binary steps since $M \ll C$. In fact, by chosing M to be the least power of the radix larger than $\sqrt{(k+3)B}$, one can avoid the cost for the division by M when computing the modulus. Therefore,

$$T_S = O((n+\log B)\, n^3 \log B)$$

and the complexity for the whole algorithm, (4.7) and (4.8), becomes

$$O(n^6 (\log B)^2 + n^5 (\log B)^3),$$

assuming that the classical arithmetic procedures are employed. The running times cited in section 1 for fast integer arithmetic can be easily deduced from lemma 2 and the above analysis.

5. Conclusion

We have shown that the complexity bound in [Lenstra, A., et al. 82] for computing a short vector in an integer lattice is too pessimistic, in fact that the total exponent can be lowered by one. We claim that our version also has its practical merits. The main gain is during the update of b_k in (2.8), provided that M is chosen an appropriate power of the radix. If no modulus were taken, by (4.11) we would have to multiply with an r which could be $3n/2$ bits longer than $r \bmod M$. Since this multiplication might be executed $O(n^4 \log B)$ times the loss of time may become quite significant.

Other practical improvements have been suggested by A. Odlyzko [Odlyzko 82]. For example, the entries in μ and β could be made floating point numbers with extended precision. Then, whenever the loss of significant digits during roundoff becomes too great to decide $\beta_k \geq \frac{1}{2}\beta_{k-1}$ in algorithm R or calculate ROUNDED(μ_{km}) in algorithm S, one can recompute the μ's and β's like in step (I) of algorithm R. This change has been successfully used to solve large problems by the VAXIMA version of the algorithm.

Acknowledgement. The presentation of this paper has greatly benefitted from the remarks of two unnamed referees.

REFERENCES

[Gantmacher 59]

Gantmacher, F. R.: Matrix Theory, vol. 1. New York: Chelsea 1959.

[Kaltofen 82]

Kaltofen, E.: A Polynomial-Time Reduction from Bivariate to Univariate Integral Polynomial Factorization. Proc. 23rd Symp. Foundations of Comp. Sci., IEEE 57-64 (1982).

[Knuth 81]

Knuth, D. E.: The Art of Computer Programming, vol.2, Seminumerical Algorithms, 2nd ed. Reading, MA: Addison Wesley 1981.

[Lagarias 82]

Lagarias, J.C.: The Computational Complexity of Simultaneous Diophantine Approximation Problems. Proc. 23rd Symp. Foundations of Comp. Sci., IEEE 32-39 (1982).

[Lenstra, A., et al. 82]

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

[Lenstra, H. 81]

Lenstra, H.W., jr.: Integer Programming with a Fixed Number of Variables. Univ. Amsterdam: Math. Inst. Report 81-03, 1981.

[Odlyzko 82]

Odlyzko, A.M.: Private Communications 1982.

[Schönhage and Strassen 71]

Schönhage, A., and Strassen, V.: Schnelle Multiplication grosser Zahlen. Computing **7,** 281-292 (1971).