Finding the closest lattice vector when it's unusually close

Philip Klein*
Brown University

Abstract

We show how randomized rounding can be applied to finding the closest lattice vector. Given the basis of a lattice, and given a vector x not in the lattice, the heuristic will with high probability find the vector in the lattice that is closest to x (according to Euclidean norm). The catch is that the time required by the heuristic depends on (1) the distance between x and the closest lattice vector and on (2) the quality of the basis supplied.

1 Introduction

The *closest lattice vector problem*, also called the *nearest lattice point problem*, is NP-hard, and no polynomial-time approximation algorithm is known with a performance ratio better than exponential. It seems worthwhile to identify circumstances in which the problem can be solved optimally.

As an example of where this arises, Furst and Kannan [2] give a distribution of n-dimensional instances of SubsetSum problems for which there is an algorithm that runs in polynomial time almost always. One ingredient in their result is an algorithm that, given a vector v and a basis of a lattice, finds the vector in the lattice that is closest to to v *assuming* the following condition holds:

> the distance from v to the lattice is less than half the length of the shortest Gram-Schmidt vector.

Indeed, they show that, assuming this condition holds, there is a unique vector closest to v.

Here the *Gram-Schmidt vectors* corresponding to a basis $\mathbf{b_1}, \ldots, \mathbf{b_n}$ are the vectors $\mathbf{b_1^\dagger}, \ldots, \mathbf{b_n^\dagger}$ where $\mathbf{b_i^\dagger}$ is the projection of $\mathbf{b_i}$ orthogonal to the vector space generated by $\mathbf{b_1}, \ldots, \mathbf{b_{i-1}}$. These are the vectors found by the Gram-Schmidt algorithm for orthogonalization.[1]

Our result can be viewed as a generalization of this algorithm—there is an algorithm that for any given k runs in $n^{k^2+O(1)}$ time and that finds the closest vector to v if the following condition holds:

> the distance from v to the lattice is at most k times the length of the shortest Gram-Schmidt vector.

1.1 Related results Some related results will help to put our result in perspective. For comparison, Kannan [4] gives a closest-vector algorithm that runs in time $\mathrm{poly}(n)n^n$ where n is the dimension of the lattice. Thus the algorithm presented here is only useful when $k = o(\sqrt{n})$. On the other hand, as mentioned above, Furst and Kannan gave an algorithm that runs in polynomial time and finds the closest vector when $k < 1/2$, so our algorithm is needed only when $k \geq 1/2$.

For any lattice basis and any vector v, the distance of v from the lattice is no more than half the *sum* of the lengths of the Gram-Schmidt vectors; furthermore, this bound is achievable. Thus our algorithm is useful only when the vector v is unusually close to the lattice.

How small can the smallest Gram-Schmidt vector be? One can choose a lattice and a basis for it so as to make the smallest Gram-Schmidt vector arbitrarily small in comparison to the shortest vector in the lattice. On the other hand, Lagarias, Lenstra, and Schnorr [5] have shown that for every lattice there exists a basis (the *Korkin-Zolotarev* basis) where the smallest Gram-Schmidt vector is at least $3/2n$ times the length of the shortest vector in the lattice. Even in this case, for our algorithm to be useful, the distance between the input vector and the lattice must be significantly less than the length of the shortest vector.

2 Notation

We use the following notation. Vectors are signified by bold face. For vectors $\mathbf{b_1}, \ldots, \mathbf{b_n}$, we denote by $V(\mathbf{b_1}, \ldots, \mathbf{b_n})$ the vector space generated by these vectors, and we denote by $L(\mathbf{b_1}, \ldots, \mathbf{b_n})$ the lattice they generate.

The Gram-Schmidt vectors corresponding to $\mathbf{b_1}, \ldots, \mathbf{b_n}$ are denoted $\mathbf{b_1^\dagger}, \ldots, \mathbf{b_n^\dagger}$. By definition, $\mathbf{b_i^\dagger}$ is the projection of $\mathbf{b_i}$ orthogonal to $V(\mathbf{b_1}, \ldots, \mathbf{b_{i-1}})$.

We give an algorithm that, on input $\mathbf{x}$ and

*research supported by NSF Grant CCR-9700146.

[1] Often the Gram-Schmidt algorithm is used to find an ortho*normal* basis, i.e. where the output vectors have norm 1. Here we assume no such normalization is performed.

$\mathbf{b_1}, \ldots, \mathbf{b_n}$, with high probability returns the vector $\mathbf{y} \in \mathbf{L}(\mathbf{b_1}, \ldots, \mathbf{b_n})$ that minimizes $|\mathbf{x}-\mathbf{y}|$. The time required is polynomial times $n^{|\mathbf{x}-\mathbf{y}|^2/\min_i |\mathbf{b_i^\dagger}|^2}$. That is, the time is polynomial if the distance from $\mathbf{x}$ to the lattice is not much more than the length of the smallest Gram-Schmidt vector.

3 The basic algorithm

First we give a procedure for random rounding of a rational to an integer. This procedure is suggested by, but not identical to, the randomized rounding method of Raghavan and Thompson. For example, it might "round" an integer to another integer.

$\texttt{randRound}_c(r)$:
Write $r = p + a$, where p is an integer and $0 \le a < 1$.
Let $b = 1 - a$.
Let $s = \sum_{i \ge 0} e^{-c(a+i)^2} + \sum_{i \ge 0} e^{-c(b+i)^2}$.
Randomly choose an integer Q according to the following distribution: for $i \ge 0$, $\Pr[Q = r - (a+i)] = e^{-c(a+i)^2}/s$ and $\Pr[Q = r + (b+i)] = e^{-c(b+i)^2}/s$.

LEMMA 3.1. $s \le \sum_{i \ge 0} e^{-ci^2} + e^{-c(1+i)^2}$

Define $s(c) = \sum_{i \ge 0} e^{-ci^2} + e^{-c(1+i)^2}$

Next, we incorporate the rounding procedure in a modification of a deterministic procedure used by Babai [1].

Let $\mathbf{b_1}, \ldots, \mathbf{b_n}$ be a basis for a lattice $\mathcal{L}$. Let $\mathbf{b_1^\dagger}, \ldots, \mathbf{b_n^\dagger}$ be the vectors derived via the Gram-Schmidt orthogonalization process.

Let c be a parameter to be determined. We use the following recursive procedure.

$\texttt{near}_A(\mathbf{x}, d)$:
Comment: Assume $\mathbf{x}$ lies in $V(\mathbf{b_1}, \ldots, \mathbf{b_d})$.
If $d = 0$ then return $\mathbf{x}$ (It is the zero vector).
Else,
  Let $r_d \mathbf{b_d^\dagger}$ be the projection of $\mathbf{x}$ in direction of $\mathbf{b_d^\dagger}$.
  Let $c_d = A|\mathbf{b_d^\dagger}|^2$.
  Let $\lambda_d := \texttt{randRound}_{c_d}(r_d)$.
  Let $\mathbf{x}' := \mathbf{x} + (\lambda_d - r_d)\mathbf{b_d^\dagger}$.
  Return $\texttt{near}_A(\mathbf{x}' - \lambda_d \mathbf{b_d}, d-1) + \lambda_d \mathbf{b_d}$.

LEMMA 3.2. $\mathbf{x}' - \lambda_d \mathbf{b_d}$ *lies in* $V(\mathbf{b_1}, \ldots, \mathbf{b_{d-1}})$.

Proof. The vector $\mathbf{b_d^\dagger}$ is the projection of $\mathbf{b_d}$ orthogonal to $V(\mathbf{b_1}, \ldots, \mathbf{b_{d-1}})$, and hence lies in $V(\mathbf{b_1}, \ldots, \mathbf{b_d})$. We assume $\mathbf{x}$ lies in $V(\mathbf{b_1}, \ldots, \mathbf{b_d})$, and hence so do $\mathbf{x}'$ and $\mathbf{x}' - \lambda_d \mathbf{b_d}$. The projection of $\mathbf{x}' - \lambda_d \mathbf{b_d}$ in the direction of $\mathbf{b_d^\dagger}$ is 0, so $\mathbf{x}' - \lambda_d \mathbf{b_d}$ lies in $V(\mathbf{b_1}, \ldots, \mathbf{b_d})$ but has no projection orthogonal to $V(\mathbf{b_1}, \ldots, \mathbf{b_{d-1}})$.

LEMMA 3.3. *The difference between* $\mathbf{x}$ *and* $\mathbf{x}'$ *is* $(\lambda_d - r_d)\mathbf{b_d^\dagger}$.

LEMMA 3.4. *For a vector* $\mathbf{x} \in V(\mathbf{b_1}, \ldots, \mathbf{b_d})$, $\texttt{near}_A(\mathbf{x}, n)$ *returns a vector*

$$\mathbf{y} = \lambda_1 \mathbf{b_1} + \cdots + \lambda_d \mathbf{b_d}$$

in $L(b_1, \ldots, b_d)$ *such that*

$$\mathbf{y} - \mathbf{x} = (\lambda_1 - r_1)\mathbf{b_1^\dagger} + \cdots + (\lambda_d - r_d)\mathbf{b_d^\dagger} \tag{3.1}$$

Proof. The vector $\mathbf{x}'$ in the procedure is assigned

$$\mathbf{x}' := \mathbf{x} + (\lambda_d - r_d)\mathbf{b_d^\dagger}$$

By the inductive hypothesis, the recursive call $\texttt{near}_A(\mathbf{x}' - \lambda_d \mathbf{b_d}, d-1)$ returns a lattice vector $\mathbf{y}' = \lambda_1 b_1 + \cdots + \lambda_{d-1} b_{d-1}$ such that

$$\mathbf{y}' - (\mathbf{x}' - \lambda_d \mathbf{b_d}) = (\lambda_1 - r_1)\mathbf{b_1^\dagger} + \cdots + (\lambda_{d-1} - r_{d-1})\mathbf{b_{d-1}^\dagger}$$

The original call then returns the vector $\mathbf{y} = \mathbf{y}' + \lambda_d \mathbf{b_d}$, and we have

$$\mathbf{y} - \mathbf{x}' = (\lambda_1 - r_1)\mathbf{b_1^\dagger} + \cdots + (\lambda_{d-1} - r_{d-1})\mathbf{b_d^\dagger}$$

Substituting $\mathbf{x} + (\lambda_d - r_d)\mathbf{b_d^\dagger}$ for $\mathbf{x}'$, we obtain

$$\mathbf{y} - \mathbf{x} = (\lambda_1 - r_1)\mathbf{b_1^\dagger} + \cdots + (\lambda_d - r_d)\mathbf{b_d^\dagger}$$

LEMMA 3.5. *Let* $\hat{\mathbf{y}}$ *be a vector in* $L(b_1, \ldots, b_d)$. *Let* $\mathbf{x}$ *be a vector in* $V(\mathbf{b_1}, \ldots, \mathbf{b_d})$. *The probability that* $\texttt{near}_A(\mathbf{x}, d)$ *returns* $\hat{\mathbf{y}}$ *is*

$$\left(\prod_{i \le d} 1/s(A|\mathbf{b_i^\dagger}|^2)\right) \exp(-A|\mathbf{y} - \mathbf{x}|^2)$$

Proof. Write $\hat{\mathbf{y}} = \delta_1 b_1 + \cdots + \delta_d b_d$, and consider the invocation of $\texttt{near}_A(\mathbf{x}, d)$. The value of r_d is determined by $\mathbf{x}$. Let E be the event that $\lambda_d = \delta_d$. Assuming E occurs, $\mathbf{x}' = \mathbf{x} + (\delta_d - r_d)\mathbf{b_d^\dagger}$. By the inductive hypothesis, the probability that the call $\texttt{near}_A(\mathbf{x}' - \delta_d b_d, d-1)$ returns $\hat{\mathbf{y}} - \delta_d b_d$ is

$$\left(\prod_{i < d} (1/s(A|\mathbf{b_i^\dagger}|^2))\right) \exp(-A|(\mathbf{x}' - \delta_d b_d) - (\hat{\mathbf{y}} - \delta_d b_d)|^2)$$

This is the probability that the original call $\texttt{near}_A(\mathbf{x}, d)$ returns $\hat{\mathbf{y}}$, assuming E occurs. Using Lemma 3.1, the probability of E is at least

$$(1/s(A|\mathbf{b_d^\dagger}|^2)) \exp(-(r_d - \delta_d)^2 A|\mathbf{b_d^\dagger}|^2)$$

By multiplying these two probabilities, we get a lower bound on the unconditioned probability that the original call returns $\hat{\mathbf{y}}$:

$$\left(\prod_{i \le d} 1/s(A|\mathbf{b_i^\dagger}|^2)\right)$$

$$\begin{aligned}
&\exp(-A(|(\mathbf{x}' - \delta_d \mathbf{b_d}) \\
&-(\hat{\mathbf{y}} - \delta_d \mathbf{b_d})|^2 + (r_d - \delta_d)^2 |\mathbf{b_d^\dagger}|^2)) \\
&\quad = \left(\prod_{i \le d} 1/s(A|\mathbf{b_i^\dagger}|^2) \right) \\
&\qquad \exp(-A(|\mathbf{x}' - \hat{\mathbf{y}}|^2 + (r_d - \delta_d)^2 |\mathbf{b_d^\dagger}|^2))
\end{aligned}$$

Since $\mathbf{x}' - \hat{\mathbf{y}} \in V(\mathbf{b_1}, \ldots, \mathbf{b_{d-1}})$ and $\mathbf{b_d^\dagger}$ is orthogonal to $V(\mathbf{b_1}, \ldots, \mathbf{b_{d-1}})$, we have

$$|\mathbf{x}' - \hat{\mathbf{y}}|^2 + (r_d - \delta_d)^2 |\mathbf{b_d^\dagger}|^2 = |(\mathbf{x}' + (r_d - \delta_d)\mathbf{b_d^\dagger}) - \hat{\mathbf{y}}|^2$$

so the probability is as stated in the lemma.

In order to find the nearest lattice vector to a vector $\mathbf{x}$, we call $\mathtt{near}_A(\mathbf{x}, n)$ many times, and output the nearest vector among those returned by these calls. The nearest vector returned will probably be in fact the nearest lattice vector if the number of calls is large enough. According to Lemma 3.5, the number of calls required depends on the distance of the nearest vector from $\mathbf{x}$, the product of the terms $1/s(\cdot)$, and the value of the parameter A.

4 Choice of parameters

Let $A = (\ln n)/\min_i |\mathbf{b_i^\dagger}|^2$. Then for every invocation of $\mathtt{near}_A(\mathbf{x}, d)$, we have $c_d \ge \ln n$. We use this bound in estimating $s(c_d)$, defined as

$$s(c_d) = \sum_{i \ge 0} e^{-c_d i^2} + e^{-c_d (1+i)^2}$$

We get

$$\begin{aligned}
s(c_d) &\le \sum_{i \ge 0} n^{-i^2} + n^{-(1+i)^2} \\
&= 1 + 2(n^{-1} + n^{-4} + n^{-9} + \ldots) \\
&= 1 + 2/n + O(n^{-4})
\end{aligned}$$

Hence

$$\begin{aligned}
\prod_{d \le n} s(c_d) &\le (\exp(\frac{2}{n} + O(n^{-4})))^n \\
&= e^2(1 + o(1))
\end{aligned}$$

With this choice of parameters, it follows from Lemma 3.5 that for any input vector $\mathbf{x}$ and any lattice vector $\hat{\mathbf{y}}$, the probability that a call to `near` returns $\hat{\mathbf{y}}$ is

$$\Omega(n^{-|\hat{\mathbf{y}} - \mathbf{x}|^2 / \min_i |\mathbf{b_i^\dagger}|^2}$$

5 The deterministic algorithm

We have presented a randomized algorithm, but it is easy to derandomize it. Consider one trial of `near`. It explores one path in a randomized decision tree of depth n, a rooted tree where the edges from each node to its children are labeled with probabilities that sum to 1. The probability of a path starting at the root is defined to be the product of the probabilities on the edges of the path.

The analysis shows that, for the closest lattice vector $\hat{\mathbf{y}}$, there is one path through the decision tree that leads to a leaf corresponding to $\hat{\mathbf{y}}$, and that path has probability $\Omega(n^{-|\hat{\mathbf{y}} - \mathbf{x}|^2 / \min_i |\mathbf{b_i^\dagger}|^2}$. Let p be this probability (or a lower bound on it).

Let S be the set of nodes v (not just leaves) in the decision tree where the probability of the root-to-v path is at least p. The deterministic algorithm is: explore the tree, visiting every node of S. Each leaf visited corresponds to a candidate closest vector; return the closest of these candidates.

By the analysis of the randomized algorithm, the deterministic algorithm will find the closest vector. It remains to determine how long the deterministic algorithm takes. Let S' be the subset of nodes $v \in S$ such that no children of v are in S. Note that $|S| \le n|S'|$. Consider a random root-to-leaf traversal of the tree, and let E_v denote the event that v is traversed during this traversal. For each $v \in S$, we have $\Pr[E_v] \ge p$. Furthermore, the events E_v for $v \in S'$ are disjoint events, so their probabilities add up to at most 1. Hence $|S'| \le 1/p$. Thus the deterministic algorithm traverses at most n/p nodes.

6 Proof of Lemma 3.1

Define $f_c^i(x) = \exp(-c(1 - x + i)^2) + \exp(-c(x + i)^2)$. Define $f_c(x) = \sum_{i \ge 0} f_c^i$. Our aim is to prove that $\max\{f_c(x) : 0 \le x \le 1\} = f_c(0)$. We assume $c \ge 6$.

Note that $f_c(1 - x) = f_c(x)$, so it suffices to prove the inequality for $0 \le x \le 1/2$.

The first derivative of $f_c(x)$ is

$$2c \sum_{i \ge 0} (1 - x + i) \exp(-c(1 - x + i)^2) - (x + i) \exp(-c(x + i)^2)$$

Lemma 6.1. *The first derivative is zero at $x = 0$.*

Proof. Let $h(j) = j \cdot \exp(-cj^2)$. The $i = 0$ term in the sum is $h(1)$ at $x = 0$. For $i > 0$, the term is $h(i+1) - h(i)$ at $x = 0$. Hence the sum of the first k terms is $h(k+1)$. The limit of $h(k+1)$ as $k \to \infty$ is 0, so the sum is zero.

The second derivative of $f_c(x)$ is

$$2c \sum_{j \ge 0} \frac{2c(j + 1 - x)^2 - 1}{\exp(c(j + 1 - x)^2)} + \frac{2c(j + x)^2 - 1}{\exp(c(j + x)^2)}$$

LEMMA 6.2. *The second derivative is negative for $x < 1/c$.*

Proof. For $x \leq 1/c$, the $j = 0$ term is bounded by

$$\begin{aligned} &(2c-1)\exp(-c(1-1/c)^2) + ((2/c)-1)\exp(-1/c) \\ &\leq (2c)\exp(-c(1-2/c)) + (2/c-1)(1-1/c) \\ &\leq (2c)\exp(-c+2) - (1-3/c) \end{aligned} \tag{6.2}$$

For $j > 1$, the j term is bounded by

$$(2c(j+1)^2)\exp(-c(1+j-1/c)^2) + (2c(j+1/c)^2)\exp(-cj^2) \tag{6.3}$$

and we have

$$\begin{aligned} &\exp(-c(1+j-1/c)^2) \\ &\leq \exp(-c((j+1)^2 - 2(1+j)/c)) \\ &\leq \exp(-(j^2+2j+1)c + 2(j+1)) \\ &\leq \exp(-cj^2)\exp(-2j(c-1)-c+2) \end{aligned}$$

so we can bound (6.3) by

$$\begin{aligned} &(2c(j+1)^2)\exp(-2j(c-1)-c+2)\exp(-cj^2) \\ &+(2c(j+1/c)^2)\exp(-cj^2) \\ &\leq \exp(-cj^2) + (2c(j+1/c)^2)\exp(-cj^2) \\ &\leq 3cj^2\exp(-cj^2) \end{aligned}$$

Hence the sum of terms $j \geq 1$ is less than

$$3\sum_{j\geq 1} cj^2\exp(-cj^2) \leq 3c\exp(-c)\sum_{k\geq 0} r^k$$

where $r = 4\exp(-3c)$. It follows that the sum is less than $3.1c\exp(-c)$.

Adding this bound to our bound (6.2) on the $j = 0$ term, we see that the second derivative is negative for $c \geq 6$ and $x \leq 1/c$.

Finally, the following lemma proves that the value of $f_c(x)$ for $0 < x < .5$ is less than the value for $x = 0$.

LEMMA 6.3. *For $0 < x < .5$, the first derivative is negative.*

Proof. By combining Lemma 6.1 and Lemma 6.2, we can infer that the first derivative is negative for $0 < x < 1/c$. To finish the proof, therefore, we need only consider $1/c \leq x < .5$

We show that each term j is negative, i.e. we show that

$$\begin{aligned} &(j+x)\exp(-c(j_x)^2) \\ &> (j+1-x)\exp(-c(j+1-x)^2) \end{aligned} \tag{6.4}$$

We substitute $x = .5-\epsilon$ into (6.4), obtaining as our new goal

$$\begin{aligned} &(j+.5-\epsilon)\exp(-c(j+.5-\epsilon)^2) \\ &> (j+.5+\epsilon)\exp(-c(j+.5+\epsilon)^2) \end{aligned} \tag{6.5}$$

To show (6.5), it suffices to show

$$\begin{aligned} &\frac{j+.5+\epsilon}{j+.5-\epsilon} \\ &< \exp(-c(j+.5-\epsilon)^2 + c(j+.5+\epsilon)^2) \end{aligned} \tag{6.6}$$

which can be rewritten as

$$\frac{1+\delta}{1-\delta} < \exp(4c(j+.5)\epsilon) \tag{6.7}$$

where $\delta = \epsilon/(j+.5)$.

We further rewrite (6.7) as

$$\begin{aligned} &(1+2\delta+2\delta^2+2\delta^3+\cdots) \\ &< 1 + (4c(j+.5)\epsilon) + \frac{1}{2!}(4c(j+.5)\epsilon)^2 \end{aligned} \tag{6.8}$$

$$+\frac{1}{3!}(4c(j+.5)\epsilon)^3 + \cdots \tag{6.9}$$

so, subtracting 1 from both sides, it suffices to show that

$$\sum_{k\geq 1}\left(\frac{(4c(j+.5))^k}{k!} - 2(\frac{1}{j+5})^k\right)\epsilon^k \tag{6.10}$$

is positive.

First we consider the case $j \geq 1$. The $k = 1$ term is

$$\left(4c(j+.5) - \frac{1}{j+.5}\right)\epsilon \tag{6.11}$$

As for the terms with $k \geq 2$, since $j \geq 1$, each such term is more than

$$-2\left(\frac{\epsilon}{j+.5}\right)^k$$

and the sum of such terms with $k \geq 2$ is more than $-3(\frac{\epsilon}{j+.5}^2)$. The absolute value of this sum is less than (6.11), so the sum of all terms is positive.

Now we consider the case $j = 0$. The $k = 1$ term is $4c(.5) - \frac{\epsilon}{.5} = (2c-4)\epsilon$, which is positive. The $k = 2$ term is

$$\frac{(4c/2)^2}{2!}\epsilon^2 - 2(2\epsilon)^2 = (2c^2-8)\epsilon^2 \tag{6.12}$$

The $k = 3$ term is at least $-2(2\epsilon)^3$. The sum of the terms with $k \geq 3$ is at least

$$-2(2\epsilon)^3(1+r+r^2+\cdots)$$

where $r = 2\epsilon$. Because we assume $x \geq 1/c$, we have $\epsilon \leq .5 - 1/c$, so $r \leq 1 - 2/c$. Hence $1 + r + r^2 + \cdots \leq c/2$, so the sum of the terms with $k \geq 3$ is at least

$$-2(2\epsilon)^3(c/2)$$

Adding this value to the $k = 2$ term, $(2c^2 - 8)\epsilon^2$, gives us a positive value as long as $4c < 2c^2 - 8$, which holds for $c \geq 5$.

Acknowledgements

Thanks to Ravi Kannan. Thanks also to a reviewer, who pointed out the deterministic algorithm.

References

[1] L. Babai, "On Lovász' lattice reduction and the nearest lattice point problem," *Combinatorica* (1986), pp. 1–13.

[2] M. L. Furst and R. Kannan, "Succinct certificates for almost all subset sum problems," *SIAM Journal on Computing* (1989), pp. 550–558.

[3] J. Hastad, "Dual vectors and lower bounds for the nearest lattice point problem," *Combinatorica* (1988), pp. 75–81.

[4] R. Kannan, "Minkowski's convex body theorem and integer programming," *Mathematics of Operations Research*, vol. 12 (1987), pp. 415-440.

[5] J. C. Lagarias, H. W. Lenstra, and C. P. Schnorr, "Korkin-Zolotarev Bases and successive minima of a lattice and its reciprocal lattice," *Combinatorica* (1990), pp. 333-348.

[6] A. K. Lenstra, H. W. Lenstra, and L. Lovász, "Factoring polynomials with rational coefficients," *Math. Ann., 261* (1982), pp. 513-534.