# Chapter 1
# The History of the LLL-Algorithm

**Ionica Smeets**
**In collaboration with Arjen Lenstra, Hendrik Lenstra, László Lovász, and Peter van Emde Boas**

**Abstract** The 25th birthday of the LLL-algorithm was celebrated in Caen from 29th June to 1st July 2007. The three day conference kicked off with a historical session of four talks about the origins of the algorithm. The speakers were the three L's and close bystander Peter van Emde Boas. These were the titles of their talks.

- *A tale of two papers* – Peter van Emde Boas.
- *The early history of LLL* – Hendrik Lenstra.
- *The ellipsoid method and basis reduction* – László Lovász.
- *Polynomial factorization and lattices in the very early 1980s* – Arjen Lenstra.

This chapter is based on those talks, conversations with these four *historic* characters, the notes that Peter van Emde Boas and Arjen Lenstra wrote for the preproceedings, and many artifacts from the phenomenal archive of Van Emde Boas.

**Fig. 1.1** On both pictures you see from left to right Peter van Emde Boas, László Lovász, Hendrik Lenstra, and Arjen Lenstra. Alexander Schrijver took the first picture in Bonn on 27th February 1982. For the poster of the conference, Van Emde Boas was digitally removed from this picture. The second picture was taken by Ghica van Emde Boas at Le moulin de Bully on 29th June 2007

I. Smeets (✉)
Mathematisch Instituut, Universiteit Leiden, Niels Bohrweg 1, 2333 CA Leiden, the Netherlands,
e-mail: ionica.smeets@gmail.com

P.Q. Nguyen and B. Vallée (eds.), *The LLL Algorithm*, Information Security
and Cryptography, DOI 10.1007/978-3-642-02295-1_1,
© Springer-Verlag Berlin Heidelberg 2010

## Skinny Triangles and Lattice Reduction

One possible starting point for the LLL-algorithm is May 1980. At that time, Peter van Emde Boas was visiting Rome. While he was there he discussed the following problem with Alberto Marchetti-Spaccamela.

**Question 1** *Given three points with rational coordinates in the plane, is it possible to decide in polynomial time whether there exists a point with integral coefficients lying within the triangle defined by these points?*

This question seemed easy to answer: for big triangles the answer will be "yes" and for small triangles there should be only a small number of integer points close to it that need checking. But for extremely long and incredibly thin triangles this does not work; see Fig. 1.2.

It is easy to transform such a skinny triangle into a "rounder" one, but this transformation changes the lattice too; see Fig. 1.3. Van Emde Boas and Marchetti-Spaccamela did not know how to handle these skewed lattices. Back in Amsterdam, Van Emde Boas went to Hendrik Lenstra with their question. Lenstra immediately replied that this problem could be solved with lattice reduction as developed by Gauss almost two hundred years ago. The method is briefly explained below.

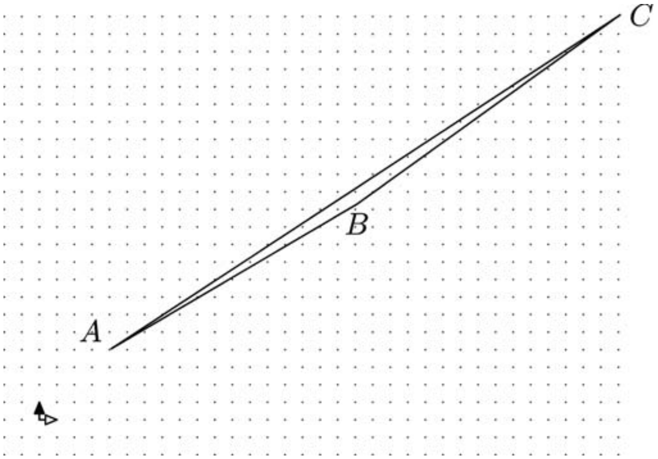


**Fig. 1.2** The problematic triangles almost look like a line: they are incredibly thin and very, very long. This picture should give you an idea; in truly interesting cases the triangle is much thinner and longer. In the lower left corner you see the standard basis for the integer lattice

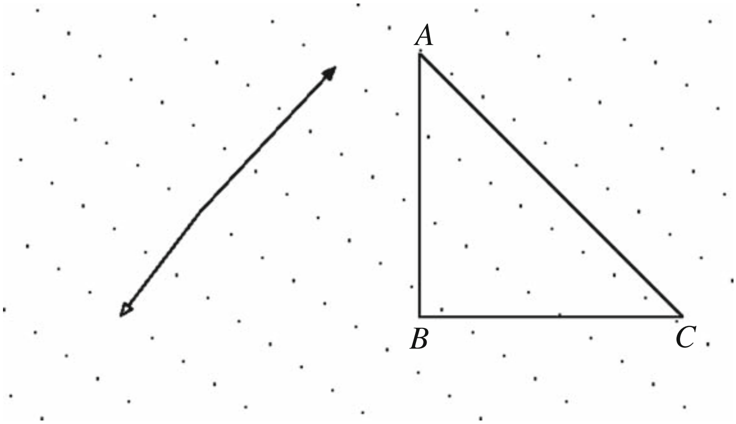


**Fig. 1.3** The triangle from Fig. 1.2 transformed into a right-angled isosceles triangle, the skewed lattice and the transformed standard basis. Now the transformed basis looks thin and long

**Method for Answering Question 1.**
First apply a linear transformation that changes the triangle into a right-angled isosceles triangle. This transforms the integer lattice into a lattice with some given basis of two rational vectors.
Find a reduced basis $(b_1, b_2)$ for this new lattice: $b_1$ is a shortest nonzero vector in the lattice and $b_2$ is a shortest vector in the lattice that is linearly independent of $b_1$. Compute $b_2^* = b_2 - \frac{\langle b_1, b_2 \rangle}{\langle b_1, b_1 \rangle} b_1$.
If the triangle is sufficiently large compared to $||b_2^*||$, then there is a lattice point in the triangle.
Otherwise, check if lines parallel to $b_1$ (with successive distances $||b_2^*||$) contain points in the triangle. Remember that in this case the size of the triangle is small compared to $||b_2^*||$, so the number of lines to be checked is small.

Van Emde Boas wrote to Marchetti in the summer of 1980: “Solution: the answer is yes.” In his letter he explained how the method worked. When Marchetti-Spaccamela was visiting Amsterdam in October of the same year, he paid Hendrik Lenstra a visit to talk about the solution. Together with Van Emde Boas, he went to Lenstra’s office. Hendrik Lenstra vividly remembers his initial feelings about this visit: “I felt a bit like a dentist. I had dealt with this problem before, so why were they asking the same question again? I told them the solution and they apparently understood it, but then they refused to walk out of my office. I had work to do and I felt that they were imposing upon my time. I was too naive to realize that this was my real work.”

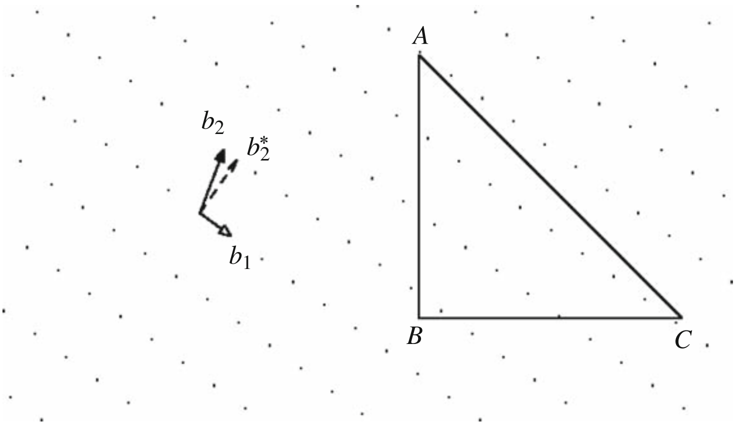


**Fig. 1.4** The skewed lattice, its reduced basis $(b_1, b_2)$ and the orthogonal projection $b_2^*$.

Lenstra opened his mouth about to say “Go away,” but he phrased this in a slightly more polite manner as: “Why is this question about the triangle interesting in the first place?” His visitors answered that it was just a special case of integer programming with a fixed number of variables. “And then I stared at it and asked, can you not do that in the same way?” Van Emde Boas recalls: “At this point I had to leave the meeting to teach a class. When I came back three quarters of an hour later, Hendrik had given the answer that it really works for any dimension.” This resulted in Lenstra’s integer linear programming algorithm.

Linear programming, sometimes known as linear optimization, is the problem of maximizing or minimizing a linear function over a convex polyhedron specified by linear nonnegativity constraints. Integer linear programming is a special case of linear programming in which all variables are required to take on integer values only.

## Integer Programming

In the early eighties, Hendrik Lenstra was not doing integer programming at all. He was among other things working on primality testing and Euclidean number fields. “I would probably not have found the integer linear programming algorithm if I had not been asked this question about a triangle in a integer lattice.” The generalized question can be stated as follows.

**Question 2** *Let $n$ and $m$ be positive integers, $A$ an $m \times n$-matrix with integral entries, and $b \in \mathbb{Z}^m$. Is there a vector $x \in \mathbb{Z}^n$ satisfying the system of $m$ inequalities $Ax \leq b$? So if $K = \{x \in \mathbb{R}^n : Ax \leq b\}$, then the question is whether $\mathbb{Z}^n \cap K$ is nonempty.*

The integer linear programming algorithm essentially consists of three stages.

**Integer Linear Programming.**
We may assume the problem is reduced to the case $0 < \operatorname{vol} K < \infty$, thus $K$ is bounded and has positive volume.

1. Find a linear transformation $\tau$ such that $\tau K$ is *round*. If we put

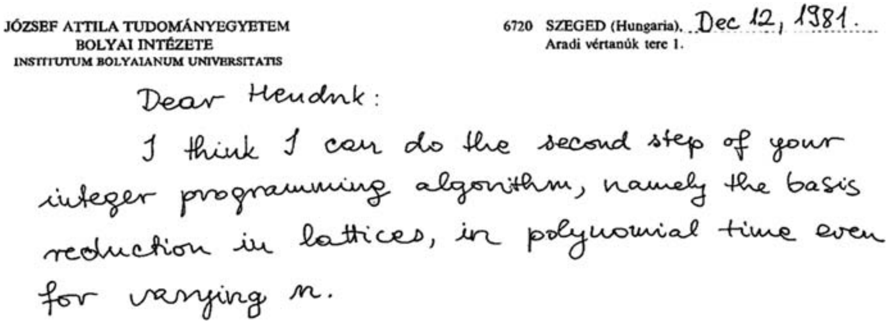
$$B(p,z) = \{x \in \mathbb{R}^n : |x - p| \leq z\} \quad \text{for } p \in \mathbb{R}^n, \quad z \in \mathbb{R}_{>0},$$

   then the formal definition of *round* is that there are spheres $B(p,r)$ and $B(p,R)$ with $B(p,r) \subset \tau K \subset B(p,R)$ and $\frac{R}{r} \leq c_1$, where $c_1$ is a constant depending only on $n$.
2. Find a reduced basis for $\tau\mathbb{Z}^n$.
3. *Either* find a point in $\tau\mathbb{Z}^n \cap \tau K$ *or* reduce the problem to a bounded number of problems in $n-1$ dimensions.

There are three versions of this algorithm: the first preprint appeared in April 1981 [3], to be followed by an improved version in November of the same year [4]. The final version was published in 1983 in *Mathematics of Operations Research* [5], the year after the LLL-algorithm appeared [8]. Lenstra:

> The reason that there are so many versions is that Lovász kept improving parts of the algorithm. He started with the first step. I had a very naive and straightforward way of finding

**Fig. 1.5** Hendrik Lenstra using his hands to explain the algorithm to Alberto Marchetti-Spaccamela, Amsterdam on 21st October 1980



JÓZSEF ATTILA TUDOMÁNYEGYETEM
BOLYAI INTÉZETE
INSTITUTUM BOLYAIANUM UNIVERSITATIS

6720 SZEGED (Hungaria), Dec 12, 1981.
Aradi vértanúk tere 1.

Dear Hendrik:

I think I can do the second step of your integer programming algorithm, namely the basis reduction in lattices, in polynomial time even for varying n.

**Fig. 1.6** The beginning of the letter from Lovász in which he explains the basis reduction algorithm

> the needed transformation, and this method was polynomial only for fixed $n$. Lovász found an algorithm to do this in polynomial time even for varying $n$.

Lovász later improved the second step, the basis reduction algorithm. In the introduction to the first preprint of his paper, Lenstra expressed some dissatisfaction with his complexity analysis of this step.

> It is not easy to bound the running time of this algorithm in a satisfactory way. We give an argument which shows that it is polynomially bounded, for fixed $n$. But the degree of this polynomial is an exponential function of $n$, and we feel that there is still room for improvement.

At the time Lenstra believed this problem was caused by his analysis, not by the algorithm. But Lovász improved the algorithm instead of the analysis.

In a letter dated 12th December 1981, Lovász explains the basis reduction algorithm. He defines two concepts that are at the core of the LLL-algorithm. Let $(b_1, \dots, b_n)$ be an ordered basis for $\mathbb{R}^n$. We say that it is *straightened* if for every $1 \le i < k \le n$ and

$$b_k = \sum_{j=1}^{i} \lambda^i_{jk} b_j + b_k^{(i)}, \text{ where } b_j^T b_k^{(i)} = 0 \text{ for } j = 1, \dots, i,$$

one has

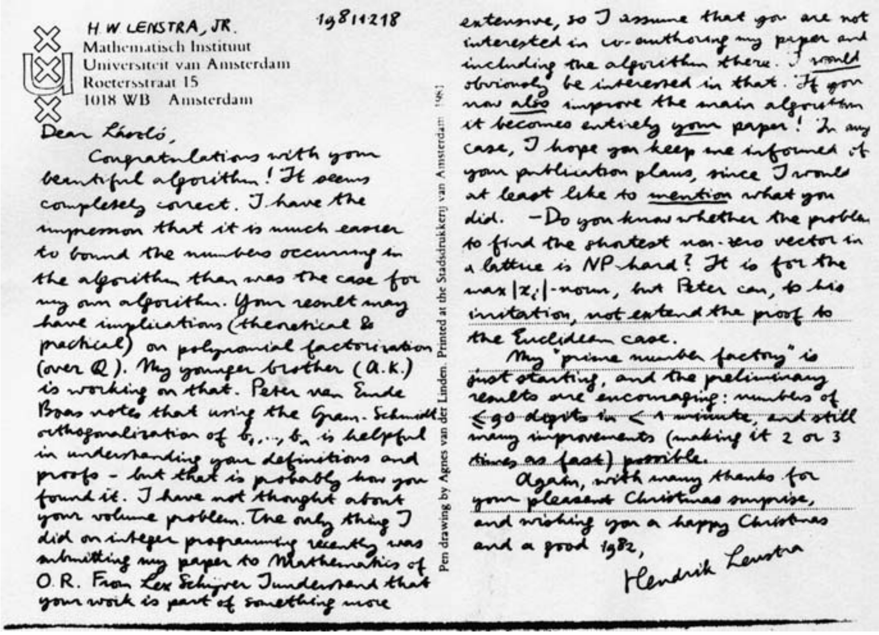
$$\left|\lambda^i_{ik}\right| \le \frac{1}{2} \quad \text{(only the last coefficient!).}$$

We say that $(b_1, \dots, b_n)$ is *weakly greedy* if

$$(b_1 \wedge b_2 \wedge \cdots \wedge b_i \wedge b_{i+2})^2 \ge \frac{3}{4}(b_1 \wedge b_2 \wedge \cdots \wedge b_i \wedge b_{i+1})^2 \tag{1.1}$$

holds for every $0 \le i \le n-2$, where

$$(b_1 \wedge \cdots \wedge b_k)^2 = \det\left((b_i^T b_j)_{i,j=1}^k\right).$$



H. W. LENSTRA, JR. 19811218
Mathematisch Instituut
Universiteit van Amsterdam
Roetersstraat 15
1018 WB Amsterdam

Dear László,

Congratulations with your beautiful algorithm! It seems completely correct. I have the impression that it is much easier to bound the numbers occurring in the algorithm than was the case for my own algorithm. Your result may have implications (theoretical & practical) on polynomial factorization (over $\mathbb{Q}$). My younger brother (A.K.) is working on that. Peter van Emde Boas notes that using the Gram-Schmidt orthogonalization of $b_1, \ldots, b_n$ is helpful in understanding your definitions and proofs - but that is probably how you found it. I have not thought about your volume problem. The only thing I did on integer programming recently was submitting my paper to Mathematics of O.R. From Lex Schrijver I understand that your work is part of something more extensive, so I assume that you are not interested in co-authoring my paper and including the algorithm there. I would obviously be interested in that. If you now also improve the main algorithm it becomes entirely your paper! In any case, I hope you keep me informed of your publication plans, since I would at least like to mention what you did. —Do you know whether the problem to find the shortest non-zero vector in a lattice is NP-hard? It is for the max$|x_i|$-norm, but Peter can, to his irritation, not extend the proof to the Euclidean case.

My "prime number factory" is just starting, and the preliminary results are encouraging: numbers of $\leq 90$ digits in $< 1$ minute, and still many improvements (making it 2 or 3 times as fast) possible.

Again, with many thanks for your pleasant Christmas surprise, and wishing you a happy Christmas and a good 1982,

Hendrik Lenstra

Pen drawing by Agnes van der Linden. Printed at the Stadsdrukkerij van Amsterdam 1981

**Fig. 1.7** The postcard that Hendrik Lenstra sent to László Lovász on 18th December 1981. Notice that the shortest vector problem he mentions is still open after 25 years

Lovász wrote:

> Thus the basis algorithm is the following: start with a basis of the lattice. Look for an $i$, $0 \leq i \leq n-2$ violating (1.1). If such an $i$ exists, interchange $b_{i+1}$ and $b_{i+2}$, and look for another $i$. If no such $i$ exists, straighten out the basis and start all over again. Stop if no exchange has been made after straightening.

A few days after Lenstra got the letter from Lovász, he sent an excited postcard to Hungary, see Fig. 1.7: "Dear László, Congratulations with your beautiful algorithm! [...] Your result may have implications (theoretical & practical) on polynomial factorization (over $\mathbb{Q}$). My younger brother (A.K.) is working on that." More on this polynomial factorization is in section "Polynomial Factorization". First Lovász explains why he was working on lattice basis reduction.

## The Ellipsoid Method

László Lovász started his talk in Caen by declaring that he was not really interested in trying to improve Lenstra's algorithm. In fact, he was interested in a tiny little detail in the ellipsoid method. It all started around 1978 with the paper

**Fig. 1.8** Hendrik Lenstra, László Lovász, and their host Bernhard Korte in Bonn (February 1982)

*A polynomial algorithm in linear programming* from Leonid Khachiyan (sometimes spelled Hačijan) [2]. Lovász: "The ellipsoid method was developed by Soviet scientists in the second half of the seventies. Khachiyan noticed that this algorithm can be applied to solve linear programming in polynomial time, which was a big unsolved problem. All of a sudden there was a big interest in these things."

Peter van Emde Boas remembers how the ellipsoid method first arrived in the west as a rumor and how "Khachiyan conquered the world and everyone became crazy." In those days, there was no email or internet and the iron curtain made things even more difficult.

Lovász:

> I was living in Hungary, but I had the possibility to travel every now and then. In 1978–1979, I spent a year in Canada and in the summer I was in Stanford. There I met Peter Gács and someone sent us Khachiyan's paper. We read and understood it. On the way back to Hungary, I took the opportunity to visit Amsterdam and Bonn. You tried to minimize the number of times you passed the iron curtain, because that was somehow limited. In Amsterdam I met Lex Schrijver and in Bonn Martin Grötschel. I told them both about the ellipsoid method and they became very enthusiastic about it and we started working on it.

Lovász and Gács wrote a report [7] about Khachiyan's paper that explained the ideas and convinced the operations research and computer science communities that the algorithm was correct.

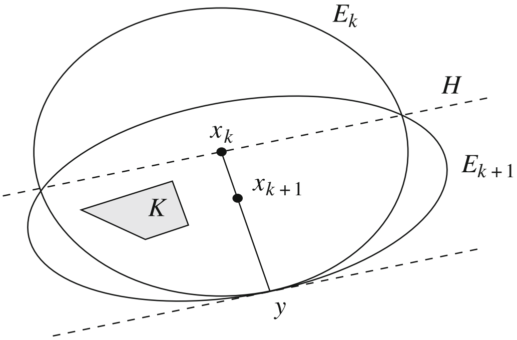


**Fig. 1.9** An illustration of one step of the ellipsoid method

**The Ellipsoid Method (As Described in [1]).**
There is a simple geometric idea behind the ellipsoid method. We start with a convex body $K$ in $R^n$, included in a big ellipsoid $E_0$, and a linear objective function $c^T x$. In the $k$th step, there is an ellipsoid $E_k$, which includes the set $K_k$ of those points $x$ of $K$ for which $c^T x$ is at least as large as the best found so far. We look at the center $x_k$ of $E_k$.
If $x_k$ is not an element of $K$, then we take a hyperplane through $x_k$ which avoids $K$. This hyperplane $H$ cuts $E_k$ into two halves; we pick that one which includes $K_k$ and include it in a new ellipsoid $E_{k+1}$, which is essentially the ellipsoid of least volume containing this half of $E_k$, except for an allowance for rounding errors. The ellipsoid $E_{k+1}$ can be geometrically described as follows. Let $F = E_k \cap H$, and let $y$ be the point where a hyperplane parallel to $H$ touches our half of $E_k$. Then the center of this smallest ellipsoid divides the segment $x_k y$ in ratio $1{:}n$, the ellipsoid intersects $H$ in $F$, and touches $E_k$ in $y$. The ellipsoid $E_{k+1}$ then arises by blowing up and rounding; see Fig. 1.9.
If $x_k \in K$, then we cut with the hyperplane $c^T x = c^T x_k$ similarly.
The volumes of the ellipsoids $E_k$ will tend to 0 exponentially fast and this guarantees that those centers $x_k$ which are in $K$ will tend to an optimum solution exponentially fast.

Consider the following problems for $K$, a nonempty convex compact set in $\mathbb{R}^n$.

1. *Strong optimization problem*: given a vector $c \in \mathbb{R}^n$, find a vector $x$ in $K$ which maximizes $c^T x$ on $K$.
2. *Strong separation problem*: given a vector $y \in \mathbb{R}^n$, decide if $y \in K$, and if not, find a hyperplane that separates $y$ from $K$; more exactly find a vector $c \in \mathbb{R}^n$ such that $c^T y > \max\{c^T x | x \in K\}$.

In 1980, Grötschel, Lovász, and Schrijver proved the following theorem [1].

**Theorem 1** *Let $\mathscr{K}$ be a class of convex bodies. There is a polynomial algorithm to solve the separation problem for the members of $\mathscr{K}$ if and only if there is a polynomial algorithm to solve the optimization problem for the members of $\mathscr{K}$.*

The proof uses the ellipsoid method. Lovász:

> Other people also noticed that the main interest of the ellipsoid method is not in practical applications for linear programming, but in theoretical applications for combinatorial optimization problems. We decided to write a book about this. For this book we wanted to make everything as nice as possible, but there was one annoying little gap.

In combinatorial applications, $K$ is typically given by a system of linear inequalities, with rational coefficients, such that each defining inequality can be written down using a polynomial number of digits. We want to know whether the ellipsoid method terminates. If the solution set $K$ is full-dimensional, then $\mathrm{vol}(K) > 0$ and one can prove that $\log(1/\mathrm{vol}(K))$ is bounded by a polynomial in the dimension $n$ and the length of the rest of input for $K$. So the ellipsoid method terminates after a polynomial number of steps in this case. If $K$ is not full-dimensional (so $\mathrm{vol}(K) = 0$), the ellipsoid method may go on forever. In many interesting applications, it is impossible to tell from the input of $K$ whether $\mathrm{vol}(K) = 0$, but luckily we can determine that this must be the case if the ellipsoids become smaller than the computable lower bound for $\mathrm{vol}(K)$. In this case we can use diophantine rounding as follows.

If $\mathrm{vol}(K) = 0$, then $K$ lies in a hyperplane, and one would like to do the ellipsoid algorithm in dimension $n - 1$. For this, one needs to find a hyperplane containing $K$. If we do the ellipsoid algorithm in dimension $n$, we get smaller and smaller ellipsoids that may never have their center in $K$. After some steps, we do find a hyperplane that approximates $K$, see Fig. 1.10. All vertices of $K$ are close to this hyperplane given by the equality

$$\alpha_1 x_1 + \cdots + \alpha_n x_n = \alpha_0.$$

We want to round this to a hyperplane containing all the vertices of $K$

$$\frac{p_1}{q} x_1 + \cdots + \frac{p_n}{q} x_n = \frac{p_0}{q}.$$

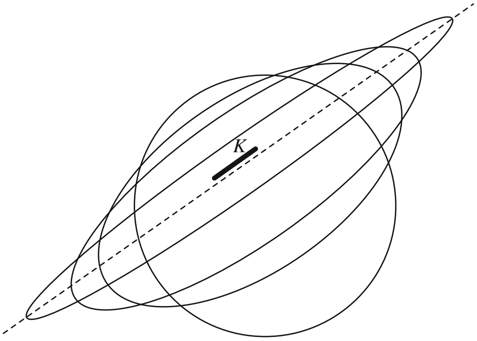


**Fig. 1.10** Some of the (nonconsecutive) ellipsoids found in the ellipsoid method. The hyperplane returned by the ellipsoid method approximates $K$

To make this rounding work, we need the following condition

$$\left|\alpha_i - \frac{p_i}{q}\right| \leq \frac{\varepsilon}{q}$$

for some $\varepsilon$ that can be computed from the problem. This is classic simultaneous Diophantine approximation. The question for Lovász was how to do this algorithmically.

> I started to play around with 1, $\sqrt{2}$, and $\sqrt{3}$ on my TI59 calculator. It was very easy to come up with ideas, it was clear that you wanted to subtract integer multiples of these numbers from each other. Whatever rule I chose, things started nicely, but after some point the process slowed down. I played around for a fairly long time until I found a way that did not slow down and seemed to make steady progress. This was of course just experimentation.
>
> I recalled that Diophantine approximation is discussed in the context of lattices and I realized that the real issue is trying to find short vectors in lattices. I remembered that when I was in Bonn six months earlier, Hendrik Lenstra gave a lecture about integer programming in which he also talked about finding short vectors in lattices. So this was really the way to go.

It took Lovász quite some time to generalize his rule for 1, $\sqrt{2}$, and $\sqrt{3}$ to higher dimensions.“It seemed that the less greedy you were, the better it worked. So I swapped only neighboring vectors and swapped only when you really made progress by a constant factor. And then I sent my letter to Hendrik.”

Hendrik Lenstra emphasized in his talk why these rules make LLL fast:

> Consider the sublattices $L_j$ spanned by the first $j$ basisvectors, $L_j = \mathbb{Z}b_1 + \cdots + \mathbb{Z}b_j$. It is really through these sublattices that you see the progress that you are making in your algorithm. In the LLL-algorithm, you only swap neighboring vectors $b_i$ and $b_{i+1}$, so only $L_i$ changes and all $L_j$ with $j \neq i$ remain the same. Throughout the entire process, none of the determinants $d(L_j)$ ever gets larger.
>
> In my original algorithm I was too greedy. If there was at some stage a very short vector at the end of the basis, I would immediately swap it up front. This makes $L_1$ better, but all the intermediate $L_j$ with $1 < j < n$ may become worse and you lose all control.

## Polynomial Factorization

Arjen Lenstra’s connection with the LLL-algorithm began while he still was a student. He opened his talk in Caen with: “My perspective is more or less that of a surprised bystander while all this violence was going on around me.” It started with a report from Hendrik Lenstra on Euclidean number fields of large degree [6]. This report from 1976 contained a large number of tables of irreducible monic polynomials over the integers. The algebraic number fields generated by the roots of these polynomials were not isomorphic. The question was if other polynomials generated the same number fields as the polynomials in the table. In those days, to answer such a question, you had to factor polynomials over number fields. For the course *Programming Methods*, Arjen Lenstra and fellow students Henk Bos and Rudolf Mak

$$\begin{array}{ccc}
\delta & \delta+1,\ \ 1,\ \ 1 & 5 \\
0 & \left\{\begin{array}{ccccccccc} 1, & 0, & 0, & 0, & 1, & -3, & 3, & -2, & 1 \\ 1 & -2 & 2 & 1 & -4 & 3 & 1 & -2 & 1 \end{array}\right. & 5 \\
\varepsilon & 1,\ \ \varepsilon,\ \ 1 & 5
\end{array} \tag{1.2}$$

**Fig. 1.11** In the tables of Arjen Lenstra's copy of the report on Euclidean number fields [6] there were polynomials pencilled in. The question was if these pencilled-in polynomials generated the same number fields as the ones above them

set out to study and implement methods to factor univariate polynomials over algebraic number fields. This was usually done using the Berlekamp-Hensel approach suggested by Zassenhaus [10].

**Polynomial Factorization for $f \in \mathbb{Q}[X]$ with Berlekamp-Hensel.**
We may assume that $f$ is square-free, as it is easy to remove repeated factors of $f$.

1. Take a prime $p$ such that $f$ modulo $p$ remains square-free and use Berlekamp's algorithm to factor $f$ modulo $p$.
2. Apply the Hensel-lemma to lift these factors to factors modulo $p^k$ for a sufficiently large $k$.
3. Try products of the modular factors to find the "true" factorization.

The big problem with this Berlekamp-Hensel approach was that the last step could be exponential in the degree of $f$, as there are irreducible polynomials that split into many factors modulo any prime. Arjen Lenstra: "No one tried to do anything about this exponential step, all people tried to do was convince themselves that it was indeed very, very much exponential. They were busy generating polynomials that were extremely bad cases for this Berlekamp-Hensel approach."

Generalizing this approach from the rationals to algebraic number fields was according to Arjen Lenstra: "sticking your head in the sand and hoping that it would work."

**Polynomial Factorization for $f \in \mathbb{Q}(\alpha)[X]$ with the Zassenhaus Approach as Described in [9].**
Let $g$ be a monic irreducible polynomial of degree $d$ over the integers, let $g(\alpha) = 0$, and let $f$ be a square-free polynomial to be factored over $\mathbb{Q}(\alpha)$.

1. If there is a prime $p$ such that $g$ modulo $p$ is irreducible and $f$ modulo $p$ is square-free
   (a) Factor $f$ over the finite field $(\mathbb{Z}/p\mathbb{Z})[X]/(g(X))$; the resulting factorization modulo $g$ and $p$ corresponds to the factorization of $f$ over $\mathbb{Q}(\alpha)$.
   (b) Follow the usual Berlekamp-Hensel method.

2. If there is no prime $p$ such that $g$ modulo $p$ is irreducible, then take a prime $p$ with $\gcd(p, \Delta(f)) = 1$ and $\gcd(p, \Delta(g)) = 1$.
   (a) Factor $f$ over several finite fields, one for each irreducible factor of $g$ modulo $p$.
   (b) Lift the factors of $g$ modulo $p$ to factors of $g$ modulo $p^k$ for a sufficiently large $k$.
   (c) Working modulo $p^k$, lift the factors of $f$ from step 2a to factors modulo the lifted factors of $g$ from step 2b.
   (d) Use Chinese remaindering to combine the resulting modular factorizations of $f$ to factors of $f$ modulo a high power of $p$.
   (e) Try combinations of these factors.

Notice that this algorithm is exponential in the product of the degrees of $g$ and $f$. The students got around halfway implementing this approach in their project. Peter van Emde Boas was one of the supervisors of this project, and when he later became Arjen's master thesis advisor, he decided that completing this project would be the perfect task for Arjen.

There were many problems. One of them was that they had to use the fairly newly developed programming language ALGOL68. Lenstra: "It was developed in

**Fig. 1.12** Arjen Lenstra defending his master thesis on 10th December 1980 – 1 year and 2 days before Lóvász posted the letter with his basis reduction algorithm. The committee was formed by Th. J. Dekker, Peter van Emde Boas, and Hendrik Lenstra (behind the desk). Also visible is Pia Pfluger from the Numerical Mathematics group

Amsterdam and if we did not use it, no-one would use it. It worked great, but it had a rather impossible two-level grammar and a seven-pass compiler. The ALGOL68 punch-card jobs had very long turnaround times, rebooting took hours, and we were only allowed limited time on the computers." Another obvious problem was that the algorithms could be exponential, but in practice they often worked. Arjen Lenstra: "I managed to answer the isomorphism questions and thus to complete my master thesis, but it was a rather unsatisfactory method.

When I discussed this with Hendrik, he asked why we used this silly Chinese remaindering and why we combined all those primes in the number field. He suggested that it might be possible to replace the Chinese remaindering by a lattice step." To explain how this lattice step works we assume without loss of generality that the minimum polynomial $g$ has a monic linear factor $h_k$ modulo some power $p^k$ of $p$. Furthermore, let $c$ in $\mathbb{Z}[\alpha]$ be a coefficient of a factor of $f$ over $\mathbb{Q}(\alpha)$. There is an integer $\ell$ and a polynomial $t$ of degree at most $d-1$ in $\mathbb{Z}[\alpha]$ such that

$$c = c_k + \ell \cdot p^k + t \cdot h_k, \tag{1.3}$$

where $c_k$ is an integer value that will appear as one of the coefficients of the (combinations of) factors of $f$ modulo $h_k$ and $p^k$.

From (1.3) it follows that we should consider the $d$-dimensional lattice spanned by the vectors

$$\begin{array}{llllll}
(p^k, & 0, & 0, & \ldots 0, & & 0), \\
(h_{k0}, & 1, & 0, & \ldots 0, & & 0), \\
(0, & h_{k0}, & 1, & 0, & \ldots & 0), \\
 & & \vdots & & & \\
(0, & 0, & \ldots 0, & h_{k0}, & & 1),
\end{array}$$

where $h_k = \alpha + h_{k0}$. One may expect that for large enough $k$, the coefficient $c$ will be the unique shortest vector that is congruent to $c_k$ modulo the lattice as generated above. If we reduce the lattice basis, we find a fundamental domain and when $k$ tends to infinity this domain should spread in all directions to make sure $c$ is contained in it.

Arjen: "I used the lattice basis reduction from Hendrik's paper on integer linear programming [5]. This reduction algorithm did not run in polynomial time, but who cares about such petty issues when dealing with an algorithm that runs in exponential time anyhow? So, the lattice approach was implemented, and it turned out to work beautifully."

The next goal was to prove that the lattice approach always works as expected, including an estimate what value of $k$ one should use to be able to derive valid irreducibility results. Arjen Lenstra:

> I started to think about this and I was not very good at these things. My lack of understanding of the situation reached its zenith when, in my confusion, I added an extra vector and used a $d+1$–dimensional lattice instead of the normal $d$–dimensional one. I was trying to prove that every vector in my lattice was very long, but this $d+1$–dimensional lattice always contained a short vector: $g$ itself. This observation baffled me for a while, but then quickly led to the desired result: apparently the property I needed was coprimality with $g$ over the

```
    $b_i^* := b_i$;                                                          }
    $\mu_{ij} := (b_i, b_j^*)/B_j$;     }                                    }
    $b_i^* := b_i^* - \mu_{ij} b_j^*$   }  for  $j = 1, 2, \dots, i-1$;      }  for $i = 1, 2, \dots, n$;
    $B_i := (b_i^*, b_i^*)$                                                  }
    $k := 2$
(1) perform $(*)$ for $l = k-1$;
    if  $B_k < \left(\frac{3}{4} - \mu_{kk-1}^2\right) B_{k-1}$,  go to (2);
    perform $(*)$ for $l = k-2, k-3, \dots, 1$;
    if  $k = n$, terminate;
    $k := k+1$;
    go to (1);
(2) $\mu := \mu_{kk-1}$; $B := B_k + \mu^2 B_{k-1}$; $\mu_{kk-1} : -\mu B_{k-1}/B$;
    $B_k := B_{k-1} B_k / B$; $B_{k-1} := B$;
    $\begin{pmatrix} b_{k-1} \\ b_k \end{pmatrix} := \begin{pmatrix} b_k \\ b_{k-1} \end{pmatrix}$;
    $\begin{pmatrix} \mu_{k-1j} \\ \mu_{kj} \end{pmatrix} := \begin{pmatrix} \mu_{kj} \\ \mu_{k-1j} \end{pmatrix}$  for  $j = 1, 2, \dots, k-2$;
    $\begin{pmatrix} \mu_{ik-1} \\ \mu_{ik} \end{pmatrix} := \begin{pmatrix} 1 & \mu_{kk-1} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & -\mu \end{pmatrix} \begin{pmatrix} \mu_{ik-1} \\ \mu_{ik} \end{pmatrix}$  for  $i = k+1, k+2, \dots, n$;
    if  $k > 2$,  then  $k := k-1$;
    go to (1).
(*) If $|\mu_{kl}| > \frac{1}{2}$,  then:
    { $r :=$ integer nearest to $\mu_{kl}$; $b_k := b_k - r b_l$;
    { $\mu_{kj} := \mu_{kj} - r\mu_{lj}$  for  $j = 1, 2, \dots, l-1$;
    { $\mu_{kl} := \mu_{kl} - r$.
```

**Fig. 1.13** The basis reduction algorithm as published in the LLL-article. The figure containing pseudocode for the algorithm was added after a suggestion by the referee

> integers, yet a factor $h_k$ in common with $g$ modulo $p^k$. This property I could then indeed use to derive the lower bound proof – a very inelegant proof that is now happily lost in oblivion. In any case, I now knew for sure that we could factor polynomials over algebraic number fields faster than before. How much faster precisely no one seemed to care, as the overall algorithm was still exponential in the degree of $f$.
>
> The initially disturbing observation had an interesting side-result, namely that if we do the entire method for a polynomial $g$ that is *not* irreducible and use the $d$-dimensional lattice, we find a factor of $g$. This implied that if one lifts far enough, the combinatorial search in Berlekamp-Hensel can be avoided at the cost of shortest vector computations in various lattices. Furthermore, by pushing $k$ even further, the shortest vector computations can be replaced by lattice basis reductions. Cute, but useless, as neither the shortest vector nor lattice basis reduction methods ran in polynomial time.

When Lovász sent his letter that lattice basis reduction could be done in polynomial time, Hendrik Lenstra started to look for an error in the proof that the factorization algorithm ran in polynomial time. A few days after he mailed his postcard to Lovász (see Fig. 1.7), Hendrik Lenstra sent a much longer letter, starting: "Ever since I got your letter I have been in a state of surprise, since it seems that your basis reduction algorithm implies that there is a polynomial algorithm for factorization in $\mathbb{Q}[X]$. For several days I have been looking for an error, and not having found one I am writing for your opinion." At that time, factoring polynomials over the rationals was so firmly established as something that could not be done in polynomial time, that something else must be spoiling their factorization algorithm. For a moment Hendrik Lenstra believed he found the wrongdoer in the prime $p$ you needed to maintain square-freeness. However, he proved that this $p$ can be bounded in such a way that Berlekamp runs in polynomial time, deterministically. And so, as Arjen Lenstra put it: "We were suddenly looking at this weird result that polynomials could be factored in polynomial time."

## The LLL-Article

On 12th May 1982, after five months of polishing the algorithm, refining the analysis and many letters to-and-fro, Hendrik Lenstra wrote to Lovász: "Perhaps we should start thinking about where to send our paper. I am personally inclined to send it to a pure mathematics journal rather than a computer science journal. This maximizes the probability of getting sensible comments from the referee. [...] What do you think of Mathematische Annalen?" Lenstra admitted in Caen that there was another reason he wanted to send the article to a pure mathematics journal: "In those days pure mathematicians were not used to doing complexity analyses of algorithms, it was considered the domain of computer scientists. I felt this was a beautiful area that – in this case – gave rise to fantastical problems in number theory and that mathematicians should be more aware of this field. This seemed a good opportunity, as we had a pretty amazing result that nobody had expected."

The unexpected result of polynomial factorization became the title of the paper. Peter van Emde Boas asked the audience in Caen what they thought of when they heard *LLL-algorithm*: was it "basis reduction" or "factoring polynomials"? All hands rose for "basis reduction." So in hindsight maybe the title should have been something like "A new basis reduction algorithm."

On 2nd July 1982, Hendrik Lenstra submitted the article to Mathematische Annalen. The article went rather swiftly through the refereeing process and appeared later that year [8]. The algorithm has made a great impact. In September 2007, the article has 486 citations on ISI Web of Knowledge. As you can see in the rest of this book, research on the LLL-algorithm and its applications are very much alive.

## References

1. M. Grötschel, L. Lovász, and A. Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1(2): 169–197, 1981.
2. L. G. Hačijan. A polynomial algorithm in linear programming. *Dokl. Akad. Nauk SSSR*, 244(5): 1093–1096, 1979.

3. H. W. Lenstra, Jr. Integer programming with a fixed number of variables. *Report 81-03 (First version)*, University of Amsterdam, April 1981.
4. H. W. Lenstra, Jr. Integer programming with a fixed number of variables. *Report 81-03 (Second version)*, University of Amsterdam, November 1981.
5. H. W. Lenstra, Jr. Integer programming with a fixed number of variables. *Math. Oper. Res.*, 8(4): 538–548, 1983.
6. H. W. Lenstra, Jr. Euclidean number fields of large degree. *Report 76-09*, University of Amsterdam, May 1976.
7. P. Gács, and L. Lovász. Khachiyan's algorithm for linear programming. *Math. Programming Stud.*, 14: 61–68, 1981.
8. A. K. Lenstra, H. W. Lenstra, Jr., and L. Lovász. Factoring polynomials with rational coefficients. *Math. Ann.*, 261(4): 515–534, 1982.
9. P. J. Weinberger and L. P. Rothschild. Factoring polynomials over algebraic number fields. *ACM Trans. Math. Software*, 2(4): 335–350, 1976.
10. H. Zassenhaus. On Hensel factorization. I. *J. Number Theory*, 1: 291–311, 1969.