Advanced Algorithms	**Lecture 8: April 18–19, 1995**
Lecturer: Johan Håstad	*Scribe: Per Lindberger*

Lovász's Lattice Basis Reduction Algorithm

Suppose we have factorized the polynomial $f \bmod p^k$, so that $f \equiv gh \pmod{p^k}$, where h is irreducible $\pmod p$. We next want to find a multiple $h' = hz \bmod p^k$ of h, such that $\deg h' \leq n-1$ (lest h' be the original f) and such that the coefficients of h' are "small".

Let us start by considering the different possibilities for h' and let us for the time being represent polynomials by the vectors of their coefficients.

Thus,

$$\begin{aligned}
h = \sum_{i=0}^{m} h_i x^i &\sim (\underbrace{h_0, h_1, \ldots, h_m, 0, \ldots, 0}_{n}),\\
xh &\sim (0, h_0, h_1, \ldots, h_m, 0, \ldots, 0),\\
&\vdots\\
x^{n-(m+1)}h &\sim (0, \ldots, 0, h_0, h_1, \ldots, h_m),\\
p^k e_1 &\sim (p^k, 0, \ldots, 0),\\
&\vdots\\
p^k e_i &\sim (\underbrace{0, \ldots, 0}_{i-1}, p^k, 0, \ldots, 0),
\end{aligned}$$

and h' is a linear combination of these vectors in $\mathbb{Z}^n$.

Finding a polynomial with small coefficients then corresponds to finding a short vector and our present problem translates to an instance of the more general problem addressed in this lecture.

8.1 Finding a short vector in a Lattice

Given vectors $b_1, b_2, \ldots, b_n$, we wish to study all vectors in the set

$$L = \{\sum_{i=1}^{n} a_i b_i \mid a_i \in \mathbb{Z}\},$$

and find one that isn't too long. If the b_i form a basis of $\mathbb{R}^n$, $L \subset \mathbb{R}^n$ is called a *lattice*. For the moment, we assume that $b_1, b_2, \ldots, b_n \in \mathbb{Z}^n$ are linearly independent and so form a basis.

Example 1 *If $n = 2, b_1 = (1,0), b_2 = (0,1)$, then $L = \{(a,b) \mid a, b \in \mathbb{Z}\}$, which is the set of all integral points in the plane.*

Example 2 *If instead we let $b_1 = (2,0)$, then L will be all integral points whose first coordinate is even.*

Example 3 *Let $n = 2, b_1 = (421, 114)$ and $b_2 = (469, 127)$. We can reduce this basis, while keeping the generated lattice intact, by the sequence of elementary operations*

$$\begin{aligned} b_2 &\leftarrow b_2 - b_1 = (48, 13) \\ b_1 &\leftarrow b_1 - 9b_2 = (-11, -3) \\ b_2 &\leftarrow b_2 + 4b_1 = (4, 1) \\ b_1 &\leftarrow b_1 + 3b_2 = (1, 0) \\ b_2 &\leftarrow b_2 - 4b_1 = (0, 1). \end{aligned}$$

This makes it plain that the basis $(421, 114), (469, 127)$ generates the same points as the simpler-looking $(1,0), (0,1)$.

Let us formulate an algorithm that for the case $n = 2$ finds a "good" basis, e g one with short and comparatively orthogonal vectors. The experience from example 3 makes the following a natural suggestion:

```
GAUSS(b_1, b_2)
repeat
    arrange that b_1 is the shorter vector
    find k ∈ Z such that b_2 − k b_1 is of minimal length
    b_2 ← b_2 − k b_1
until k = 0
return b_1, b_2
```

And indeed, the output from GAUSS will be such that b_1 is the shortest vector in the lattice and b_2 is roughly orthogonal to b_1. A simple geometric argument reveals that the angle between the output vectors is in the range $90° \pm 30°$.

Example 4 *Now, let* $n = 3, p = 5, k = 3$ *and* $h \equiv x + 57 \pmod{p^k}$. *Then,*

$$\begin{aligned} h &\sim (57, 1, 0) = b_1, \\ xh &\sim (0, 57, 1) = b_2, \\ p^k e_1 &\sim (125, 0, 0) = b_3 \end{aligned}$$

form a basis. After the operations

$$\begin{aligned} b_3 &\leftarrow b_3 - 2b_1 = (11, -2, 0), \\ b_1 &\leftarrow b_1 - 5b_3 = (2, 11, 0), \\ b_2 &\leftarrow b_2 - 5b_1 = (-10, 2, 1) \end{aligned}$$

we have $b_2 + b_3 = (1, 0, 1) \sim x^2 + 1$, *which certainly is a multiple of* $h \bmod p^k$ *whose coefficients are conveniently small. Recall that* $(x + 57)(x - 57) \equiv x^2 + 1 \pmod{125}$.

Seeing that the naïve extension of GAUSS to three dimensions

```
repeat
   GAUSS(b1, b2)
   GAUSS(b2, b3)
   ⋮
until ...
```

will not work for degenerate cases, where there are no sufficiently parallel vectors, we consider projecting out offending coordinates and arrive at the following algorithm.

```
REDUCE(b_1, b_2, b_3)
repeat
    GAUSS(b_1, b_2)
    b_2^* ← b_2 projected orthogonally to b_1
    b_3^* ← b_3 projected orthogonally to b_1
    GAUSS(b_2^*, b_3^*)
    find k_2 ∈ Z such that b_2 − k_2 b_1 is of minimal length
    b_2 ← b_2 − k_2 b_1
    find k_3 ∈ Z such that b_3 − k_3 b_1 is of minimal length
    b_3 ← b_3 − k_3 b_1
    arrange that b_1 is the shortest vector
until k_2 = 0 and k_3 = 0
return (b_1, b_2, b_3)
```

In the next section we will see how to generalize basis reduction to n dimensions. Before then, let's make a general observation.

The basis changes we make are all of the form $b_i \leftarrow b_i - \sum_{j \neq i} c_j b_j$, where $c_j \in \mathbb{Z}$, and do not change the lattice. The *determinant* of the lattice we define by

$$\det(L) = |\det \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}|.$$

This is the volume of the parallelepiped spanned by the basis vectors, or, if you will, the reciprocal of the "density" of L, a number that does not depend on the particular choice of basis. This invariant we will need in the analysis of the general algorithm.

8.2 General Basis Reduction

To begin generalizing to arbitrary dimension, consider the following. Assume we have a basis $b_1, b_2, \ldots, b_n$ to reduce. Define

$$\begin{aligned} b_1^* &= b_1, \\ b_i^* &= b_i \text{ projected orthogonally to } b_1, \ldots, b_{i-1}, 2 \leq i \leq n, \\ e_i^* &= \text{ unit vector in the direction of } b_i^*, \\ \beta_i &= |b_i^*|. \end{aligned}$$

What we have done is called a *Gram-Schmidt orthogonalization* and in the orthogonal basis e^* we have

$$\begin{aligned} b_1 &= (\beta_1, 0, \ldots, 0), \\ b_2 &= (\mu_{12}, \beta_2, 0, \ldots, 0), \\ &\vdots \\ b_n &= (\mu_{1n}, \mu_{2n}, \ldots, \mu_{(n-1)n}, \beta_n). \end{aligned}$$

Note that no vector in the lattice is shorter than $\min_{i=1}^n \beta_i$, because they all are of the form $\sum_{i=1}^n a_i b_i$, where $a_i \in \mathbb{Z}$, so the last non-zero component will always be a multiple of some β_i.

We are now ready for the algorithm L^3, so called because it first appeared in a paper by Lenstra, Lenstra and Lovász [1], but it is likely due to the last author alone. Bearing in mind that we want to single out b_1 as our "short" vector, i e that what we desire is that upon termination there will be no i such that $\beta_i \ll \beta_1$, we state the algorithm and then analyze it.

Input: a basis and its orthogonalization
Output: a reduced basis

(1) arrange that $|\mu_{ij}| \leq \frac{1}{2}\beta_i$, **for** $1 \leq i \leq n-1, 2 \leq j \leq n$
(2) **if** $\exists i : \beta_{i+1}^2 \leq \frac{1}{2}\beta_i^2$
(3) exchange b_i and b_{i+1}
(4) **goto** 1

Our analysis will have to answer two questions:

1. How good is the result, i e how much longer than $\min_{i=1}^n \beta_i$ can b_1 be upon termination?
2. How many times will we iterate in the worst case and how long does an iteration take?

Upon termination it will be the case that

$$\beta_1^2 \leq 2\beta_2^2 \leq \ldots \leq 2^{n-1}\beta_n^2,$$

so

$$|b_1| = \beta_1 \leq 2^{\frac{n-1}{2}} \min_{i=1}^n \beta_i \leq 2^{\frac{n-1}{2}} |\text{the shortest vector in L}|.$$

This answers the first question. For the second, first note that what takes time in an iteration is the first step. In principle being a Gaussian elimination it takes time $O(n^3)$. Next, suppose we exchange b_2 and b_3. After the exchange, we have $\beta_2' = |b_2^{*\prime}| =$ the length of the part of b_2' that is orthogonal to b_1, but $b_2' = b_3$, so

$$\beta_2' = \sqrt{\beta_3^2 + \mu_{23}^2} \leq \sqrt{\frac{1}{2}\beta_2^2 + \frac{1}{4}\beta_2^2} = \sqrt{\frac{3}{4}}\beta_2.$$

For $j \geq 4$, we have $\beta_j' = \beta_j$ and since $\det(L) = \prod_{i=1}^n \beta_i = \prod_{i=1}^n \beta_i'$, it follows that

$$\beta_3' = \frac{\beta_2\beta_3}{\beta_2'} \geq \sqrt{\frac{4}{3}}\beta_3.$$

So we conclude that an exchange at i only affects β_i and β_{i+1}, the first of which decreases and the latter increases. Furthermore, we have lowerbounded their rate of change. Consider how an exchange affects the product $B = \prod_{j=1}^n \beta_j^{n-j}$. The only interesting part is

$$\beta_i^{n-i}\beta_{i+1}^{n-i-1} = \beta_i \underbrace{(\beta_i\beta_{i+1})^{n-i-1}}_{\text{unchanged}},$$

so B decreases by the same factor as does the particular β_i in each iteration.

If the longest input vector has length D, then $\beta_j \leq D$, which implies that to begin with

$$B \leq \prod_{j=1}^n D^{n-j} = D^{\frac{n(n-1)}{2}},$$

and in the end $B \geq 1$, because

$$\begin{aligned}
\beta_1 &\geq 1 \text{ (length of } b_1\text{)} \\
\beta_1\beta_2 &\geq 1 \left(\det\left(\begin{pmatrix} b_1 \\ b_2 \end{pmatrix}\begin{pmatrix} b_1^T & b_2^T \end{pmatrix}\right) = \beta_1^2\beta_2^2, \text{ an integer}\right) \\
&\vdots \\
\beta_1\beta_2\cdots\beta_j &\geq 1 \\
&\vdots \\
\beta_1\beta_2\cdots\beta_n = \det(L) &\geq 1.
\end{aligned}$$

Thus we see that the number of iterations is in $O(n^2 \log D)$, and that the total running time of the algorithm is in $O(n^5 \log D)$. This answers the second question and concludes our analysis of L^3. We are ready to apply it to our problem of factorizing rational polynomials.

Reference

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