CS810A/MA810A Computer Algebra

Alex Myasnikov and Werner Backes

Stevens Institute of Technology

CS810A/MA810A 2008
Short vectors in lattices

- Short vectors in lattices
  - Lattices.
  - Lattice basic reduction.
  - Applications.

Modern Computer Algebra (2nd edition), by Joachim von zur Gathen and Jürgen Gerhard, Pages 461 – 489

The slides are not a substitute for reading the book!
Lattices

Definition

Let \( n, k \in \mathbb{N} \) with \( k \leq n \). A lattice \( L \subset \mathbb{R}^n \) is a discrete, additive subgroup of \( \mathbb{R}^n \), such that \( L = \{ \sum_{i=1}^{k} x_i b_i \mid x_i \in \mathbb{Z}, i = 1, \ldots, k \} \), where \( b_1, b_2, \ldots, b_k \in \mathbb{R}^n \) are linearly independent vectors.

We call \( B = (b_1, \ldots, b_k) \in \mathbb{R}^{n \times k} \) a basis of the \( k \)-dimensional lattice \( L \).

- ”discrete subgroup” here means set of points in \( \mathbb{R}^n \).
- ”additive” linear combinations of base vectors.
Lattices

Example:
Applications:

- Cryptography
  - Ajtai-Dwork cryptosystem
  - Knapsack problem
  - GGH cryptosystem
  - NTRU cryptosystem
  - Cryptanalysis
- Integer programming
- Factoring
- Discrete logarithm
- Algorithmic number theory
For a lattice \( L \in \mathbb{R}^n \) with basis \( B \in \mathbb{R}^{n \times k} \) the determinant \( \det(L) \) is defined as follows:

\[
\det(L) = \left| \det(B^T B) \right|^{\frac{1}{2}}
\]

- The determinant of a lattice is an invariant.
- It follows, that the determinant is independent of a particular basis.
- The determinant is also called the norm of lattice \( L \) (book).
- Hadamard’s inequality gives us an upper bound on the determinant of a lattice.
Lattice bases $B$ and $B'$ of lattice $L$ give us a different bound according to Hadamard’s inequality.

The determinant of a lattice $L$ is independent of $B$ and $B'$.

Hadamard’s inequality in combination with $\det(L)$ can be used to define the quality of a particular lattice basis.
Lattice bases for a lattice $L \subset \mathbb{R}^n$ are not unique, but they uniquely defined by unimodular transformations.
Definition

A matrix $U \in \mathbb{Z}^{n \times n}$ is unimodular, iff $|\det U| = 1$.

Allowed operations on a lattice basis:

Given a lattice $L \subset \mathbb{R}^n$ with basis $B = (b_1, \ldots, b_n) \in \mathbb{R}^{n \times k}$.

- $b_i \leftarrow b_i + k \cdot b_j$ with $k \in \mathbb{Z}$
  Add an integer multiple of one basis vector to another basis vector.
- $b_i \leftrightarrow b_j$. Swap two basis vectors.
- $b_i = -b_i$. Negate a basis vector.

Two lattice bases $B$ and $B'$ are equivalent if you can transform $B$ to $B'$ using the above operations.
Lattice basis reduction:
Find a basis $B' = (b'_1, \ldots, b'_k)$ for lattice $L(B)$ with $BU = B'$ ($U$ unimodular) and as short and orthogonal basis vectors as possible.

Other (open) problems:

- Shortest/Closest lattice vector problem.
- Minimum basis problem.
- Membership problem.
- Basis computation problem.
Other (open) problems:
**Definition**

For a $k$-dimensional lattice $L \subset \mathbb{R}^n$ and $i \in \mathbb{N}$ ($1 \leq i \leq k$) let $r \in \mathbb{R}^+$ be the minimal value for which linear independent vectors $v_1, \ldots, v_i$ exists with

$$\|v_j\| \leq r \quad (1 \leq j \leq i).$$

We call $r$ the $i$-th successive minima, denoted by $\lambda_i(L)$.

**Lemma**

For a $k$-dimensional lattice $L \subset \mathbb{R}^n$ the following holds:

$$0 < \lambda_1(L) \leq \lambda_2(L) \leq \cdots \leq \lambda_k(L).$$
Remark

The length of the shortest non-zero lattice vector in a $k$-dimensional lattice $L \subset \mathbb{R}^n$ is equal to the first successive minima $\lambda_1(L)$.

- Finding the shortest vector in a given lattice in NP-hard.
- Known algorithms to compute a shortest vector are exponential.
- Algorithms need a ”good” basis to start with.
- For some application it is sufficient to compute a ”relatively short” basis vector.
Lenstra, Lenstra and Lovász in 1982 developed polynomial time algorithm to find a ”good” basis.
Gram-Schmidt orthogonalization:

\[ b_1^* = b_1 \]
\[ b_i^* = b_i - \sum_{j=1}^{i-1} \mu_{ij} b_j^* \quad \text{for } 2 \leq i \leq k \]

\[ \mu_{ij} = \frac{\langle b_i, b_j^* \rangle}{\|b_j^*\|^2} \quad \text{for } 1 \leq j < i \leq k \]

- \( U_k = \sum_{1\leq i \leq k} \mathbb{R}b_i = \sum_{1\leq i \leq k} \mathbb{R}b_i^* \) is the \( \mathbb{R} \) subspace spanned by \( b_1, \ldots, b_k \).
- \( b_i^* \) is the projection of \( b_i \) onto the orthogonal complement

\[ U_{i-1}^\perp = \{ b \in \mathbb{R}^n : b \cdot u = 0 \text{ for all } u \in U_{i-1} \} \]

of \( U_{i-1} \), and hence \( \|b_i^*\| \leq \|b_i\| \).
- \( b_1^*, \ldots, b_k^* \) are pairwise orthogonal.
Example:
Lattice Basis Reduction

Definition (LLL Reduction)

A basis \( B = (b_1, \ldots, b_k) \) of the lattice \( L \subset \mathbb{R}^n \) is called \textit{LLL-reduced} for reduction parameter \( \frac{1}{4} < y < 1 \) if the following holds:

\[
|\mu_{ij}| \leq \frac{1}{2} \quad \text{for} \ 1 \leq j < i \leq k
\]

\[
\|b_i^* + \mu_{ii-1}b_{i-1}^*\|^2 \geq y\|b_{i-1}^*\|^2 \quad \text{for} \ 1 < i \leq k
\]

- The first condition ensures, that the basis vectors are as orthogonal as possible.
- The second condition ensures reasonably short basis vectors.
Theorem

For a lattice $L \subset \mathbb{R}^n$ with LLL-reduced basis $B = (b_1, \ldots, b_k) \in \mathbb{R}^{n \times k}$ the following holds:

\[
\|b_j\|^2 \leq 2^{i-1}\|b_i\|^2 \quad \text{for } 1 \leq j \leq i \leq k
\]

\[
\|b_1\| \cdots \|b_k\| \leq 2^{\frac{k(k-1)}{4}} \det(L)
\]

\[
\|b_1\| \leq 2^{\frac{k-1}{4}} k^{\frac{1}{2}} \det(L)
\]

\[
\|b_1\|^2 \leq 2^{k-1}\|\mathbf{v}\|^2 \quad \text{for all } \mathbf{v} \in L, \mathbf{v} \neq 0
\]

\[
2^{\frac{1-i}{2}} \lambda_i(L) \leq \|b_i\| \leq 2^{\frac{k-1}{2}} \lambda_i(L)
\]

- Worst case the vectors in a LLL-reduced basis are exponentially far away from the shortest vector of the lattice.
- Much better in practice.
Features:

- Run time: $O(k^3 n \log C)$ arithmetic operations, with $C \in \mathbb{R}, C \geq 2, \|b_i\|^2 \leq C$
- Size of integers: $O(k \log C)$ bit
- Provable bounds

Problems:

- Performance (long integer arithmetic)
- Stability (floating point arithmetic)
LLL-Reduction (variants):

- Benne de Weger algorithm
- MLLL algorithm
- Buchmann-Kessler algorithm
- **Schnorr-Euchner algorithm**

**Schnorr-Euchner algorithm:**

- Floating point approximation for Gram-Schmidt orthogonalization
- Long integer arithmetic for size reduction
- Correction steps to improve stability
- Fast and stable in practice, but termination not proven.
Lattice Basis Reduction – Schnorr-Euchner algorithm

Schorr-Euchner algorithm:

**Input:** Lattice basis $B = (b_1, \ldots, b_k) \in \mathbb{Z}^{n \times k}$, $\delta$

**Output:** LLL-reduced lattice basis

(1) $\text{APPROX\_BASIS}(B', B)$
(2) $B_1 = \|b'_1\|^2$, $i = 2$
(3) $F_c = \text{false}$, $F_r = \text{false}$
(4) while ($i \leq k$) do
(5) $B_i = \|b'_i\|^2$
(6) for ($2 \leq j \leq i$) do
(7) if ($|\langle b'_i, b'_j \rangle| < 2^{\frac{r}{2}} \|b'_i\| \|b'_j\|)$ then
(8) $s = \text{APPROX\_VALUE}(\langle b_i, b_j \rangle)$
(9) else
(10) $s = \langle b'_i, b'_j \rangle$
(11) $\mu_{ij} = (s - \sum_{m=1}^{j-1} \mu_{jm} \mu_{im} B_m) / B_j$
(12) $B_i = B_i - \mu_{ij}^2 B_j$
(13) $\mu_{ii} = 1$
(14) for ($i > j \geq 1$) do
(15) if ($|\mu_{ij}| > \frac{1}{2}$) then
(16) $F_r = \text{true}$
(17) if ($|\|\mu_{ij}\|| > 2^{\frac{r}{2}}$) then
(18) $F_c = \text{true}$
(19) $b_i = b_i - \lceil \mu_{ij} \rceil b_j$
(20) for ($1 \leq m \leq j$) do
(21) $\mu_{im} = \mu_{im} - \lceil \mu_{ij} \rceil \mu_{jm}$
(22) if ($F_r = \text{true}$) then
(23) $\text{APPROX\_VECTOR}(b'_i, b_i)$
(24) $F_r = \text{false}$
(25) if ($F_c = \text{true}$) then
(26) $i = \max\{i - 1, 2\}$
(27) $F_c = \text{false}$
(28) else
(29) if ($B_i < (\delta - \mu_{ii}^2) B_{i-1}$) then
(30) $\text{SWAP}(b_{i-1}, b_i)$
(31) if ($i = 2$) then
(32) $B_1 = \|b'_1\|^2$
(33) $i = \max\{i - 1, 2\}$
(34) else
(35) $i = i + 1$
- Correction steps (exact scalar products and step-backs) affect running time
- Exact scalar products have bigger impact.
Definition (Gram matrix) 

For a lattice $L$ with basis $B = (b_1, \ldots, b_k) \in \mathbb{R}^{n \times k}$, the corresponding Gram matrix $G$ is defined as $G = B^T B$.

- Gram matrix is the matrix of scalar products, therefore symmetric

**Advantage:**
- Avoid expensive exact scalar products.

**Problem:**
- Gram matrix not of interest: update of Gram matrix and lattice basis necessary.
Lattice Basis Reduction – LLL Gram

Developed by Backes, Wetzel in 2006.

**INPUT:** Lattice basis \( B = (b_1, \ldots, b_k) \in \mathbb{Z}^{n \times k}, \delta \)

**OUTPUT:** LLL-reduced lattice basis

1. **COMPUTE** GRAM\((A, B)\)
2. **APPROX** BASIS GRAM\((A', A)\)
3. \( R_{11} = A'_{11}, i = 2 \)
4. \( F_c = false, F_r = false \)
5. while \( (i \leq k) \) do
6. \( S_1 = R_{ii} \)
7. for \( (2 \leq j \leq i) \) do
8. \( R_{ij} = A'_i - \sum_{m=1}^{j-1} R_{im} \mu_{im} \)
9. \( \mu_{ij} = \frac{R_{ij}}{R_{ii}} \)
10. \( R_{ii} = R_{ii} - R_{ij} \mu_{ij} \)
11. \( S_{i+1} = R_{ii} \)
12. \( \mu_{ii} = 1 \)
13. for \( (i > j \geq 1) \) do
14. if \( (|\mu_{i,j}| > \frac{1}{2}) \) then
15. \( F_r = true \)
16. \( b_i = b_i - \lceil \mu_{ij} \rceil b_j \)
17. **REDUCE** GRAM\((A, i, \lceil \mu_{ij} \rceil, j)\)
18. if \( (|\mu_{ij}| > 2^{\frac{k}{2}}) \) then
19. \( F_c = true \)
20. for \( (1 \leq m \leq j) \) do
21. \( \mu_{im} = \mu_{im} - \lceil \mu_{ij} \rceil \mu_{im} \)
22. if \( (F_r = true) \) then
23. **APPROX** VECTOR GRAM\((A', A, i)\)
24. \( F_r = false \)
25. if \( (F_c = true) \) then
26. \( i = max(i - 1, 2) \)
27. \( F_c = false \)
28. else
29. \( i' = i \)
30. while \( ((i > 1) \land (\delta \cdot R_{(i-1)(i-1)} > S_{i-1})) \) do
31. \( b_i \leftrightarrow b_{i-1} \)
32. **SWAP** GRAM\((A, i - 1, i)\)
33. **SWAP** GRAM\((A', i - 1, i)\)
34. \( i = i - 1 \)
35. if \( (i \neq i') \) then
36. if \( (i = 1) \) then
37. \( R_{11} = A'_{11}, i = 2 \)
38. \( i = i + 1 \)
We use a lower triangular matrix to represent the symmetric Gram matrix.

Update operations are therefore more complicated.

REDUCE_GRAM($A, l, \lceil \mu_{ij} \rceil, j$)

**INPUT:** Gram matrix $A$, indices $l, j, \lceil \mu_{ij} \rceil$

**OUTPUT:** Gram matrix $A$

1. $T = A_{l,l} - 2 \cdot \lceil \mu_{ij} \rceil \cdot A_{j,l} - \lceil \mu_{ij} \rceil^2 \cdot A_{j,j}$
2. for ($1 \leq m < j$) do
3. $A_{m,l} = A_{m,l} - \lceil \mu_{ij} \rceil \cdot A_{m,j}$
4. for ($j \leq m < l$) do
5. $A_{m,l} = A_{m,l} - \lceil \mu_{ij} \rceil \cdot A_{j,m}$
6. for ($l + 1 \leq m < k$) do
7. $A_{l,m} = A_{l,m} - \lceil \mu_{ij} \rceil \cdot A_{j,m}$
8. $A_{l,l} = T$

SWAP_GRAM($A, i, j$)

**INPUT:** Gram matrix $A$, indices $i, j$

**OUTPUT:** Gram matrix $A$

1. if ($i > j$) then
2. $i \leftrightarrow j$
3. for ($1 \leq m < j$) do
4. $A_{m,i} \leftrightarrow A_{m,j}$
5. for ($j \leq m < j$) do
6. $A_{m,i} \leftrightarrow A_{j,m}$
7. for ($j \leq m < i$) do
8. $A_{i,m} \leftrightarrow A_{j,m}$
9. $A_{i,i} \leftrightarrow A_{i,j}$
Lattice Basis Reduction – LLL Gram

Buffer transformation to lattice basis using machine integers.

**Algorithm:** BUFFERED\_TRANSFORM\((B, i, \lceil \mu_{ij} \rceil, j)\)

**Input:** Lattice Basis \(B = (b_1, \ldots, b_k) \in \mathbb{Z}^{n \times k}\), indices \(i, j, \lceil \mu_{ij} \rceil\)

**Output:** Lattice Basis \(B'(1)\)

1. if \((\left( T_{\text{max}}^i + \lceil \mu_{ij} \rceil \right) T_{\text{max}}^j) \geq 2^{m-1})\) then
2. for \((\text{pos}_{\text{min}} \leq x \leq \text{pos}_{\text{max}})\) do
3. for \((1 \leq z \leq n)\) do
4. \(B'_{xz} = 0\)
5. for \((\text{pos}_{\text{min}} \leq y \leq \text{pos}_{\text{max}})\) do
6. for \((1 \leq z \leq n)\) do
7. \(B'_{xz} = B'_{xz} + T_{xy} \cdot B_{yz}\)
8. \(B \leftarrow B'\)
9. \(T = I_n\)
10. \(T_{\text{max}} = (1, \ldots, 1)^T\)
11. \(\text{pos}_{\text{max}} = i\)
12. \(\text{pos}_{\text{min}} = j\)

13. if \(\lceil \mu_{ij} \rceil > 2^{m-1} - 1\) then
14. \(b_i = b_i - \lceil \mu_{ij} \rceil \cdot b_j\)
15. return
16. \(t_i = t_i - \lceil \mu_{ij} \rceil \cdot t_j\)
17. \(T_{\text{max}}^i = T_{\text{max}}^i + \lceil \mu_{ij} \rceil \cdot T_{\text{max}}^j\)
18. if \(\text{pos}_{\text{max}} < i\) then
19. \(\text{pos}_{\text{max}} = i\)
20. if \(\text{pos}_{\text{min}} > j\) then
21. \(\text{pos}_{\text{min}} = j\)
Avoid expensive operations (Multiplications).

Vectorization (Multimedia Streaming Extensions).

(7) if \( T_{xy} \neq 0 \) then
(8)   if \( T_{xy} = 1 \) then
(9)     for \( 1 \leq z \leq n \) do
(10)    \( B'_{xz} = B'_{xz} + B_{yz} \)
(11) else
(12)   if \( T_{xy} = -1 \) then
(13)     for \( 1 \leq z \leq n \) do
(14)       \( B'_{xz} = B'_{xz} - B_{yz} \)
(15) else
(16)   for \( 1 \leq z \leq n \) do
(17)     \( B'_{xz} = B'_{xz} + T_{xy} \cdot B_{yz} \)
(18)   \( T_{i,l+1} = T_{i,l+1} - \lceil \mu_{ij} \rceil \cdot T_{j,l} \)
(19)   \( T_{i,l+2} = T_{i,l+2} - \lceil \mu_{ij} \rceil \cdot T_{j,l+1} \)
(20)   \( T_{i,l+3} = T_{i,l+3} - \lceil \mu_{ij} \rceil \cdot T_{j,l+2} \)

...
Lattice Basis Reduction – LLL Gram

- Test System: Sun X4100 Server, AMD Opteron 2.2 GHz and 4GB RAM, Solaris 10
- LLL Gram with and without buffered transformations.
**Lattice Basis Reduction – Parallel LLL**

- **Test System:** Sun X2200 Server, AMD Opteron 2.2 GHz and 4GB RAM, Solaris 10
- **Parallel LLL** developed by Backes, Wetzel in 2008.
Applications

Application

Polynomial Equations

Formulate

Short Vectors

LLL reduction

Shortest vector

Lattice

Generate Lattice basis
Coppersmith’s technique:

- Derived general algorithm for finding small roots of polynomial equations.
- Root finding algorithm essentially based on the LLL-reduction algorithm.
- The key idea is to encode polynomial equations with small solutions as coefficient vectors that have a small Euclidean norm (length).
  \( f(x) = 3x^3 + 2x + 20, \; f = (20, 2, 0, 3) \) corresponding vector
- Small roots can efficiently (polynomial time) be found by applying the LLL-reduction algorithm.
Coppersmith’s technique

- Polynomial equations
  - Modular
    - Univariate
    - Multivariate
  - Integer
    - Bivariate
    - Multivariate
Applications – Coppersmith’s technique

Problem: the univariate case

- Given large integer $N \in \mathbb{N}$, for which factorization is unknown.
- Given a polynomial $f \in \mathbb{Z}[x]$ of degree $d$.
- With a modular equation:

$$f(x) = a_d x^d + a_{d-1} x^{d-1} + \cdots + a_1 x + a_0 \equiv 0 \mod N$$

- The goal is to find $x_0 \in \mathbb{Z}$ such that $f(x_0) \equiv 0 \mod N$.

Solution?

- For the general case there are no known, efficient algorithms.
- Use Coppersmith’s technique to find small roots.
Applications – Coppersmith’s technique

**Definition (euclidean norm)**

For a univariate polynomial $f(x) = \sum_{i=0}^{d} a_i x^i$ we define the euclidean norm of $f$ as

$$\|f\|^2 = \sum_{i=0}^{d} a_i^2.$$ 

**Definition (root container)**

A polynomial $h$ is a root container of a polynomial $f$ if each root of $f$ is also a root of $h$. In case roots are considered modulo $N$, we call $h$ a root container of $f$ modulo $N$. 
Applications – Coppersmith’s technique

Problem: more details

- Recover small modular roots by transforming the modular equation to to an equation over integers.
- Efficiently find all roots \( x_0 \) such that \(|x_0| < X\) for a bound \( X \).
- The goal is to maximize the bound \( X \).

Basic idea:

- Find a polynomial \( h(x) \in \mathbb{Z}[x] \) such that \( h(x_0) = f(x_0) \equiv 0 \mod N \) with small euclidean norm \( \|h\|^2 \).
Question:

- Under which conditions can the modular equation be transformed to an integer equation?
- How can we calculate or find a inequality for the bound $X$.

Lemma (Howgrave-Graham)

Let $h(x)$ be an univariate polynomial with at most $\omega$ monomials. In addition we assume that $h$ satisfies the following two conditions:

1. $h(x_0) \equiv 0 \mod N$ where $|x_0| < X$ and
2. $\|h(xX)\| \leq \frac{N}{\sqrt{\omega}}$

Then $h(x_0) = 0$ holds over integers.
Applications – Coppersmith’s technique

Set of root container polynomials:

\[ Z_1 = \{ g_0(x) = N, g_1(x) = Nx, \ldots, g_{d-1}(x) = Nx^{d-1}, g_d = f(x) \} \]

We consider the lattice \( L_1 \) with basis

\[
B_1 = \begin{pmatrix}
N & 0 & \cdots & 0 & f_0 \\
0 & XN & \cdots & 0 & Xf_1 \\
0 & 0 & \cdots & 0 & X^2f_2 \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
0 & 0 & \cdots & X^{d-1}N & X^{d-1}f_{d-1} \\
0 & 0 & \cdots & 0 & X^d
\end{pmatrix}
\]

Lattice is generated by column vectors in basis \( B_1 \).
Each lattice point in \( L_1 \) corresponds to the coefficient vector of a polynomial \( h(xX) \sum_{i=0}^{d} c_i g_i(xX) \).

From \( f(x_0) \equiv 0 \mod N \) follows \( h(x_0) \equiv 0 \mod N \)

We apply the LLL-reduction algorithm to \( B_1 \) to get an equivalent (reduced) basis \( B'_1 = (b'_1, \ldots, b'_{d+1}) \).

\( b'_1 \) is the coefficient vector of a \( h(xX) \) such that

\[
\| b'_1 \| = \| h(xX) \| \leq 2^{\frac{d}{4}} \cdot \det(L_1)^{\frac{1}{d+1}}
\]

using the proven bounds for the LLL-reduction algorithm.
Second condition of Howgrave-Graham lemma is satisfied, if

\[ 2^{\frac{d}{4}} \det(L_1)^{\frac{1}{d+1}} < \frac{N}{\sqrt{d + 1}} \Rightarrow \cdots \Rightarrow \]

\[ X \leq k(d) N^{\frac{2}{d(d+1)}} \]

where \( k(d) \) is a small, on \( d \) dependent, constant factor.

Summary:

- Using construction \( \mathcal{Z}_1 \) for lattice \( L_1 \) we can find all roots \( x_0 \), s.t. \( f(x_0) \equiv 0 \mod N \) and \( |x_0| < k(d) N^{\frac{2}{d(d+1)}} \)
Improvement (Coppersmith):

- Construction $\mathcal{Z}_2$ for Lattice $L_2$ gives us $X \leq l(d)N^{\frac{1}{2d-1}}$.

$$\mathcal{Z}_2 = \{ N, Nx, \ldots, Nx^{d-1} \} \cup \{ f(x), xf(x), \ldots, x^{d-1}f(x) \}$$

- Further improvements give the bound $X \leq N^{\frac{1}{d}}$.

Theorem (Coppersmith, univariate polynomial equations)

Let $f(x)$ be a monic polynomial of degree $d$ and $N \in MN$ an integer of unknown factorization. If there exist a $x_0$ such that $f(x_0) \equiv 0 \mod N$ and $x_0 < N^{\frac{1}{d}}$, then one can find $x_0$ in polynomial time in $(\log N, d)$.
Applications – Coppersmith’s technique

Example:

- Let $f(x) = x^2 + ax + b$ with $a, b > 0$.
- Consider the modular equation $f(x) \equiv 0 \pmod{N}$.
- Determine bound $X$ with $|p(x)| < N$ for all $x$ with $|x| < X$.
- Consider polynomials $g_0(x) = N$ and $g_1(x) = Nx$.
- Polynomial $h(x) = h_0 + h_1x + h_2x^2$ is a linear combination of polynomials $g_0, g_1$ and $f$.
- Lattice $L_1$ with Basis $B_1$
  
  $$B_1 = \begin{pmatrix}
  N & 0 & b \\
  0 & NX & aX \\
  0 & 0 & X^2 \\
  \end{pmatrix}$$

- LLL returns $h$ with $\|h(xX)\| \leq 2^{\frac{3}{4} - \frac{1}{4}} \det(L_1)^\frac{1}{3} = \sqrt{2}N^{\frac{2}{3}}X$.
- We can find small roots in $[-X, X]$ with $X < \frac{N^{\frac{1}{3}}}{\sqrt{6}}$. 
Applications – Coppersmith’s technique

Algorithm (Coppersmith’s method)

**Input:** $f(x)$ univariate polynomial, $N \in \mathbb{N}$ of unknown factorization
**Output:** Small root $x_0$

1. Use $f(x)$ to construct Basis $B$ of Lattice $L$ where lattice points correspond to polynomials that are root containers of $f$.
2. Run LLL-reduction algorithm on $B$ to get reduced $B'$ with small first basis vector $b'_1$.
3. Consider polynomial $h(x)$ corresponding to $b'_1$ and solve the equation $h(x) = 0$ over the integers.
4. Test the roots obtained in (3) and accept only those that satisfy $f(x_0) \equiv 0 \mod N$.

Preceding analysis guarantees, that all the modular roots of $f(x)$ with $|x_0| < N^{\frac{1}{d}}$ will be found.
Problem: the bivariate case

- Given a bivariate polynomial $p \in \mathbb{Z}[x, y]$ with integer coefficients with maximal degree $\delta$.
- And an integer equation:

\[ p(x, y) = \sum_{i,j} p_{i,j} \cdot x^i y^j = 0 \]

- The goal is to find all pairs $(x_0, y_0)$ such that $p(x_0, y_0) = 0$.

Solution?

- For the general case there are no known, efficient algorithms.
- Use Coppersmith’s technique to find small root pairs.
Theorem (Coppersmith, bivariate integer equations)

Let $p(x, y) \in \mathbb{Z}[x, y]$ be an irreducible polynomial with maximum degree $\delta$ in $x, y$ separately. $X$ and $Y$ are the upper bounds on the solution $(x_0, y_0)$ for $p(x_0, y_0) = 0$ and $W = \max_{i,j}\{|p_{i,j}|X^iY^j\}$.

If $XY \leq W^{2\delta}$, one can find all pairs $(x_0, y_0)$ with $|x_0| \leq X$ and $|y_0| \leq Y$ such that $p(x_0, y_0) = 0$ in polynomial time in $\log W$ and $2\delta$. $p(x_0, y_0) = 0$ with

Basic idea:

- Replace monomial $x^iy^j$ with independent variable $r_{i,j}$.
- $p$ becomes linear relation among several independent bounded variables: $\sum_{i,j} p_{i,j} r_{i,j} = 0$ with $|r_{i,j}| \leq X^iY^j = R_{i,j}$.
Applications – Coppersmith’s technique

Factoring problem: the RSA case

- Find $p, q$ prime for given $N = p \cdot q$.
- Can be modeled as polynomial equation: $f(x, y) = N - xy$.
- The integer roots of this polynomial equation are $(1, N), (p, q), (q, p)$ and $(N, 1)$.
- $p, q$ are of same size, so finding all integer solutions which are in absolute value small than roughly $\sqrt{N}$ is sufficient.
- Define bounds $X, Y$ for finding small solutions.
- Ultimate goal is to find a polynomial time algorithm which succeeds, whenever $XY \leq N$.

Problem:

- No algorithm known so far, therefore relax factorization problem.
Factoring with some bits known:

- Narrow down the search space for the prime factors.
- Assume we are given an approximation $\tilde{p}$ of $p$ such that $|p - \tilde{p}|$ is significantly smaller than $\sqrt{N}$.
- Define an approximation $\tilde{q}$ of $q$ as $\tilde{q} = \frac{N}{\tilde{p}}$.
- We obtain a new bivariate polynomial equation $f(x, y) = N - (\tilde{p} + x)(\tilde{q} + y)$.
- Polynomial $f$ now has small root $(p - \tilde{p}, q - \tilde{q})$.
- The size of the root depends on the quality of the approximation.
- Coppersmith showed, that solution can be found in polynomial time if $XY \leq N^{\frac{1}{2}}$. 
Challenge:

- Some information on bits of primes $p$ and $q$. $p$ and $q$ are of the same bit size.
- We want to recover $p$, hence the factorization of $N = pq$.
- Knowledge of half the bits of $p$ is sufficient to recover $p$ and factor $N$.

Proof.

Sketch: Let $n$ be the bit size of $N$. We write $p = p_12^{\frac{n}{4}} + p_0$ and $q = q_12^{\frac{n}{4}} + q_0$, where $p_i, q_i < 2^{\frac{n}{4}}$. 

\[\square\]
Applications – Coppersmith’s technique

Proof.

Sketch (continued): If we are given the $\frac{n}{4}$ bits of $p$ (half the bits), and we therefore know $p_0$ and thus $q_0$ since $p_0 q_0 \equiv N \mod 2^{\frac{n}{4}}$. We define

$$f(x, y) = \frac{1}{2^{\frac{n}{4}}} \left( \left( x 2^{\frac{n}{4}} + p_0 \right) \left( y 2^{\frac{n}{4}} + q_0 \right) - N \right)$$

$$= xy 2^{\frac{n}{4}} + q_0 x + q_0 y + \frac{1}{2^{\frac{n}{4}}} (p_0 q_0 - N).$$

$f(x, y) \in \mathbb{Z}[x, y]$ with degree $d = 1$ in $x, y$ and $f(p_1, q_1) = 0$. Let $X = Y = N^{\frac{1}{4} - \epsilon}$, then $p_1 < X$ and $q_1 < Y$. In addition we have $W = \|f(x, y)\|_{\infty} \approx N^{\frac{3}{4}}$. Thus $X Y = N^{\frac{1}{2} - 2\epsilon} < (N^{\frac{3}{4}})^{\frac{2}{3}} = W^{\frac{2}{3d}}$.

We can then apply Coppersmith’s theorem for the bivariate case and recover $p_1$ and $q_1$. 

\[\square\]
Other applications:

- Affine Padding: Franklin-Reiter’s attack on RSA. Two plain text messages $m$ and $m'$ that satisfy an affine $m' = m + r$.
- Polynomially related RSA messages: Hastad’s attack. We need messages $m$ encrypted modulo $N_1, \ldots, N_k$ pairwise co-prime. Recover $m$.
- Finding smooth numbers and factoring. $N$ is $B$-smooth if, prime factors $p_i$ are smaller than $B$. We can model this as univariate polynomial equation.
- RSA Key recovering problem: Given a small key.

⇒ A lot of ongoing research on improving the given bounds and on finding new applications for lattice based methods.
NTRU Cryptosystem:

- A (polynomial) ring-based public key cryptosystem.
- In fact a cryptosystem based on the difficulty of the Shortest Vector Problem.

Features:

- Fast encryption and decryption in $O(N^2)$ (RSA in $O(N^3)$).
- Has small keys of size $O(N)$ (like RSA).
- Secure?
Applications – NTRU Cryptosystem

Lattice-based attack by Coppersmith and Shamir

Construct lattice $L_{CS}$ with basis:

$$
\begin{pmatrix}
1 & 0 & \ldots & 0 & 0 & 0 & \ldots & 0 \\
0 & 1 & \ldots & 0 & 0 & 0 & \ldots & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & \ldots & 1 & 0 & 0 & \ldots & 0 \\
h_0 & h_{N-1} & \ldots & h_1 & q & 0 & \ldots & 0 \\
h_1 & h_0 & \ldots & h_2 & 0 & q & \ldots & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
h_{N-1} & h_{N-2} & \ldots & h_0 & 0 & 0 & \ldots & q \\
\end{pmatrix}
$$

We need to find a short vector in this lattice. LLL-reduction not sufficient. Stronger methods needed.
Applications

Summary:

- LLL-reduction is the basis for efficient algorithm by Coppersmith.
- This opened a whole new line of research and enabled new directions for tackling challenging problems. (RSA or factorization problems)
- Lattice theory and in particular lattice basis reduction continues to play an integral role in cryptography.
- Provides effective cryptanalysis tools. Could be source for new cryptographic primitives that exhibit strong security even in the presence of quantum computers.