Exposé Inria

An LLL algorithm for module lattices

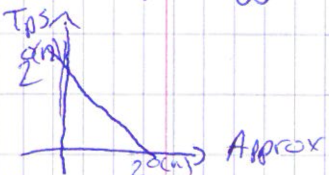
with C. Lee, D. Stehlé, A. Wallet
ePrint 2019/1035

I) Introduction:

* lattice, SVP, CVP → norme = $\| \ \|_2$
  and sometimes $\| \ \|_\infty$

Schnorr87, Schnorr-Euchner94 ↓ Lenstra×2, Lovász 82 ↓

* lattice reduction : BKZ, LLL[82].



(compute a short basis → helps with SVP and CVP)

* module lattices :
  - lattice : storage n^2 (cf codes)

↳ structured lattices

en colonne → $M_a = \begin{bmatrix} a_0 & \cdots & a_{n-1} \\ a_{n-1} \; a_0 & \cdots & a_{n-2} \\ & \ddots & \end{bmatrix}^T$ (cf cyclic codes)

↳ mult by $a = a_0 + a_1 X + \cdots + a_{n-1} X^{n-1}$ mod $X^n - 1$

$R = \mathbb{Z}[X]/X^n - 1$

$\sigma : R \longrightarrow \mathbb{R}^n$ (or even $\mathbb{Z}^n$)

$a \longrightarrow (a_0, \cdots, a_{n-1})$

$M_a = \begin{bmatrix} \sigma(a) & \sigma(Xa) & - & \sigma(X^{n-1}a) \end{bmatrix}$

$$\boxed{L(M_a) = \sigma(\langle a \rangle)}$$

↳ the lattice is an ideal in R (via σ)
called "ideal lattice"

In more generality: ideal lattices $L = \sigma(I)$

* I ideal maybe not principal
  ↳ but in this talk always principal (for simplicity)

* R can be another ring
  ~~usually~~ ↳ for this work: R = ring of integers of K

  for $K = \mathbb{Q}[x]/p(x)$ $\quad$ p irreducible monic degree d
  ↑ number field

  for simplicity: can think of $R = \mathbb{Z}[x]/p(x)$

Eg: $R = \mathbb{Z}[x]/_{x^d+1}$ $\quad d = 2^k$ $\quad$ (power of 2 cyclo)

$R = \mathbb{Z}[x]/_{x^d-x-1}$ $\quad d$ prime $\quad$ (NTRUPrime)

Module lattice: fix some K and $R = \mathbb{Z}[x]/p$

$$B = \begin{pmatrix} M_{a_{11}} & M_{a_{12}} & \cdots & M_{a_{1r}} \\ \vdots & & \ddots & \vdots \\ M_{a_{r1}} & & \cdots & M_{a_{rr}} \end{pmatrix} \quad \text{(blocks of size } d\text{, total } dr\text{)}$$

recall $M_a = \left(\sigma(a) \;\middle|\; \cdots \;\middle|\; \sigma(x^{d-1}a) \right)$

$L(B)$ is a (free) module lattice

Why the name? $\quad \vec{b_i} = (a_{1i}, a_{2i}, \dots, a_{ri}) \in R^r \to K$-linearly indep

$$M = \left\{ \sum x_i \vec{b_i} \,,\ x_i \in R \right\}$$

is an (free) R-module in K^r

and $L(B) = \cancel{\{\sigma(x)\}}\ \sigma(M)$ $\quad$ ($\sigma(\vec{b_i})$ is the concatenated vector $(\sigma(a_{1i}) \| \cdots \| \sigma(a_{ri}))$)

More generally: A module lattice is $\sigma(M)$ for $M \subseteq K^r$ an R-module

↳ pseudo-basis instead of basis

For us today: a module lattice is $\sigma(M)$

where $M = \left\{ \sum_{i=1}^{r} x_i \vec{b_i},\ x_i \in R \right\}$

with $\vec{b_i} \in K^r$ linearly indep.

* $(\vec{b_1}, \dots, \vec{b_r})$ basis of M
* r rank of M

↳ basis of the module:

$$\left(\begin{array}{ccc|c|ccc} \sigma(a_{11}) & \cdots & \sigma(a_{11}X^{d-1}) & \cdots & & & \\ \hline & \vdots & & \ddots & & & \\ \hline \sigma(a_{r1}) & \cdots & \sigma(a_{r1}X^{d-1}) & \cdots & \sigma(a_{rr}) & \cdots & \sigma(a_{rr}X^{d-1}) \end{array} \right)$$

Embeddings $\boxed{\text{Here!}}$ (next page)

Why do we care about module lattices?

RLWE, RSIS, Module-LWE, Module-SIS are equivalent to SIVP in module lattices.
(which is no harder than SVP)

(And NTRU is no harder than SVP in module lattices)

NIST: 11 of the 12 lattice submissions use NTRU, RLWE, RSIS, ...

↳ the modules involved have rank $\simeq 3, 4$, always ≤ 10

$\Rightarrow$ Solving SVP in modules of small rank (≤ 10) has an impact on these constructions.

Embeddings : $\sigma : K \longrightarrow \mathbb{R}^d$

coefficient embedding

$a_0 + a_1 X + \dots + a_{d-1} X^{d-1} \longmapsto (a_0, \dots, a_{d-1})$

canonical embedding : $\sigma : K \longrightarrow \mathbb{C}^d$

$a(X) \longmapsto (a(\alpha_1), \dots, a(\alpha_d))$

with $\alpha_1, \dots, \alpha_d$ the complex roots of P. ~~...~~ $(K = \mathbb{Q}[X]/P)$

A module lattice is $\sigma(M)$ with the same basis as before, but we can change the embedding σ.

* Constructions usually use σ_{coeff} (easier to handle : elements in $\mathbb{Q}$ or even in $\mathbb{Z}$)

* Cryptanalyse usually use $\sigma_{canonical}$ (nice algebraic properties : eg mult is coordinate-wise)

$\sigma_{coeff}(M)$ is usually not the same lattice as $\sigma_{canonical}$

But : * for power-of-2 cyclo → same lattice (up to rotation and scaling)

* NTRU Prime and other "nice" fields (that we want to use) → similar geometry (not exactly the same, but ok).

From now on : $\sigma = \sigma_{canonical}$

A module is a lattice over R

module: $\mathcal{M} = \{ \sum x_i \vec{b_i} ,\ x_i \in R \}$ $\quad \vec{b_i} \in K^r$ linearly indep

$$= \boxed{\vec{b_1} \cdots \vec{b_r}} \times R^r$$

lattice: $\mathcal{L} = \{ \sum x_i \vec{b_i} ,\ x_i \in \mathbb{Z} \}$ $\quad \vec{b_i} \in \mathbb{R}^r$ linearly indep

$$= \boxed{\vec{b_1} \cdots \vec{b_n}}\ \mathbb{Z}^n$$

A module is both:

- "lattice" of rank $\cancel{d}$ r over R

$$\boxed{\vec{b_1} \cdots \vec{b_r}}\ R^r$$

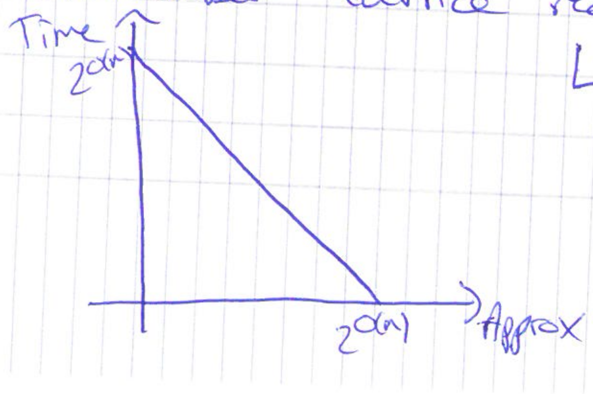
- lattice of rank dr over $\mathbb{Z}$

in practice (NIST): $r \simeq 3, 4$ $\quad (\leq 10)$

$d \simeq 512, 1024, 256$

s.t. $500 \lesssim dr \lesssim 1000$

Remember lattice reduction



↳ this works for lattices over $\mathbb{Z}$. What about lattices over R?

Objective: Adapt LLL to lattices over R.

Why LLL not BKZ? * easier → this is the true reason

* sufficient for modules of rank 3, 4 ...

History: ~~Napias '96 : specific number fields~~
~~no bound~~

	bound on quality	bound on run time
Napias '96 specific number fields	✗	✗
Fieker-Pohst 96 All number fields totally real fields	✗ ✗	✗ ✓
Kim-Lee 17 norm Euclidean fields biquadratic fields	✗ ✓	✓ ✓
This work any field	✓	≈ (✓ if we have an oracle solving CVP in a fixed lattice)

II) LLL in dimension 2 = Lagrange-Gauss algorithm

Over $\mathbb{Z}$: animation on slides

1) What is $\| \ \|$ over R?

- $a \in R$ def: $\|a\|_2 = \|\sigma(a)\|_2$

$\vec{b} \in R^r$ $\|\vec{b}\|_2 = \|(\sigma(b_1)\| \cdots \|\sigma(b_r)\|\|_2$

$= \sqrt{\sum \|\sigma(b_i)\|^2}$ (Pythagore)

This is what we want: small in $R^r \Leftrightarrow$ small in $\mathbb{Z}^{rd}$

Two notions of number theory: $\mathrm{Tr}(a) = \sum \sigma(a)_i$

$\mathcal{N}(a) = \prod \sigma(a)_i$

~~* assume $\bar{a}$ is well defined~~ ~~(e.g. $R = \mathbb{Z}[x]/x^n+1$)~~

Define $\bar{a} = (\overline{\sigma(a)_1}, \dots, \overline{\sigma(a)_d})$ and assume $\bar{a} \in R$ if $a \in R$ (for example $R = \mathbb{Z}[X]/X^n+1$)

Then $\|a\|_2 = \sqrt{\sum |\sigma(a)_i|^2} = \sqrt{\mathrm{Tr}(a\bar{a})}$

and $\|\vec{b}\|_2 = \sqrt{\sum_i \mathrm{Tr}(b_i\bar{b_i})} = \sqrt{\mathrm{Tr}(\sum b_i\bar{b_i})}$

↳ define a "scalar product"

$\langle \vec{b}, \vec{c} \rangle = \sum_i b_i \bar{c_i} \in R$ (or K)

2) QR factorisation over $\mathbb{R}^2$

~~For i⇒~~

$$b_1^* = b_1 \qquad b_2^* = b_2 - \overbrace{\frac{\langle b_2, b_1^* \rangle}{\langle b_1^*, b_1^* \rangle}}^{\in K} b_1^* \in K^2$$

$\hookrightarrow \langle b_2^*, b_1^* \rangle = 0$

$$Q = \begin{pmatrix} \frac{b_1^*}{\|b_1^*\|} & \frac{b_2^*}{\|b_2^*\|} \end{pmatrix} \qquad R = \begin{pmatrix} \|b_1^*\| & \frac{\langle b_2, b_1^* \rangle}{\|b_1^*\|} \\ 0 & \|b_2^*\| \end{pmatrix}$$

$\|b_i^*\| = \sqrt{\langle b_i^*, b_i^* \rangle}$

<u>Qu</u>: $\sqrt{\ }$ in K? → can avoid $\sqrt{\ }$ by using "Gram-Schmidt"

$\hookrightarrow$ div of $\frac{\langle b_2, b_1^* \rangle}{\|b_1^*\|}$ by $\|b_1^*\|$

$\Leftrightarrow$ div of $\langle b_2, b_1^* \rangle$ by $\|b_1^*\|^2$

$\hookrightarrow$ or, we can define $K_{\mathbb{R}}$ and $\sqrt{\ }$ is well defined

<u>Properties</u>:
* $QR = \underbrace{(b_1 \ b_2)}_{B}$
* R triangular
* $Q\bar{Q}^T = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$
* $R\begin{pmatrix} u \\ v \end{pmatrix}$ short $\Leftrightarrow B\begin{pmatrix} u \\ v \end{pmatrix}$ short

(preserves geometry)

$\hookrightarrow \langle B\begin{pmatrix} u \\ v \end{pmatrix}, B\begin{pmatrix} u \\ v \end{pmatrix} \rangle$

$= (\bar{u} \ \bar{v}) \bar{B}^T B \begin{pmatrix} u \\ v \end{pmatrix}$

$= (\bar{u} \ \bar{v}) \bar{R}^T \underbrace{\bar{Q}^T Q}_{I_2} R \begin{pmatrix} u \\ v \end{pmatrix} = \langle R\begin{pmatrix} u \\ v \end{pmatrix}, R\begin{pmatrix} u \\ v \end{pmatrix} \rangle$

and $\| \ \|$ follows because $\sqrt{Tr(\langle , \rangle)}$

↳ QR factorisation is ok, we can assume that our basis is triangular

3) Euclidean division

$a, b \in R$ (or K)

we would like $r \in R$ s.t. $\|a + rb\| \leq \frac{1}{2}\|b\|$

<u>Pb</u>: Most of the number fields we are interested in are not euclidean → no such r.

<u>But</u>: There should exist (counting argument) $u, v \in R$ s.t.

* $\|au + bv\| \leq \frac{1}{2}\|b\|$
* $\|u\| \leq poly(d)$

(here, R is "nice"
small basis → (small volume)
$\Delta^{1/d}$ in general
or even $\lambda_n(R)$)

↳ we can use it to reduce a small multiple of $\vec{b_2}$ by $\vec{b_1}$.

<u>Rq</u>: over $\mathbb{Z}$: when we have $\begin{pmatrix} r_{11} & r_{12} \\ 0 & r_{22} \end{pmatrix}$

and $r_{11} \lesssim r_{22}$, we are roughly done

because any vector is $\geq \min(r_{11}, r_{22}) \simeq r_{11}$

↳ so $\begin{pmatrix} r_{11} \\ 0 \end{pmatrix}$ is a small vector.

→ Same is true over R.

The case where we make progress is $r_{22} \ll r_{11}$. And then, even if we compute

$$\left\| u \begin{pmatrix} r_{12} \\ r_{22} \end{pmatrix} + v \begin{pmatrix} r_{11} \\ 0 \end{pmatrix} \right\|^2$$ with u not too large

$$= \underbrace{u^2 r_{22}^2}_{\ll r_{11}^2} + \underbrace{(u r_{12} + v r_{11})^2}_{\leq \left(\frac{r_{11}}{2}\right)^2} \lesssim \frac{r_{11}^2}{4}$$

↳ we make progress

↳ again, same is true over R

3.2) How to compute it?

input: $a, b \in R$

output: $u, v \in R$ s.t. $\|au + bv\| \leq \frac{1}{2}\|b\|$

$\|u\| \leq \mathrm{poly}(d)$

1. $\|au + bv\| \leq \frac{1}{2}\|b\|$ means au should be close to $-bv$ (or to bv, if we change the sign of v)

2. au and bv: products $\longrightarrow$ take the Log to have sums

def $\mathrm{Log}: R \longrightarrow \mathbb{R}^d$

$x \longmapsto (\log(|\sigma(x)_i|))_i$

typical in number theory

pb: $\mathrm{Log}(i) = \mathrm{Log}(1)$ but i not close to 1

so $\|\mathrm{Log}(au) - \mathrm{Log}(bv)\|$ small

$\not\Rightarrow$ $\|au - bv\|$ small

$\rightarrow$ use $\overline{\mathrm{Log}} : R \rightarrow \mathbb{R}^{d} \times (\mathbb{R}/2\pi)^d$

$$x \longmapsto \left(\log|\sigma_i(x)|, \theta(\sigma_i(x))\right)_i$$

$$\theta(re^{ir}) = r$$

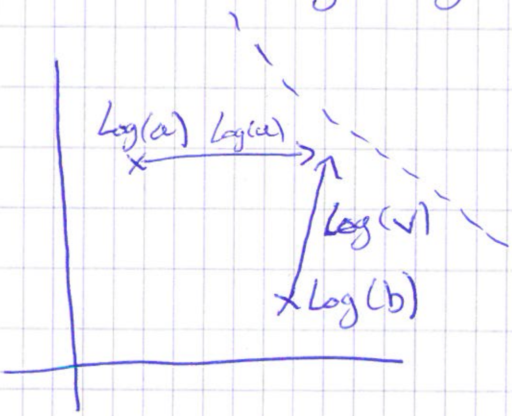
<u>Now</u>: if $\|\overline{\mathrm{Log}}(x) - \overline{\mathrm{Log}}(y)\| \leq \varepsilon$ $+ \varepsilon \leq \frac{1}{2}$

then $\|x-y\|_\infty \leq 4\varepsilon \min(\|x\|_\infty, \|y\|_\infty)$

$$\left(\|x-y\| \leq \varepsilon^{(\sqrt{d})} \min(\|x\|, \|y\|) \right)$$

$\hookrightarrow$ obj: $\|\mathrm{Log}(ua) - \mathrm{Log}(vb)\| \leq \frac{1}{\mathrm{poly}(d)}$ + $\|\mathrm{Log}(u)\|, \|\mathrm{Log}(v)\| \leq O(\log(d))$

<u>Picture</u> (with only Log) just for the idea (maths after)



$\mathrm{Log}(au) = \mathrm{Log}(a) + \mathrm{Log}(u)$

obj: $\mathrm{Log}(u/v)$ close to $\mathrm{Log}(b/a)$

$\hookrightarrow$ this is a CVP over a set which is not a lattice

$\hookrightarrow$ make it a lattice: consider a factor basis $g_1, \ldots, g_k$ and look for u, v as a product of g_i's

$$B = \begin{bmatrix} \overline{\mathrm{Log}}(g_1) & \cdots & \overline{\mathrm{Log}}(g_k) \\ | & & | \\ \hline 1 & & \\ & \ddots & \\ & & 1 \end{bmatrix} \qquad L = \mathcal{L}(B)$$

$$r = \begin{bmatrix} \overline{\mathrm{Log}}(b/a) \\ \hline 0 \end{bmatrix}$$

$\rightarrow$ add a block of 2π + a block of units

Algo : * compute L and t

* solve CVP in L with t → output s

* write $s = \begin{bmatrix} \mathrm{Log}(u/v) \\ * \end{bmatrix}$

* output u, v

Why does it work? (Correctness of Algo) How close is s to t? (i.e. How close $\mathrm{Log}(ua)$ to $\mathrm{Log}(vb)$)

↳ depends on the density of L.

Vol(L) is fixed → increasing dim shorter vectors

→ with a heuristic counting argument, we believe that $k = O(d^2)$ is enough for L to be dense enough.

Run time of Algo : everything poly except "solve CVP in L"

↳ assume oracle and everything becomes poly-time

Now, we have the 2 key ingredients: QR and Euclidean div → we can do LLL in R^2.

↳ to prove that LLL terminates in poly time + output is small we need an extra ingredient : $\mathcal{N}(\cdot)$

III) LLL in dim r

Very similar to dim 2 :

QR $\begin{pmatrix} r_{11} & & r_{ij} \\ & \ddots & \\ 0 & & r_{rr} \end{pmatrix}$ → choose i s.t. $\begin{pmatrix} r_{ii} & r_{i\,i+1} \\ 0 & r_{i+1\,i+1} \end{pmatrix}$

can be improved

and do the reduce

and SWAP.

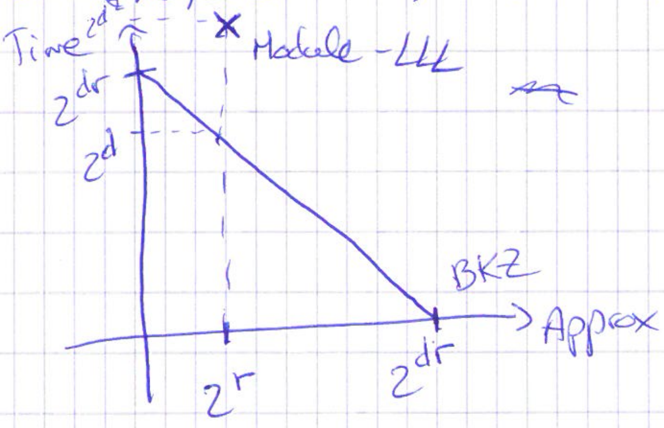
Result : Approx factor quasi-poly$(d)^{O(r)}$

Time poly(d, k) if oracle

what if we actually want to run it?

need to instanciate the oracle → with generic algorithms

$\dim(L) = d^2$ → CVP in L can be done in 2^{d^2}



$$\boxed{\text{Don't use it in practice}}$$

Open problems :

* improve CVP in $\mathcal{L}$?
  - ↳ decrease its dim ?
  - ↳ use its structure ?

* Generalize LLL to BKZ ?

LLL : leverage SVP in dim 2 —> 2^n-SVP in dim n

BKZ : SVP in dim β —> $2^{n/\beta}$-SVP in dim n

The reduction should be easy
The hard part is how to solve SVP in dim β
(That was the hard part in LLL too)