Lattice Reduction: Problems and Algorithms

Damien STEHLÉ

Pythagorion, Samos, Greece

http://www.loria.fr/~stehle/

stehle@loria.fr
Plan of the talk.

- Mathematical definitions: lattices, lattice invariants, reduction.
- Algorithmic lattice problems: $\gamma$-SVP, $\gamma$-CVP.
- Lattice algorithms: Gauss, LLL, BKZ.
- Practical lattice reduction.
Usefulness of lattices in computer science.

- Great tool for cryptanalysis.
- Interesting for building cryptosystems.
- Computer algebra: factorization of polynomials over $\mathbb{Z}$.
- Algorithmic number theory: ideals in number fields, small roots of polynomials, minimal polynomials . . .
Basic Geometry of Numbers
A first definition of a lattice.

A lattice is a discrete subgroup of a Euclidean space.

- **Euclidean space**: we are living in \( \mathbb{R}^n \).
- **Subgroup**: 1) \( b \in L \Rightarrow -b \in L \),
  2) \( b_1, b_2 \in L \Rightarrow b_1 + b_2 \in L \).
- **Thus**: \( 0 \in L \), and \( L \) is stable by linear integer combinations.
- **Discrete**: no accumulation point,
  i.e., there is a small open ball containing only \( 0 \).
First examples.

- Simplest non-trivial example: \( \mathbb{Z} \subset \mathbb{R} \).
- Quite simple too: \( \mathbb{Z}^d \subset \mathbb{R}^n \) with \( d \leq n \).
- Any subgroup of \( \mathbb{Z}^d \subset \mathbb{R}^n \) with \( d \leq n \).
A 2-dimensional lattice.
The same lattice.
Second definition of a lattice.

A lattice is the set of all integer linear combinations of some linearly independent vectors in a Euclidean space.

- The two definitions are equivalent.
- \( L = \left\{ \sum_{i=1}^{d} x_i b_i, (x_1, \ldots, x_d) \in \mathbb{Z}^d \right\} \),
  where the \( b_i \)'s are linearly independent vectors of \( \mathbb{R}^n \).
- **Lattice dimension**: \( d \).
- **Embedding dimension**: \( n \).
- \( b_1, \ldots, b_d \) is a lattice basis. It is not unique.
The second definition is not always the good one.

Let \( A = (a_{i,j})_{i,j} \) be an \( n \times m \) matrix of integers with \( n < m \).

Consider the system of integer equations:

\[
\begin{align*}
    a_{1,1}x_1 &+ a_{1,2}x_2 + \ldots + a_{1,m}x_m = 0 \\
    a_{2,1}x_1 &+ a_{2,2}x_2 + \ldots + a_{2,m}x_m = 0 \\
    &\vdots &\vdots &\ddots &\vdots &\vdots \\
    a_{n,1}x_1 &+ a_{n,2}x_2 + \ldots + a_{n,m}x_m = 0
\end{align*}
\]

The set of solutions \((x_1, \ldots, x_m)\) is a lattice \( L \).

If the rows of \( A \) are linearly independent, \( \dim(L) = m - n \).
Two bases of a 3-dimensional lattice.
An infinity of bases for a given lattice.

- For a given lattice, the bases are related by **unimodular transformations**: $d \times d$ integral matrices with determinant $\pm 1$.
- You can: 1) permute vectors,
  2) add to a given basis vector another basis vector.
- Interesting bases are made of **short and orthogonal** vectors.
**Lattice volume:** $\det(L)$.

- The $d$-dimensional volume of the parallelepiped spanned by lattice basis vectors, for any basis.
- If $d = n$, absolute value of the determinant of a lattice basis.
- In general, $\det(L) = \sqrt{\det G(b_1, \ldots, b_d)}$, where $G$ is the Gram matrix of the $b_i$’s: $(\langle b_i, b_j \rangle)_{i,j}$.

- The orthogonality defect $\frac{\|b_1\| \ldots \|b_d\|}{\det(L)}$ gives a measure for the quality of a basis $(b_1, \ldots, b_d)$ of $L$. 
The lattice volume is a lattice invariant.
The second definition is not always the good one.

- Suppose that $\gcd(a_1, \ldots, a_n) = 1$, and $N \in \mathbb{Z}$.
- $a_1x_1 + a_2x_2 + \ldots + a_nx_n = 0 \mod N$.
- The set of solutions $(x_1, \ldots, x_n)$ is a $n$-dimensional lattice $L$.
- Let $\phi : \mathbb{Z}^n \rightarrow \mathbb{Z}$
  
  \[
  (x_1, \ldots, x_n) \rightarrow \sum a_ix_i \mod N.
  \]
- $L = \ker \phi$ and $\phi$ is onto $\Rightarrow \mathbb{Z}^n / L \cong \mathbb{Z}_N$.
- $\det(L) = [\mathbb{Z}^n : L] \cdot \det(\mathbb{Z}^n) = N \cdot 1$. 
Lattice minima: $\lambda_i(L)$.

- There exists a shortest non-zero vector, its length is $\lambda_1(L)$.
- For $i \leq d$, $\lambda_i(L)$ is the minimum radius $r$ for which $B(0, r)$ contains $i$ linearly independent lattice vectors.
- Fact: there exist linearly independent vectors reaching the $\lambda_i$’s.
The two Minkowski theorems.

- Based on the pigeon-hole principle.
- Minkowski 1: \( \lambda_1 \leq \sqrt{d} \cdot (\text{det}(L))^{1/d} \).
- Minkowski 2: \( \lambda_1 \ldots \lambda_d \leq d^{d/2} \cdot \text{det}(L) \).

- For a “random” lattice, we expect these bounds to be tight:
  \[ \lambda_1 \approx \lambda_2 \approx \ldots \approx \lambda_d \approx \text{det}(L)^{1/d}. \]
Lattice basis reduction (1/2).

- A reduced basis is made of rather orthogonal and short vectors.
- What would be the best definition?
- A basis reaching the $\lambda_i$’s? Not always possible when $d \geq 5$:

\[
\begin{pmatrix}
2 & 0 & 0 & 0 & 0 \\
0 & 2 & 0 & 0 & 0 \\
0 & 0 & 2 & 0 & 0 \\
0 & 0 & 0 & 2 & 0 \\
1 & 1 & 1 & 1 & 1
\end{pmatrix}
\]

$\forall i, \lambda_i = 2$, but any basis made of norm-2 vectors is orthogonal.
Lattice basis reduction (2/2).

- Several definitions to work around the failure of the natural one.
- A basis is reduced if the lengths of its vectors are close to the $\lambda_i$'s.
- Minkowski, Hermite-Korkine-Zolotarev: very strong reductions.
- LLL, BKZ: weaker definitions, but easier to get.
Lattice Related Algorithmic Problems
How to represent a lattice?

- 1st difficulty: a lattice is infinite.  
  ⇒ A lattice is represented by one of its bases.

- 2nd difficulty: basis vectors may have real coordinates.  
  ⇒ We consider only integral lattices: sublattices of $\mathbb{Z}^n$.

- Basically, a lattice is represented by a $d \times n$ integral matrix.
The shortest vector problem: SVP.

- Given a basis of $L$, compute a vector of length $\lambda_1(L)$.
- $\gamma$-SVP: Compute a vector of length $\leq \gamma \cdot \lambda_1(L)$.
- Expected solution: a vector of length $\approx \det(L)^{1/d}$.
- If $\lambda_1$ is much shorter than this, it might be easier.
Effective Minkowski theorem problem: EMTP.

- **EMTP**: Compute a lattice vector $\mathbf{b}$ with $\|\mathbf{b}\| \leq \sqrt{n} \cdot \det(L)^{1/d}$.

- **$\gamma$-EMTP**: Compute a lattice vector $\mathbf{b}$ with $\|\mathbf{b}\| \leq \gamma \cdot \det(L)^{1/d}$.

- **$\gamma$-EMTP2**: Compute a lattice basis $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ with $\|\mathbf{b}_1\| \ldots \|\mathbf{b}_d\| \leq \gamma \cdot \det(L)$. 
The closest vector problem: CVP.

- Given a basis of $L$ and a vector $t$ of the embedding space, compute a lattice vector closest to $t$.

- $\gamma$-CVP: Given a basis of $L$ and a target vector $t$, compute a lattice vector $b_0$ such that $\|b_0 - t\| \leq \gamma \cdot \min_{b \in L} \|b - t\|$.

![Diagram of the closest vector problem](image)
More on CVP.

- A “general” solution should be at distance $\det(L)^{1/d}$ of $t$.
- Intuition of the difficulty: Consider $t = (1/2, \ldots, 1/2)$ and slightly shake $\mathbb{Z}^d$. Which one of the $2^d$ vertices is the solution?
- CVP is considered harder than SVP.
Lattice Reduction Algorithms
Two different goals.

- Get a reasonable approximation factor very quickly.
- Spend some time to get a better approximation factor.
- Often we want a trade-off between both goals.
- What we can afford: Polynomial Time / Practicability.
The 2-dimensional case.

- Gauss (Lagrange?) algorithm **solves everything**.
- Vectorial generalization of Euclid’s algorithm.
- Running time: $O(\log^2 B)$, where $B = \max(\|b_1^{\text{init}}\|, \|b_2^{\text{init}}\|)$.
- Algorithm: shorten the long vector by adding to it an integer multiple of the short one, until this is possible.
The 2-dimensional Case.
The 2-dimensional case.
The 2-dimensional case.
The 2-dimensional case.
The 2-dimensional case.
The 2-dimensional case.
The 2-dimensional case.
When the dimension remains low.

- Suppose we want a $d$-dimensional HKZ-reduced basis.
- For small $d$, exponential algorithms remain feasible.
- Algorithms: Kannan, Ajtai-Kumar-Sivakumar.
- SVP and CVP solved in practice up to dimension $\approx 25 - 30.$
When the dimension grows significantly (1/2).

- Use the LLL algorithm (Lenstra, Lenstra, Lovász - 1982).

- It gives an LLL-reduced basis \( (b_1, \ldots, b_d) \) with:

\[
\|b_1\| \leq c^d \cdot \det(L)^{1/d},
\]

\[
\|b_i\| \leq c^{2d} \cdot \lambda_i(L),
\]

where \( c = (4/3)^{1/4} - \varepsilon \approx 1.075 \).

- Time: \( O(d^5 n \log^3 B) \), with \( B = \max_{i \leq d} \|b_i^{\text{init}}\| \).

- With floating-point arithmetic: \( O(d^4 n (d + \log B) \log B) \).
When the dimension grows significantly (2/2).

- What if you want a basis more reduced than given by LLL?
  \[ \Rightarrow \text{Mix LLL and HKZ-reduction.} \]

- This is Schnorr’s Block-Korkine-Zolotarev algorithm:
  \[ \text{LLL} = \text{BKZ}_2, \quad \text{HKZ} = \text{BKZ}_d. \]

- BKZ\(_k\) costs \( k^{O(k)} \) and gives \( \gamma = k^{O(n/k)} \) for SVP.

  \[ \Rightarrow \text{Best } \gamma \text{ for deterministic polynomial time: } 2^{O\left( k \left( \frac{\log \log k}{\log k} \right)^2 \right)} . \]

- BKZ is feasible for \( k \leq 25 \) to 30.
Practical Lattice Reduction.
Quoting Shoup’s NTL documentation.

“I think it is safe to say that nobody really understands how the LLL algorithm works. The theoretical analyses are a long way from describing what "really" happens in practice. Choosing the best variant for a certain application ultimately is a matter of trial and error.”
Lattices arising in real life (1/2).

- Small-dimensional lattices (e.g., in Wiener’s attack).
- Knapsack-like lattices (knapsacks):

\[
\begin{pmatrix}
X_1 & 1 & 0 & \ldots & 0 & 0 \\
X_2 & 0 & 1 & \ldots & 0 & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
X_d & 0 & 0 & \ldots & 0 & 1 \\
\end{pmatrix}
\]

with rather large $X_i$’s, and a large $d$ (100-200).
Sometimes LLL suffices, but BKZ is usually required.

- Coppersmith-type lattices: very large entries, medium dimension (70), LLL suffices.
Lattices arising in real life (2/2).

NTRU-like lattices: small entries (< 10 bits), very large dimension (167-503), very good reduction is required.

\[
\begin{pmatrix}
1 & 0 & \ldots & 0 & h_0 & h_1 & \ldots & h_{n-1} \\
0 & 1 & \ldots & 0 & h_1 & h_2 & \ldots & h_0 \\
\vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \ddots & \vdots \\
0 & 0 & \ldots & 1 & h_{n-1} & h_0 & \ldots & h_{n-2} \\
0 & 0 & \ldots & 0 & q & 0 & \ldots & 0 \\
0 & 0 & \ldots & 0 & 0 & q & \ldots & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \ddots & \vdots \\
0 & 0 & \ldots & 0 & 0 & 0 & \ldots & q \\
\end{pmatrix}
\]
Where can one find lattice algorithms implementations?

- NTL: efficient LLL, improved BKZ.
- Magma: less efficient LLL, nice routines in low dimensions.
- On my webpage: floating-point LLL (no BKZ), quite fast.
- Others: Lidia, Maple, Mathematica, PSLQ.
What can one do in practice?

- Complete (KZ) reduction in low dimensions: $d \leq 25$ to $30$.
- LLL-reduction of large lattices ($d \leq 1000$).
  
  Knapsack-type lattice with $d = 121$ and $\log B = 29000$: 15 min.
  
  Knapsack-type lattice with $d = 50$ and $\log B = 100000$: 6 min.

- BKZ$_{30}$ in dimension $\leq 300$ takes some time but should terminate.
On the quality of bases output by LLL.

Does LLL find vectors much shorter than expected?

- $\|b\| \leq (4/3)^{d/4} \det(L)^{1/d}$, with $(4/3)^{1/4} \approx 1.075$.
- Experimentally, for “random” lattices: $1.075 \rightarrow 1.03$ (?)..
- Widespread belief: LLL gives a solution to SVP very often, and approximates SVP very well.
- Explanation: in the 80’s, people were working with medium-size lattices, and: $(1.03)^{30} \approx 2.4$, $(1.03)^{50} \approx 4.4$, $(1.03)^{70} \approx 7.9$.
- Yet much remains unknown about its behavior.
Main open problems.

- Comprehension of the practical behavior of LLL and BKZ.
- Faster lattice reduction algorithms.
- An efficient algorithm solving Poly($d$)-SVP.
Some bibliography.

- Siegel, Lectures on the Geometry of Numbers.
- Cohen, A Course in Computational Algebraic Number Theory.
- Micciancio & Goldwasser: Complexity of Lattice Problems, A Cryptographic Perspective.