The LLL Algorithm for Lattices

G. Eric Moorhouse, UW Math

References

Henri Cohen, *A Course in Computational Algebraic Number Theory,* Springer, 1993.

A.J. Menezes et al., *Handbook of Applied Cryptography,* CRC Press, 1997.

A.K. Lenstra, H.W. Lenstra and L. Lovász, 'Factoring polynomials with rational coefficients', *Math. Ann.* **261** (1982), 515–534.

M. Pohst, 'A modification of the LLL-algorithm', *J. Symb. Comp.* **4** (1987), 123–128.

Definitions

A **lattice** L is a pair $(\mathbb{Z}^n, Q)$ where

$$Q : \mathbb{Z}^n \to \mathbb{R}$$

is a positive definite quadratic form, i.e. $Q(\mathbf{x}) = \mathbf{x}^\top A\mathbf{x}$ where the real $n \times n$ matrix A is symmetric positive definite. We call A a **Gram matrix** of L.

Two lattices $(\mathbb{Z}^n, Q)$, $(\mathbb{Z}^n, Q')$ are *isometric* if there exists a **unimodular integer transformation** $M \in GL(n, \mathbb{Z})$ (i.e. M and M^{-1} have integer entries) such that

$$Q'(\mathbf{x}) = Q(M\mathbf{x}) \quad \text{for all } \mathbf{x} \in \mathbb{Z}^n;$$

equivalently, $A' = M^\top AM$.

Every lattice $L = (\mathbb{Z}^n, Q)$ is isometric to a subset of $\mathbb{R}^m$ (for each $m \geq n$) using the standard real inner product $\langle\ ,\ \rangle$. This gives an alternative definition of a lattice:

A **lattice** L is a discrete additive subgroup of $\mathbb{R}^m$; that is, L is the $\mathbb{Z}$-span of a linearly independent subset of $\mathbb{R}^m$:

$$L = \mathbb{Z}\mathbf{b}_1 + \mathbb{Z}\mathbf{b}_2 + \cdots + \mathbb{Z}\mathbf{b}_n$$

with the quadratic form $Q(\mathbf{x}) = \langle \mathbf{x}, \mathbf{x} \rangle$ for $\mathbf{x} \in L$. (Note: $n \leq m$.) The vectors $\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n$ are a **basis** for L, and $A = [\langle \mathbf{b}_i, \mathbf{b}_j \rangle]_{1 \leq i,j \leq n}$ is the corresponding Gram matrix.

Two linearly independent sets of vectors generate the same lattice iff they are related by a unimodular integer transformation on $\mathbb{R}^m$. Two Gram matrices represent isometric lattices iff they are **integrally congruent**: $A' = M^\top A M$ for some $M \in GL(n, \mathbb{Z})$.

Reduced Bases

The lattice $L \subset \mathbb{R}^2$ with basis

$$\mathbf{b}_1 = \binom{10}{14}, \quad \mathbf{b}_2 = \binom{24}{33}$$

and Gram matrix

$$A = \begin{bmatrix} 296 & 702 \\ 702 & 1665 \end{bmatrix}$$

has reduced basis

$$\mathbf{b}'_1 = -7\mathbf{b}_1 + 3\mathbf{b}_2 = \binom{2}{1},$$

$$\mathbf{b}'_2 = 19\mathbf{b}_1 - 8\mathbf{b}_2 = \binom{-2}{2}$$

and Gram matrix

$$A' = M^\top A M = \begin{bmatrix} 5 & -2 \\ -2 & 8 \end{bmatrix}$$

where $M = \left[\begin{smallmatrix} -7 & 19 \\ 3 & -8 \end{smallmatrix}\right]$.

The technical definition of "reduced" later...

Important Algorithms

LLL Algorithm—Given a lattice L by way of a basis $\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n$ for $L \subset \mathbb{R}^m$, we find (in polynomial time) a "reduced" basis $\mathbf{b}'_1, \mathbf{b}'_2, \ldots, \mathbf{b}'_n$ for L in $\mathbb{R}^m$.

Or given a Gram matrix A for L, we find (in polynomial time) the Gram matrix A' for L with respect to a reduced basis.

In both cases, the unimodular integer matrix M is also determined.

Often the shortest lattice vectors in L are among the basis vectors found by LLL.

If A has integer entries, all computations can be done *exactly* in $\mathbb{Z}$ using arbitrary precision integer arithmetic.

MLLL Algorithm—Modified LLL algorithm due to M. Pohst (1987). We are given an $m \times n$ real matrix W whose columns generate a lattice $L \subset \mathbb{R}^m$. (The columns need not be linearly independent.) We find (in polynomial time) a reduced basis for L, *and* a (reduced) basis for the kernel of the map $W : \mathbb{Z}^n \to \mathbb{Z}^m$.

Or given the positive semidefinite Gram matrix of a set of vectors $\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n \in \mathbb{R}^m$ generating a lattice L, we find a reduced basis for L (expressed as linear combinations of the $\mathbf{b}_i$'s), *and* a reduced basis for the lattice of relations

$$\{(r_1, r_2, \ldots, r_n) \in \mathbb{Z}^n \ : \ \sum_{i=1}^{n} r_i \mathbf{b}_i = \mathbf{0}\}.$$

A pure integer version exists.

Fincke-Pohst Algorithm—Given a lattice $L = (\mathbb{Z}^n, Q)$ and a constant $C > 0$, find all $\mathbf{x} \in \mathbb{Z}^n$ such that $Q(\mathbf{x}) < C$. The algorithm runs in exponential time but works in many practical situations. It makes use of LLL as a subalgorithm.

The best way to determine with certainty *the shortest* nonzero vectors in L is to let C be the norm of the shortest basis vector in a reduced basis (found using LLL); then to use Fincke-Pohst to search for smaller vectors in L, if any.

Determinants of Lattices

The **determinant** of L is

$$d(L) = \sqrt{\det(A)}$$

where A is a Gram matrix for L. Or equivalently (if $L \subset \mathbb{R}^n$ has rank n), $d(L) = |det(B)|$ where B is an $n \times n$ matrix whose columns form a basis $\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n$ for L.

Hadamard's Inequality

$d(L) \leq \prod_{j=1}^{n} \|\mathbf{b}_j\|$, and equality holds iff the $\mathbf{b}_j$'s are orthogonal.

A "reduced" basis should have $\prod_{j=1}^{n} \|\mathbf{b}_j\|$ rather small; equivalently, the $\mathbf{b}_j$'s should be close to orthogonal.

Gram-Schmidt Process

We have

$$0 \subset L_1 \subset L_2 \subset \cdots \subset L_n = L$$

where

$$L_j = \mathbb{Z}\mathbf{b}_1 + \mathbb{Z}\mathbf{b}_2 + \cdots + \mathbb{Z}\mathbf{b}_j\,.$$

The orthogonal projection of $\mathbf{b}_j$ onto $L_{j-1}^{\perp}$ is found recursively to be

$$\mathbf{b}_j^* = \mathbf{b}_j - \sum_{1 \leq k < j} \mu_{j,k}\mathbf{b}_k^*$$

where

$$\mu_{j,k} = \frac{\mathbf{b}_j \cdot \mathbf{b}_k^*}{\mathbf{b}_k^* \cdot \mathbf{b}_k^*}\,.$$

Then $\{\mathbf{b}_1^*, \mathbf{b}_2^*, \ldots, \mathbf{b}_n^*\}$ is an orthogonal (not necessarily orthonormal) basis of $\mathbb{R}L = \mathbb{R} \otimes_{\mathbb{Z}} L$.

Note that $d(L) = \prod_{j=1}^{n} \|\mathbf{b}_j^*\|$.

Definition of Reduced Basis

A basis $\{\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n\}$ for L is **reduced** if

(i) $|\mu_{j,k}| \le \frac{1}{2}$ for $1 \le j < k \le n$, and

(ii) $\|\mathbf{b}_j^*\|^2 \ge (\frac{3}{4} - \mu_{j,j-1}^2)\|\mathbf{b}_{j-1}^*\|^2$ for $1 < j \le n$.

The latter inequality is equivalent to

(ii)' $\|\underbrace{\mathbf{b}_j^* + \mu_{j,j-1}\mathbf{b}_{j-1}^*}_{\text{proj}_{L_{j-2}^{\perp}}(\mathbf{b}_j)}\|^2 \ge \frac{3}{4}\|\underbrace{\mathbf{b}_{j-1}^*}_{\text{proj}_{L_{j-2}^{\perp}}(\mathbf{b}_{j-1})}\|^2$

Theorem. *A reduced basis satisfies*

$$d(L) \le \prod_{j=1}^{n} \|\mathbf{b}_j\| \le 2^{n(n-1)/4} d(L);$$

$$\|\mathbf{b}_1\| \le 2^{(n-1)/2}\|\mathbf{x}\| \quad \textit{for all nonzero } \mathbf{x} \in L;$$

$$\|\mathbf{b}_1\| \le 2^{(n-1)/4} d(L).$$

LLL Algorithm

Input a basis $\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n$ for L. The following procedure replaces these vectors by a reduced basis.

1. Set $j = 1$.

2. For each $k = 1, 2, 3, \ldots, j-1$, if $|\mu_{j,k}| > \frac{1}{2}$, replace $\mathbf{b}_j$ by $\mathbf{b}_j - r\mathbf{b}_k$ where $r \in \mathbb{Z}$ is chosen so that

$$\mu'_{j,k} = \frac{(\mathbf{b}_j - r\mathbf{b}_k) \cdot \mathbf{b}_k^*}{\mathbf{b}_k^* \cdot \mathbf{b}_k^*} = \mu_{j,k} - r \in [-0.5, 0.5].$$

3. If the Lovász condition (ii) is satisfied, increment k by one and go to Step 2 (unless $k = n$, in which case we are done).

Otherwise interchange $\mathbf{b}_{k-1}$ with $\mathbf{b}_k$, decrease k by 1 and go to Step 2.

Why the Algorithm Terminates

Let $D = \prod_{j=1}^{n} d(L_j)$ where $d(L_j) = \prod_{k=1}^{j} \|\mathbf{b}_k^*\|^2$.

The value of D changes only in Step 3, where

L_j changes only for $j = k-1$;

$d(L_{k-1})$ is replaced by
$d(L'_{k-1}) \le (\frac{3}{4})^{1/2} d(L_{k-1})$; and

D is replaced by $D' \le (\frac{3}{4})^{1/2} D$.

Since $d(L_{k-1}) \ge (\|\mathbf{x}\|/\gamma_{k-1}^{1/2})^{k-1}$ where γ_{k-1} is Hermite's constant (the maximum of $\min\{\|\mathbf{v}\| : \mathbf{0} \ne \mathbf{v} \in \Lambda\}$ for all lattices Λ of rank $k-1$ and determinant 1) and $\mathbf{x}$ is a shortest nonzero vector in L, step 3 can be executed only a finite number of times.

More careful analysis shows that the running time is $O(n^6 (\log M)^3)$ where $M = \max \|\mathbf{b}_i\|^2$.

Implementations of LLL

1. **MAPLE V Release 5**. LLL only (no MLLL or Fincke-Pohst). Very accessible. But doesn't use Gram matrices; requires an explicit list of generators.

2. **Keith Matthews' CALC**. LLL, MLLL, Fincke-Pohst and lots more number-theoretical algorithms. Unsophisticated, quite accessible and easily installed. Freely available at

`http://www.maths.uq.edu.au/~krm/`

3. **LiDIA**. The most comprehensive, but tricky to install. LLL, MLLL, Fincke-Pohst but doesn't work with Gram matrices; needs an explicit list of vectors. Freely available from Darmstadt at

`http://www.informatik.tu-darmstadt.de/TI/LiDIA/`

4. **Pate Williams** has programmed many of the algorithms in Cohen's book, including LLL (no MLLL or Fincke-Pohst).

`http://www.mindspring.com/~pate/`

He uses Arjen Lenstra's LIP code for large integer arithmetic in C, which is hard to read; e.g. `c=a+b;` is written as

```
zmul(a,b,&c);
```

5. I have written my own code for LLL and Fincke-Pohst in C++ using Owen Astrachan's code (1996) for arbitrary precision integer arithmetic. This came out a little before LIP. His `bigint.h` and `bigint.cc` are widely available over the WWW. This allows us to use `+`, `*`, `/`, `%` etc. in class BigInt.

Kreher's Komputations

Let G be a permutation group of degree v, and let A_{tk} be the 'incidence matrix' of G-orbits on t-subsets of points, versus G-orbits on k-subsets of points.

(The $(\mathcal{O}, \mathcal{O}')$-entry of A_{tk} equals the number of $B \in \mathcal{O}'$ containing a fixed $A \in \mathcal{O}$.)

G-invariant t-(v, k, λ) designs are equivalent to $(0, 1)$-solutions of

$$A_{t,k}\mathbf{x} = \lambda\mathbf{1}$$

which can be solved using LLL or MLLL.

This led Kreher et al. to discover many new designs.