The Shortest Vector Problem (Lattice Reduction Algorithms)

"Approximation Algorithms" by V. Vazirani, Chapter 27

- Problem statement, general discussion
- Lattices: brief introduction
- The Gauss' algorithm in $\mathbf{R}^2$
- Lower bound via Gram-Schmidt orthogonalization
- Gauss reduced basis, the lattice reduction algorithm
- The dual lattice and approximate No certificates for the shortest vector problem

The Shortest Vector Problem

Given n linearly independent vectors in $\mathbf{Q}^n$: $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n$, find the shortest vector, in L_2 norm, in the lattice $\mathcal{L}$ generated by these vectors. Here $\mathcal{L} = \{\lambda_1 \boldsymbol{a}_1 + \ldots + \lambda_n \boldsymbol{a}_n \mid \lambda_i \in \mathbf{Z}\}$.

Remark 1. We're in $\mathbf{R}^n$, of course, but having rational coordinates is essential because we want to express efficiency of our algorithms in the length of input.

Remark 2. We will consider only full rank lattices. Dealing with those that do not span the entire space is, in general, more involved.

We will present an exponential in n factor approximation algorithm for this problem that runs in time polynomial in n and the input length. This may not sound impressive, but finding a polynomial factor algorithm is a long-standing open problem.

M. Ajtai [1997] showed that the shortest vector problem is *NP*-hard for randomized reductions. That is, there is a probabilistic Turing machine that in polynomial time reduces any problem in *NP* to instances of the shortest vector problem. Given an oracle solving the shortest vector problem (the input being a basis of the corresponding lattice), this machine solves in polynomial time any problem in *NP* with probability at least 1/2. Even a (very) good approximate solution to the shortest vector problem would be sufficient for solving (in the probabilistic sense again) any problem in *NP*.

The lattice reduction algorithm has numerous applications in computational number theory and cryptography. In particular, even the weak approximation guarantee that we have is sufficient to break certain cryptographic primitives under certain circumstances. Thus, understanding the lattice reduction techniques and related attacks is important in the design of cryptographic primitives.

Example. The problem of solving bivariate integer polynomial equations seems to be hard. Letting $P(x, y)$ be a polynomial in two variables with integer coefficients,
$P(x, y) = \sum_{i,j} p_{ij} x^i y^j$
it consists in finding all integer pairs (x_0, y_0) such that $P(x_0, y_0) = 0$. Obviously, integer factorization is a special case of this as one can take $P(x, y) = N - x y$.

D. Coppersmith [1995] showed that using LLL, the problem of finding small roots of bivariate polynomial equations is easy:
Theorem. Let $P(x, y)$ be an irreducible polynomial over **Z**, of maximum degree δ in each variable separately. Let X and Y be upper bounds on the desired integer solution (x_0, y_0), and let $W = \max_{i,j} | p_{ij} | X^i Y^j$. If $XY < W^{2/(3\delta)}$, then in time polynomial in $(\log W, 2^{\delta})$, one can find all integer pairs (x_0, y_0) such that $P(x_0, y_0) = 0$, $| x_0| \le X$, and $| y_0| \le Y$.

This can be used to factor in polynomial-time an RSA-modulus $n = pq$ such that half of the least significant or most significant bits of p are known.

Lattices: bases and determinants

Given a lattice $\mathcal{L}$ basis $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n$, let A denote the n x n matrix whose rows are the basis vectors. If vectors $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n$ belong to $\mathcal{L}$ and B is the corresponding matrix, we can write $B = \Lambda A$, where Λ is an n x n integer matrix. Thus, det(B) is a multiple of det(A).

A square matrix M with integer entries and such that | det(M)| = 1 is called *unimodular*. Its inverse is obviously also unimodular.

Theorem. Let vectors $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n$ belong to $\mathcal{L}$. The following conditions are equivalent:

1. $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n$ form a basis of $\mathcal{L}$
2. | det(B)| = | det(A)|
3. there is an n x n unimodular matrix U such that $B = U A$.

Proof: 1 => 2 => 3 => 1.

So, the determinant of a basis for $\mathcal{L}$ is invariant, up to sign. We'll call | det(A)| the *determinant of lattice* $\mathcal{L}$ and denote it by det $\mathcal{L}$.

Observation. We can move from basis to basis by applying unimodular transforms.

Observation. The most desirable basis to obtain is an orthogonal basis, as its shortest vector is also the shortest vector of the lattice. However, not every lattice admits such a basis.

Hadamard's inequality: for any n x n real matrix A, $|\det(A)| \leq \|\boldsymbol{a}_1\| \ldots \|\boldsymbol{a}_n\|$, and this holds with equality iff either one of the rows is the zero vector or the rows are mutually orthogonal.

Applying it to any lattice basis $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n$, we get $\det \mathcal{L} \leq \|\boldsymbol{b}_1\| \ldots \|\boldsymbol{b}_n\|$.

We define the *orthogonality defect* of basis $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n$ to be $\|\boldsymbol{b}_1\| \ldots \|\boldsymbol{b}_n\| / \det \mathcal{L}$.
Intuition: the smaller the orthogonality defect of a basis, the shorter its vectors must be. The ideal is reached for orthogonal bases.

We say that linearly independent vectors $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_k$ are primitive if they can be extended to a basis of $\mathcal{L}$.

Theorem. Vector $\boldsymbol{a} \in \mathcal{L}$ is primitive iff $\boldsymbol{a}$ is shortest in its direction.

The Gauss' algorithm (shortest vector in $\mathbf{R}^2$)

In $\mathbf{R}^2$ a weaker condition than orthogonality suffices for ensuring that a basis contains a shortest vector. Let φ denote the angle between the basis vectors $\boldsymbol{b}_1$, $\boldsymbol{b}_2$, $0 < \varphi < 180$.
Thus, $\det \mathcal{L} = ||\boldsymbol{b}_1||\ ||\boldsymbol{b}_2|| \sin \varphi$.

Theorem. If $||\boldsymbol{b}_1|| \le ||\boldsymbol{b}_2||$ and $60 \le \varphi \le 120$, then $\boldsymbol{b}_1$ is a shortest vector in the lattice.

Define $\mu_{21} = (\boldsymbol{b}_1, \boldsymbol{b}_2) / ||\boldsymbol{b}_1||^2$. [Here, $(\boldsymbol{b}_1, \boldsymbol{b}_2)$ is inner product of $\boldsymbol{b}_1$ and $\boldsymbol{b}_2$.]

Note that $\mu_{21}\, \boldsymbol{b}_1$ is the component of $\boldsymbol{b}_2$ in the direction of $\boldsymbol{b}_1$.

Proposition. If basis $(\boldsymbol{b}_1, \boldsymbol{b}_2)$ satisfies $||\boldsymbol{b}_1|| \le ||\boldsymbol{b}_2||$ and $|\mu_{21}| \le 1/2$, then $60 \le \varphi \le 120$.

The original algorithm by Gauss.

While the conditions of the above Proposition are not meet, do
(a) If $|\mu_{21}| > 1/2$, let $\boldsymbol{b}_2 \leftarrow \boldsymbol{b}_2 - m\, \boldsymbol{b}_1$, where m is the integer closest to μ_{21}.
(b) If $||\boldsymbol{b}_1|| > ||\boldsymbol{b}_2||$, interchange $\boldsymbol{b}_1$ and $\boldsymbol{b}_2$.

Output $\boldsymbol{b}_1$.

Although the algorithm clearly works, proving its efficiency is hard. The following careful modification, due to Kaib and Schnorr, makes it possible to prove a polynomial bound on the running time.

We say that basis $(\boldsymbol{b}_1, \boldsymbol{b}_2)$ is *well ordered* if $\|\boldsymbol{b}_1\| \le \|\boldsymbol{b}_1 - \boldsymbol{b}_2\| \le \|\boldsymbol{b}_2\|$.
If $\|\boldsymbol{b}_1\| \le \|\boldsymbol{b}_2\|$, then one of the three bases $(\boldsymbol{b}_1, \boldsymbol{b}_2)$, $(\boldsymbol{b}_1, \boldsymbol{b}_1 - \boldsymbol{b}_2)$, and $(\boldsymbol{b}_2 - \boldsymbol{b}_1, \boldsymbol{b}_2)$ is well ordered.

The enhanced algorithm.

Start with a well ordered basis.

(a) If $|\mu_{21}| > 1/2$, let $\boldsymbol{b}_2 \leftarrow \boldsymbol{b}_2 - m\,\boldsymbol{b}_1$, where m is the integer closest to μ_{21}.
(b) If $(\boldsymbol{b}_1, \boldsymbol{b}_2) < 0$, let $\boldsymbol{b}_2 \leftarrow -\boldsymbol{b}_2$.
(c) If $\|\boldsymbol{b}_1\| > \|\boldsymbol{b}_2\|$, make a well ordered basis from $\boldsymbol{b}_1$, $\boldsymbol{b}_2$, $\boldsymbol{b}_1 - \boldsymbol{b}_2$, and go to step (a).
Otherwise, output $\boldsymbol{b}_1$ and halt.

We can show that each iteration, after which we don't terminate, reduces the length of the basis' longest vector by a factor of at least 3 / 2 sqrt(2). That's enough to show that the number of iterations is bounded by a polynomial in the input length.

Lower bounding the shortest vector length (OPT) via Gram-Schmidt orthogonalization

Gram-Schmidt orthogonalization is a process of transforming a given basis $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n$ into orthogonal one $\boldsymbol{b}_1^*, \ldots, \boldsymbol{b}_n^*$, where $\boldsymbol{b}_1^* = \boldsymbol{b}_1$, and $\boldsymbol{b}_i^*$ is the component of $\boldsymbol{b}_i$ orthogonal to $\boldsymbol{b}_1^*, \ldots, \boldsymbol{b}_{i-1}^*$.
Here is what we do for $i \geq 2$:

$$\boldsymbol{b}_i^* = \boldsymbol{b}_i - \sum_{j=1}^{i-1} [(\boldsymbol{b}_i, \boldsymbol{b}_j^*) / ||\boldsymbol{b}_j^*||^2] \boldsymbol{b}_j^*$$

Remark. The Gram-Schmidt orthogonalization depends not only on the basis chosen, but also on the order of the basis vectors.

For $1 \leq j < i \leq n$, we define $\mu_{ij} = (\boldsymbol{b}_i, \boldsymbol{b}_j^*) / ||\boldsymbol{b}_j^*||^2$, and define $\mu_{ii} = 1$.

Then $\boldsymbol{b}_i = \sum_{j=1}^{i} \mu_{ij} \boldsymbol{b}_j^*$.

Remark. Subspaces generated by $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_k$ and $\boldsymbol{b}_1^*, \ldots, \boldsymbol{b}_k^*$ are the same for any k.

For $j \le i$, we define $\boldsymbol{b}_i(j)$ to be the component of $\boldsymbol{b}_i$ orthogonal to $\boldsymbol{b}_1^*, \ldots, \boldsymbol{b}_{j-1}^*$ (and also, clearly, to $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_{j-1}$) , that is,

$$\boldsymbol{b}_i(j) = \mu_{ij}\, \boldsymbol{b}_j^* + \mu_{i,j+1}\, \boldsymbol{b}_{j+1}^* + \ldots + \boldsymbol{b}_i^*$$

Obviously, $\det \mathcal{L} = \|\boldsymbol{b}_1^*\| \ldots \|\boldsymbol{b}_n^*\|$.

Lemma. Let $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n$ be a basis for lattice $\mathcal{L}$, and $\boldsymbol{b}_1^*, \ldots, \boldsymbol{b}_n^*$ be the Gram-Schmidt orthogonalization for it. Then

$$\mathrm{OPT} \ge \min \{\|\boldsymbol{b}_1^*\|, \ldots, \|\boldsymbol{b}_n^*\|\}.$$

Proof: express a shortest vector v via $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n$, and let k be the largest index, such that $\boldsymbol{b}_k$ is in the expression, and then express via $\boldsymbol{b}_1^*, \ldots, \boldsymbol{b}_n^*$ …

Gauss reduced bases, the Lattice Reduction Algorithm (LLL)

The LLL algorithm (by Lenstra, Lenstra, and Lovasz) is based on two ideas:

- the reduction technique of Gauss (the algorithm for $\mathbf{R}^2$);
- ensuring that the lengths of the corresponding Gram-Schmidt basis's vectors do not decrease too fast, thus, placing a lower bound on the length of shortest of them.

We will say that basis $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n$ is *Gauss reduced* if for $i < n$
$\|\boldsymbol{b}_i(i)\|^2 \leq 4/3 \, \|\boldsymbol{b}_{i+1}(i)\|^2$ and
$|\mu_{i+1,i}| \leq 1/2$.

The Lattice Reduction Algorithm (LLL).

While basis $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n$ is not Gauss reduced, do

(a) For each $i < n$, ensure that $|\mu_{i+1,i}| \leq 1/2$.
If $|\mu_{i+1,i}| > 1/2$, then $\boldsymbol{b}_{i+1} \leftarrow \boldsymbol{b}_{i+1} - m \, \boldsymbol{b}_i$, where m is the integer closest to $\mu_{i+1,i}$.

(b) Pick any i such that $\|\boldsymbol{b}_i(i)\|^2 > 4/3 \, \|\boldsymbol{b}_{i+1}(i)\|^2$, and interchange $\boldsymbol{b}_{i+1}$ and $\boldsymbol{b}_i$.

Output $\boldsymbol{b}_1$.

Theorem. The LLL algorithm terminates in a polynomial number of iterations (in n and the input length) and achieves an approximation guarantee of $2^{(n-1)/2}$.

Proof (sketch):
Proving the approximation guarantee is straightforward.
To upper bound the number of iterations:
- Work with integer bases.
- Introduce potential function $\Phi = \prod_{i=1}^{n} \|\boldsymbol{b}_i^*\|^{(n-i)}$.
- Show that the initial value of Φ is upper bounded by a polynomial in the input length, Φ is always lower bounded by 1, and each execution of step (b) reduces the value of Φ by a factor of at least 2/sqrt(3).

Now, strictly speaking, bounding the number of iterations is not exactly what we want. We actually want to bound the number of bit operations, that is, to show that the numbers being handled by the algorithm can be written using a polynomial number of bits.

The stronger notion of a *Lovasz reduced* basis and appropriate extension of the LLL algorithm make it possible to prove a polynomial upper bound on the number of bit operations.

We say that basis $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n$ is *weakly reduced* if for $1 \le j \le i \le n$: $|\mu_{i,j}| \le 1/2$.

The basis is said to be Lovasz reduced if it is Gauss reduced and weakly reduced.

It is possible to obtain a weakly reduced basis from a given basis, without changing its Gram-Schmidt orthogonalization, using at most $n(n-1)/2$ arithmetic operations, and we simply need to use that procedure instead of step (a) of the original LLL algorithm.

The dual lattice and existential factor *n* approximate No certificates

The *dual lattice*, $\mathcal{L}^*$, is defined by $\mathcal{L}^* = \{\boldsymbol{v} \in \mathbf{R}^n \mid \text{for any } \boldsymbol{b} \in \mathcal{L}, (\boldsymbol{b}, \boldsymbol{v}) \in \mathbf{Z}\}$.
If B is the matrix of a basis $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n$, then $(B^{-1})^T$ is the corresponding basis matrix for the dual lattice.

There are techniques based on appropriate selecting of primitive vectors in dual lattices that allow constructing (though not efficient!) of bases for primal lattices with "good" Gram-Schmidt orthogonalizations.

Theorem. There is a basis for $\mathcal{L}$ whose Gram-Schmidt lower bound is at least OPT/n.

Thus, we have factor *n* approximate No certificates for the shortest vector problem. However, we do not know how to construct those efficiently.