MPRI 2-12-1: Phong Nguyen **Fall 2024**

Lecture 3: Hard Lattice Problems

We introduce hard lattice problems.

1 The Shortest Vector Problem

The most famous lattice problem is the so-called *shortest vector problem* (SVP), which asks to find a shortest non-zero vector in an integer lattice L given by a basis, that is, a non-zero vector of the form $a_1\mathbf{b}_1 + \cdots + a_n\mathbf{b}_n$ (where $a_i \in \mathbb{Z}$) and of minimal Euclidean norm $\lambda_1(L)$. SVP can be viewed as a geometric generalization of gcd computations: Euclid's algorithm actually computes the smallest (in absolute value) non-zero linear combination of two integers, since $\gcd(a, b)\mathbb{Z} = a\mathbb{Z} + b\mathbb{Z}$, which means that we are replacing the integers a and b by an arbitrary number of vectors $\mathbf{b}_1, \ldots, \mathbf{b}_n$ with integer coordinates.

1.1 Variants

We introduce the main variants of SVP, but there are many others, such as one changes the norm, replacing the Euclidean norm by a different norm.

Problem 1.1 (Decisional-SVP) *Given as input a basis of an integer lattice L, and an integer t, decide if $\lambda_1(L)^2 \leq t$.*

Problem 1.2 (Optimization-SVP) *Given as input a basis of an integer lattice L, return the integer $\lambda_1(L)^2$.*

Clearly, Optimization-SVP is at least as hard as Decisional-SVP. Reciprocally, using dichotomy, if we had an oracle solving Decisional-SVP in polynomial time, we could also solve Optimization-SVP in polynomial time. So Decisional-SVP is as hard as Optimization-SVP.

It can be proved that Optimization-SVP is equivalent to SVP.

GapSVP is the approximate version of Decisional-SVP, which is used to study the hardness of approximating SVP:

Problem 1.3 (GapSVP) *Given as input a basis of an integer lattice L and two rational numbers t and γ, decide if $\lambda_1(L) \leq t$ or $\lambda_1(L) > t\gamma$. Here, we are promised to be in one of the two previous cases: there is no constraint on the answer if $t < \lambda_1(L) \leq t\gamma$.*

There are natural approximate versions of SVP:

Problem 1.4 (Approx-SVP) *Given as input a basis of an integer lattice L and an approximation factor $\gamma \geq 1$, find a non-zero $\mathbf{u} \in L$ such that $\|\mathbf{u}\| \leq \gamma\lambda_1(L)$.*

Problem 1.5 (Hermite-SVP) *Given as input a basis of a d-rank integer lattice L and an approximation factor $\gamma \geq 1$, find a non-zero $\mathbf{u} \in L$ such that $\|\mathbf{u}\| \leq \gamma\mathrm{covol}(L)^{1/d}$.*

Hermite-SVP is different from SVP and all previous variants in the sense that its result can be checked: the inequality $\|\mathbf{u}\| \leq \gamma\mathrm{covol}(L)^{1/d}$ can be checked in polynomial time.

Obviously, if one can solve Approx-SVP with factor γ, then one can solve Hermite-SVP with factor $\gamma\sqrt{\gamma_d}$. Surprisingly, Lovász [8] proved that if one can solve Hermite-SVP with a non-decreasing factor γ, then one can solve Approx-SVP with factor γ^2:

Theorem 1.6 *Assume that one can solve Hermite-SVP with factor $f(d)$ where d is the input lattice rank, then one can solve Approx-SVP in polynomial time with factor $\max_{1 \leq i \leq d} f(i)^2$.*

Proof. One is given a basis $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ of a lattice L and a HSVP oracle $\mathcal{O}$. Call the oracle twice on L and $L^\times$ to obtain $\mathbf{c}_1 \in L$ and $\mathbf{d}_1 \in L^\times$ such that $\|\mathbf{c}_1\| \leq f(d)\mathrm{covol}(L)^{1/d}$ and $\|\mathbf{d}_1\| \leq f(d)\mathrm{covol}(L^\times)^{1/d}$. Thus $\|\mathbf{c}_1\| \times \|\mathbf{d}_1\| \leq f(d)^2$. Let $L_2 = L \cap \mathrm{span}(\mathbf{d}_1)^\perp$. Call the oracle twice on L_2 and $L_2^\times$ to obtain $\mathbf{c}_2 \in L_2$ and $\mathbf{d}_2 \in L_2^\times$ such that $\|\mathbf{c}_2\| \leq f(d-1)\mathrm{covol}(L_2)^{1/(d-1)}$ and $\|\mathbf{d}_2\| \leq f(d-1)\mathrm{covol}(L_2^\times)^{1/(d-1)}$. Thus $\|\mathbf{c}_2\| \times \|\mathbf{d}_2\| \leq f(d-1)^2$. By iterating with the lattice $L_i = L \cap \mathrm{span}(\mathbf{d}_1, \ldots, \mathbf{d}_{i-1})^\perp$ and its dual, we obtain $\mathbf{c}_1, \ldots, \mathbf{c}_d) \in L$ and $\mathbf{d}_1, \ldots, \mathbf{d}_d) \in \mathbb{R}^n$ such that $\|\mathbf{c}_i\| \times \|\mathbf{d}_i\| \leq f(d-i+1)^2$ for all $1 \leq i \leq d$.

Let us show that $1/\max_{1 \leq i \leq d} \|\mathbf{d}_i\| \leq \lambda_1(L)$. We note that the $\mathbf{d}_i$'s are pairwise orthogonal because $\mathbf{d}_i \in \mathrm{span}(\mathbf{d}_1, \ldots, \mathbf{d}_{i-1})^\perp$. Let $\mathbf{u} \in L$ such that $\|\mathbf{u}\| = \lambda_1(L)$. We claim that there exists $i \in \{1, \ldots, d\}$ such that $\langle \mathbf{u}, \mathbf{d}_i \rangle \in \mathbb{Z} \setminus \{0\}$. Since $\mathbf{d}_1 \in L^\times$, we know that $\langle \mathbf{u}, \mathbf{d}_1 \rangle \in \mathbb{Z}$. If $\langle \mathbf{u}, \mathbf{d}_1 \rangle \neq 0$, we are done. Otherwise, $\langle \mathbf{u}, \mathbf{d}_1 \rangle = 0$, which means that $\mathbf{u} \in L_2$. Since $\mathbf{d}_2 \in L_2^\times$, we know that $\langle \mathbf{u}, \mathbf{d}_2 \rangle \in \mathbb{Z}$. If $\langle \mathbf{u}, \mathbf{d}_2 \rangle \neq 0$, we are done. Otherwise, we iterate the process, and we note that $\mathbf{u}$ cannot be orthogonal to all the

$\mathbf{d}_i$'s, since the $\mathbf{d}_i$ span span(L). So there must be an index i such that $\mathbf{u} \in L_i$ and $\langle \mathbf{u}, \mathbf{d}_i \rangle \in \mathbb{Z} \setminus \{0\}$.

Therefore $|\langle \mathbf{u}, \mathbf{d}_i \rangle| \geq 1$, which implies by Cauchy-Schwarz that $1/\|\mathbf{d_i}\| \leq \|\mathbf{u}\| = \lambda_1(L)$. Hence $\min_{1 \leq i \leq d} \ 1/\|\mathbf{d_i}\| \leq \lambda_1(L)$.

We choose $\mathbf{c} \in L$ among the $\mathbf{c}_i$'s such that $\mathbf{d}_i$'s is maximized. Then both $1/\|\mathbf{d_i}\| \leq \lambda_1(L)$ and $\|\mathbf{c}_i\| \times \|\mathbf{d}_i\| \leq f(d-i+1)^2$. It follows that $\|\mathbf{c}\| \leq \max_{1 \leq i \leq d} f(i)^2$. □

1.2 Complexity results

SVP was conjectured NP-hard as early as 1981 [5] (see also [8]). Ajtai showed NP-hardness under randomized reductions in 1998 [2]: this implies that if we could solve SVP efficiently, then we would also have an efficient randomized algorithm for any problem in NP, which is considered unlikely. However, it is a long-standing open problem to prove that SVP is NP-hard under deterministic reductions: surprisingly, NP-hardness under deterministic reductions is known for the analogue problem in coding theory. The best result so far suggests that it is unlikely that one can efficiently approximate SVP to within quasi-polynomial factors. But NP-hardness results have limits: essentially, approximating SVP within a factor $\sqrt{d/\log d}$ is unlikely to be NP-hard. More precisely, Aharonov and Regev [1] showed that there exists a constant c such that approximating SVP with a factor $c\sqrt{d}$ is in the l'intersection NP∩coNP, while Goldreich and Goldwasser [6] showed that each constant c, approximating SVP with a factor $c\sqrt{d/\log d}$ is in NP∩coAM.

2 The Closest Vector Problem

The *closest vector problem* (CVP) is the inhomogeneous variant of SVP: given an integer lattice $L \subseteq \mathbb{Z}^n$ given by a basis and a target $\mathbf{t} \in \mathbb{Z}^n$, find $\mathbf{u} \in L$ minimizing $\|\mathbf{u} - \mathbf{t}\|$. This can be viewed as some sort of higher-dimensional generalization of the Euclidean division. SVP is not a special case of CVP: if one asks for the closest vector to $\mathbf{t} = 0$, the answer is zero. However, Goldreich *et al.* [7] observed that SVP is not harder than CVP: if one can solve CVP, then one can also solve SVP efficiently.

Theorem 2.1 ([7]) *Given an oracle for CVP, one can solve SVP in polynomial time.*

Proof. We are given a basis $(\mathbf{b}_1, \ldots, \mathbf{b}_n)$ of a lattice L, and we want to find $\mathbf{x} \in L$ such that $\|\mathbf{x}\| = \lambda_1(L)$. There exist $x_1, \ldots, x_n \in \mathbb{Z}$ such that

$\mathbf{x} = \sum_{i=1}^{n} x_i \mathbf{b}_i$. For $1 \leq i \leq n$, let L_i be the sublattice generated by the $2\mathbf{b}_i$ and the $\mathbf{b}_j$'s for $j \neq i$. We note that there must exist an index i such that x_i is odd, otherwise $\mathbf{x}/2$ would be a non-zero lattice vector shorter than $\mathbf{x}$. Then $x_i = 2y_i + 1$ for some integer y_i. Therefore $\mathbf{x} = \mathbf{y} + \mathbf{b}_i$ where $\mathbf{y} = y_i(2\mathbf{b}_i) + \sum_{j \neq i} x_j \mathbf{b}_j \in L_i$. So $\mathbf{x} \in \mathbf{b}_i + L_i$. By definition, $\mathbf{b}_i \notin L_i$ so $\mathbf{b}_i + L_i \subseteq L \setminus \{0\}$. It follows that $\mathbf{x}$ is a shortest element of $\mathbf{b}_i + L_i$, which means that $\mathbf{y} \in L_i$ is a closest vector to $-\mathbf{b}_i$.

We deduce the following algorithm: for every index i, call the CVP oracle on L_i and target $-\mathbf{b}_i$. This outputs a closest vector $\mathbf{y}_i \in L_i$. Compute $\mathbf{x}_i = \mathbf{y}_i + \mathbf{b}_i$ and output the shortest vector among the $\mathbf{x}_i$'s. □

Most variants of SVP can be adapted to CVP:

Problem 2.2 (Approx-CVP) *Given as input a basis of an integer lattice $L \subseteq \mathbb{Z}^n$, a target $\mathbf{t} \in \mathbb{Z}^n$, and an approximation factor $\gamma \geq 1$, find $\mathbf{u} \in L$ such that $\|\mathbf{u} - \mathbf{t}\| \leq \gamma \operatorname{dist}(\mathbf{t}, L)$.*

Problem 2.3 (Decisional-CVP) *Given as input a basis of an integer lattice $L \subseteq \mathbb{Z}^n$, a target $\mathbf{t} \in \mathbb{Z}^n$, and an integer k, decide if $\operatorname{dist}(\mathbf{t}, L)^2 \leq k$.*

Problem 2.4 (Optimization-CVP) *Given as input a basis of an integer lattice $L \subseteq \mathbb{Z}^n$, a target $\mathbf{t} \in \mathbb{Z}^n$, return the integer $\operatorname{dist}(\mathbf{t}, L)^2$.*

Similarly to SVP, Decisional-CVP is as hard as Optimization-CVP. And it is very easy to show that Decisional-CVP is NP-complete:

Theorem 2.5 *The subset sum problem can be reduced to Decisional-CVP in deterministic polynomial time.*

Proof. We are given $s, a_1, \ldots, a_n \in \mathbb{Z}$ and we want to know if s is of the form $s = \sum_{i=1}^{n} x_i a_i$ where $x_i \in \{0, 1\}$. We consider the lattice L spanned by the rows of the following $n \times (n+1)$ matrix:

$$B = \begin{pmatrix} a_1 & 2 & 0 & \ldots & 0 \\ a_2 & 0 & 2 & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ a_n & 0 & \ldots. & 0 & 2 \end{pmatrix}$$

Any vector of L is of the shape $\mathbf{x} = (\sum_{i=1} n x_i a_i, 2x_1, \ldots, 2x_n)$ where $x_i \in \mathbb{Z}$. One can check that s is of the form $s = \sum_{i=1}^{n} x_i a_i$ where $x_i \in \{0, 1\}$ if and only if there exists $\mathbf{x} \in L$ such that $\|\mathbf{x} - \mathbf{t}\| \leq \sqrt{n}$ where $\mathbf{t} = (s, 1, 1, \ldots, 1)$. □

Theorem 2.6 *If one can solve Optimization-CVP efficiently, then one can solve CVP in polynomial time.*

Proof. We are given a CVP instance: a basis $(\mathbf{b}_1, \ldots, \mathbf{b}_n)$ of a lattice L, and a target $\mathbf{t}$. We want to find $\mathbf{x} \in L$ such that $\|\mathbf{x} - \mathbf{t}\| = \operatorname{dist}(\mathbf{t}, L)$, but there could be many solutions. There exist $x_1, \ldots, x_n \in \mathbb{Z}$ such that $\mathbf{x} = \sum_{i=1}^{n} x_i \mathbf{b}_i$: the size of the x_i's can be bounded by the same polynomial bound, for all solutions $\mathbf{x}$.

Let M_i be the sublattice of L generated by the $\mathbf{b}_j$'s for $j \neq i$. First, we call the optimization oracle if $\operatorname{dist}(\mathbf{t}, L) = \operatorname{dist}(\mathbf{t}, M_1)$. If it is, then we know that there is a solution with $x_1 = 0$, so we can replace L by M_1 and decrease the lattice rank by 1. Otherwise, we're going to recover x_1 bit by bit. Like in the proof of Th. 2.1, for $1 \leq i \leq n$, let L_i be the sublattice generated by the $2\mathbf{b}_i$ and the $\mathbf{b}_j$'s for $j \neq i$. We call the optimization oracle to check if $\operatorname{dist}(\mathbf{t}, L) = \operatorname{dist}(\mathbf{t}, L_1)$:

- If $\operatorname{dist}(\mathbf{t}, L) = \operatorname{dist}(\mathbf{t}, L_1)$, we know that there is a solution $\mathbf{x}$ such that x_1 is even, so we replace L by L_1.
- Otherwise, we know that for all closest vectors, the x_1 coordinate is odd. Then $x_1 = 2y_1 + 1$ for some integer y_1. Therefore $\mathbf{x} = \mathbf{y} + \mathbf{b}_1$ where $\mathbf{y} = y_1(2\mathbf{b}_1) + \sum_{j \neq i} x_j \mathbf{b}_j \in L_1$. So $\mathbf{x} \in \mathbf{b}_1 + L_1$. Thus, we replace $(\mathbf{t}, L)$ by $(\mathbf{t} - \mathbf{b}_1, L_1)$.

In both cases, we have removed one bit of x_1 for at least one solution. Since all x_1's have (uniform) polynomial size, we must eventually recover the full x_1 of one solution. After that, we do the same iteratively for all the remaining coordinates $x_2, x_3, \ldots, x_n$. □

GapCVP is the approximate version of Decisional-CVP, which is used to study the hardness of approximating CVP:

Problem 2.7 (GapCVP) *Given as input a basis of an integer lattice $L \subseteq \mathbb{Z}^n$, a target $\mathbf{t} \in \mathbb{Z}^n$, and two rationals k and γ, decide if $\operatorname{dist}(\mathbf{t}, L) \leq k$ or $\operatorname{dist}(\mathbf{t}, L) > k\gamma$. Here, we are promised to be in one of the two previous cases.*

An important special case of CVP is the following:

Problem 2.8 (Bounded Distance Decoding (BDD)) *Given as input a basis of an integer lattice $L \subseteq \mathbb{Z}^n$, a target $\mathbf{t} \in \mathbb{Z}^n$ such that $\operatorname{dist}(\mathbf{t}, L) \leq c\lambda_1(L)$ for some given $c < 1/2$, find the closest lattice point to $\mathbf{t}$.*

Because $\operatorname{dist}(\mathbf{t}, L) < \lambda_1(L)/2$, the solution is unique.

There are many other lattice problems, but we introduced the main ones. Complexity theory studies the relationships between these problems.

3 Low Dimension

The most interesting lattice problems become very hard as the dimension increases. However, in fixed dimension, most problems are easy: they can be solved in time polynomial in the size of the input basis.

3.1 Dimension 1

SVP is trivial in dimension 1, since a one-dimensional basis is unique, up to sign. For CVP, we have the following elementary result:

Lemma 3.1 *Let L be a 1-rank lattice generated by $\mathbf{b} \in \mathbb{R}^n$. Then a closest lattice vector to $\mathbf{t} \in \mathbb{R}^n$ is given by:*

$$\mathbf{u} = \lfloor \frac{\langle \mathbf{t}, \mathbf{b} \rangle}{\|\mathbf{b}\|^2} \rceil vecb.$$

If $\mathbf{b}$ and $\mathbf{t}$ are in $\mathbb{Z}^n$, then $\mathbf{u}$ can be computed in deterministic polynomial time.

Proof. Let $x \in \mathbb{Z}$. Then:

$$\|\mathbf{t} - x\mathbf{b}\|^2 = \|\mathbf{t}\|^2 + x^2\|\mathbf{b}\|^2 - 2x\langle \mathbf{t}, \mathbf{b} \rangle.$$

Thus, $\|\mathbf{t} - x\mathbf{b}\|$ is minimized when $(x - \frac{\langle \mathbf{t}, \mathbf{b} \rangle}{\|\mathbf{b}\|^2})^2$ is minimized, which proves the claim. □

3.2 Dimension 2

In dimension 2, there's a natural SVP algorithm published by Lagrange in 1773, which is very similar to Euclid's algorithm. We start by defining reduction in the sense of Lagrange:

Definition 3.2 (Lagrange reduction) *Let L be a two-rank lattice. A basis $(\mathbf{u}, \mathbf{v})$ of L is* Lagrange-reduced *if $\|\mathbf{u}\| \leq \|\mathbf{v}\|$ and $|\langle \mathbf{u}, \mathbf{v} \rangle| \leq \|\mathbf{u}\|^2/2$.*

Lagrange bases reach both minima:

Theorem 3.3 *Let $(\mathbf{u}, \mathbf{v})$ be a Lagrange-reduced basis of L. Then: $\|\mathbf{u}\| = \lambda_1(L) \leq (4/3)^{1/4}\mathrm{covol}(L)^{1/2}$ and $\|\mathbf{v}\| = \lambda_2(L)$.*

Proof. Let $(a, b) \in \mathbb{Z}^2$ be non zero. Then:

$$\begin{aligned}\|a\mathbf{u} + b\mathbf{v}\|^2 &= a^2\|\mathbf{u}\|^2 + b^2\|\mathbf{v}\|^2 + 2ab\langle\mathbf{u}, \mathbf{v}\rangle \\ &\geq (a^2 + b^2 - |ab|)\|\mathbf{u}\|^2 \\ &\geq \|\mathbf{u}\|^2\end{aligned}$$

This proves that $\|\mathbf{u}\| = \lambda_1(L)$. Furthermore, by definition of the Gram matrix and Lagrange-reduction:

$$\mathrm{covol}(L)^2 = \|\mathbf{u}\|^2\|\mathbf{v}\|^2 - \langle\mathbf{u}, \mathbf{v}\rangle^2 \geq (1 - 1/4)\|\mathbf{u}\|^4 = \frac{3}{4}\|\mathbf{u}\|^4.$$

Hence:

$$\|\mathbf{u}\| \leq (4/3)^{1/4}\mathrm{covol}(L)^{1/2}.$$

We also proved:

$$\|a\mathbf{u} + b\mathbf{v}\|^2 \geq b^2\|\mathbf{v}\|^2 + (a^2 - |ab|)\|\mathbf{u}\|^2.$$

Assume that $b \neq 0$:

- If $|a| \geq |b|$ then $a^2 - |ab| \geq 0$, so $\|a\mathbf{u} + b\mathbf{v}\|^2 \geq b^2\|\mathbf{v}\|^2 \geq \|\mathbf{v}\|^2$.
- Otherwise, $|a| < |b|$, then $a^2 - |ab| \leq 0$ and:

  $$\|a\mathbf{u} + b\mathbf{v}\|^2 \geq (b^2 + a^2 - |ab|)\|\mathbf{v}\|^2$$

  Since $a^2 - |ab| \geq a^2 - |b|(|b| - 1)$, we get that:

  $$\|a\mathbf{u} + b\mathbf{v}\|^2 \geq (a^2 + |b|)\|\mathbf{v}\|^2 \geq \|\mathbf{v}\|^2.$$

Hence, we proved that any lattice point which is not in $\mathbb{Z}\mathbf{u}$ has norm $\geq \|\mathbf{v}\|$. We therefore proved: $\|\mathbf{v}\| = \lambda_2(L)$. □

In 1773, Lagrange published the following two-dimensional reduction algorithm (Alg. 1) to find a Lagrange-reduced basis. Lagrange's algorithm is a greedy algorithm: Line 5 of Algorithm 1 selects $q \in \mathbb{Z}$ minimizing $\|\mathbf{u} - q\mathbf{v}\|$, which means it find the closest vector to $\mathbf{u}$ in the lattice $\mathbb{Z}\mathbf{v}$.

Theorem 3.4 *Given as input any basis* $(\mathbf{u}, \mathbf{v})$ *of a two-rank lattice* L*, Lagrange's algorithm described in Fig. 1 outputs a Lagrange-reduced basis of the lattice* L *in time polynomial in* $\log\max(\|\mathbf{u}\|, \|\mathbf{v}\|)$.

Proof. Consider the integer q of Step 5. We observe that:

Algorithm 1 Lagrange's reduction algorithm.

Input: a basis $(\mathbf{u}, \mathbf{v})$ of a two-rank lattice L.
Output: a Lagrange-reduced basis of L.

1: **if** $\|\mathbf{u}\| < \|\mathbf{v}\|$ **then**
2: swap $\mathbf{u}$ and $\mathbf{v}$
3: **end if**
4: **repeat**
5: $\mathbf{r} \longleftarrow \mathbf{u} - q\mathbf{v}$ where $q = \left\lfloor \frac{\langle \mathbf{u}, \mathbf{v} \rangle}{\|\mathbf{v}\|^2} \right\rceil$ and $\lfloor x \rceil$ denotes an integer closest to x.
6: $\mathbf{u} \longleftarrow \mathbf{v}$
7: $\mathbf{v} \longleftarrow \mathbf{r}$
8: **until** $\|\mathbf{u}\| \leq \|\mathbf{v}\|$
9: Output $(\mathbf{u}, \mathbf{v})$.

- if $q = 0$, then this must be the last iteration of the loop, because the basis is reduced by definition.
- if $|q| = 1$, then this must be either the first or last iteration of the loop. Indeed, assume that it is not the last iteration: $\|\mathbf{u} - q\mathbf{v}\| < \|\mathbf{v}\| < \|\mathbf{u}\|$. But $\|\mathbf{u} - q\mathbf{v}\| = \| \mathbf{v} - q\mathbf{u}\|$ which means that $\mathbf{v}$ could be shortened with $\mathbf{u}$. Due to the greedy strategy, this can only happens at the first iteration.

Hence, except maybe the first and last iteration, we always have $|q| \geq 2$. Letting $\mu = \frac{\langle \mathbf{u}, \mathbf{v} \rangle}{\|\mathbf{v}\|^2}$, this means that $|\mu| \geq 3/2$. But $\mu\mathbf{v}$ is the projection of $\mathbf{u}$ over span$(\mathbf{u})$ so:

$$\|\mathbf{u}\|^2 \geq \|\mu\mathbf{v}\|^2 \geq \frac{9}{4}\|\mathbf{v}\|^2.$$

Therefore $\|\mathbf{v}\|^2 \leq \frac{4}{9}\|\mathbf{u}\|^2$. Thus, except maybe the first and last iteration, the norm decreases by at least a factor $4/9$, so the number of iterations is linear in $\log \max(\|\mathbf{u}\|, \|\mathbf{v}\|)$. □

3.3 High dimension

In high dimension n, there are two type of algorithms:

a) Exact algorithms. These algorithms provably find a shortest vector, but they are expensive, with a running time at least exponential in the dimension. There are two types of algorithms, depending on the

memory usage: polynomial-space algorithms and exponential-space algorithms. Exponential-space algorithms have the fastest running time known: the best exponential-space algorithms run in $2^{O(n)}$ polynomial-time operations. The first such algorithm was the sieve algorithm of Ajtai, Kumar and Sivakumar (AKS) [3, 9]. On the other hand, the best polynomial-space algorithms have super-exponential running time: in theory, the best polynomial-space algorithm known is Kannan's enumeration algorithm, which runs in time $n^{n/(2\pi e)+o(n)}$ polynomial-time operations.

b) Approximation algorithms. We will later see the LLL algorithm, which can approximate SVP within some exponential factor. All polynomial-time approximation algorithms known can be viewed as (more or less tight) algorithmic versions of upper bounds on Hermite's constant. For instance, LLL can be viewed as an algorithmic version of Hermite's inequality. Stronger algorithms correspond to Mordell's inequality.

References

[1] Dorit Aharonov and Oded Regev. Lattice problems in NP ∩ coNP. *J. ACM*, 52(5):749–765 (electronic), 2005.

[2] M. Ajtai. The shortest vector problem in L_2 is NP-hard for randomized reductions. In *Proc. of 30th STOC*. ACM, 1998. Available at [4] as TR97-047.

[3] M. Ajtai, R. Kumar, and D. Sivakumar. A sieve algorithm for the shortest lattice vector problem. In *Proc. 33rd STOC*, pages 601–610. ACM, 2001.

[4] ECCC. `http://www.eccc.uni-trier.de/eccc/`. The Electronic Colloquium on Computational Complexity.

[5] P. van Emde Boas. Another NP-complete problem and the complexity of computing short vectors in a lattice. Technical report, Mathematische Instituut, University of Amsterdam, 1981. Report 81-04. Available at `http://turing.wins.uva.nl/~peter/`.

[6] O. Goldreich and S. Goldwasser. On the limits of non-approximability of lattice problems. In *Proc. of 30th STOC*. ACM, 1998. Available at [4] as TR97-031.

[7] Oded Goldreich, Daniele Micciancio, Shmuel Safra, and Jean-Pierre Seifert. Approximating shortest lattice vectors is not harder than approximating closest lattice vectors. *Inf. Process. Lett.*, 71(2):55–61, 1999.

[8] L. Lovász. *An Algorithmic Theory of Numbers, Graphs and Convexity*, volume 50. SIAM Publications, 1986. CBMS-NSF Regional Conference Series in Applied Mathematics.

[9] P. Q. Nguyen and T. Vidick. Sieve algorithms for the shortest vector problem are practical. *J. of Mathematical Cryptology*, 2(2):181–207, 2008.