A Survey of Solving SVP Algorithms and Recent Strategies for Solving the SVP Challenge

Masaya Yasuda

Abstract Recently, lattice-based cryptography has received attention as a candidate of post-quantum cryptography (PQC). The essential security of lattice-based cryptography is based on the hardness of classical lattice problems such as the shortest vector problem (SVP) and the closest vector problem (CVP). A number of algorithms have been proposed for solving SVP exactly or approximately, and most of them are useful also for solving CVP. In this paper, we give a survey of typical algorithms for solving SVP from a mathematical point of view. We also present recent strategies for solving the Darmstadt SVP challenge in dimensions higher than 150.

Keywords Shortest vector problem (SVP) · Enumeration · Sieve · Lattice basis reduction · LLL · BKZ · Random sampling · Sub-sieving

1 Introduction

There has recently been a substantial amount of research for large-scale quantum computers. On the other hand, if such computers were built, they could break currently used public-key cryptosystems such as the RSA cryptosystem and the elliptic curve cryptography. (See Shor 1994 for Shor's quantum algorithms.) In order to prepare information security systems to be able to resist quantum computing, the US National Institute of Standards and Technology (NIST) began a process to develop new standards for PQC in 2015 and called for proposals in 2016. It has rapidly accelerated to research lattice-based cryptography as a candidate of PQC. Specifically, at the submission deadline of the end of November 2017 for the call, NIST received more than 20 proposals of lattice-based cryptosystems. Among them, more than 10 proposals were allowed for Round 2 submissions around the end of January 2019. (See the web page of NIST 2016.) The security of such proposals relies on the hard-

M. Yasuda (✉)
Institute of Mathematics for Industry, Kyushu University, 744 Motooka, Nishi-ku Fukuoka 819–0395, Japan
e-mail: yasuda@imi.kyushu-u.ac.jp

T. Takagi et al. (eds.), *International Symposium on Mathematics, Quantum Theory, and Cryptography*, Mathematics for Industry 33,
https://doi.org/10.1007/978-981-15-5191-8_15

ness of cryptographic lattice problems such as learning with errors (LWE) and NTRU. Such problems are reduced to approximate-SVP or approximate-CVP. (For example, see Albrecht et al. 2018 for details.) Therefore, it is becoming more important to understand classical lattice problems for evaluating the security of lattice-based PQC candidates.

For a positive integer n, a (full-rank) *lattice* L in $\mathbb{R}^n$ is the set of all integral linear combinations of linearly independent vectors $\mathbf{b}_1, \dots, \mathbf{b}_n$ in $\mathbb{R}^n$. (The set of the $\mathbf{b}_i$'s is called a *basis* of L.) Given a basis of a lattice L, SVP asks to find the non-zero shortest vector in L. In this paper, we give a survey of typical algorithms for solving SVP from a mathematical point of view. These algorithms can be classified into two categories, depending on whether they solve SVP exactly or approximately. Exact-SVP algorithms perform an exhaustive search for an integer combination of the basis vectors $\mathbf{b}_i$'s to find the non-zero shortest lattice vector $\mathbf{v} = \sum_{i=1}^{n} v_i \mathbf{b}_i \in L$, and their cost is expensive. In contrast, approximate-SVP algorithms are much faster than exact algorithms, but they find short lattice vectors, not necessarily the shortest ones. However, exact- and approximate-SVP algorithms are complementary. For example, exact algorithms apply an approximation algorithm as a preprocessing to reduce their expensive cost, while several approximate-SVP algorithms call many times an exact algorithm in low dimension as a subroutine to find a very short lattice vector. In this paper, we also introduce recent strategies for solving the Darmstadt SVP challenge Darmstadt (2010), in which sample lattice bases are presented in order to test algorithms solving SVP. In particular, these strategies combine approximate- and exact-SVP algorithms to efficiently solve SVP in high dimensions such as $n \geq 150$.

Notation. The symbols $\mathbb{Z}$, $\mathbb{Q}$, and $\mathbb{R}$ denote the ring of integers, the field of rational numbers, and the field of real numbers, respectively. Let $\lfloor z \rceil$ denote the rounding integer of an integer z. We represent all vectors in *column format*. For $\mathbf{a} = (a_1, \dots, a_n)^\top \in \mathbb{R}^n$, let $\|\mathbf{a}\|$ denote its Euclidean norm. For $\mathbf{a} = (a_1, \dots, a_n)^\top$ and $\mathbf{b} = (b_1, \dots, b_n)^\top$, let $\langle \mathbf{a}, \mathbf{b} \rangle$ denote the inner product $\sum_{i=1}^{n} a_i b_i$. Denote by $V_n(R)$ the volume of the n-dimensional ball of radius $R > 0$ centered at the origin. In particular, we let $\nu_n = V_n(1)$ denote the volume of the unit ball. Then $V_n(R) = \nu_n R^n$ and

$$\nu_n = \frac{\pi^{n/2}}{\Gamma(1 + n/2)} \sim \frac{1}{\sqrt{\pi n}} \left(\frac{2\pi e}{n} \right)^{n/2}$$

using Stirling's formula, where $\Gamma(s) = \int_0^\infty t^{s-1} e^{-t} dt$ denotes the Gamma function.

2 Mathematical Background

In this section, we introduce basic definitions and properties on lattices, and present famous lattice problems whose hardness ensures the essential security of lattice-based cryptography. (For example, see Galbraith 2012, Part IV or Nguyen 2009 for more details.)

2.1 Lattices and Their Bases

For a positive integer n, let $\mathbf{b}_1, \ldots, \mathbf{b}_n$ be n linearly independent (column) vectors in $\mathbb{R}^n$. The set of all integral linear combinations of the $\mathbf{b}_i$'s is a (full-rank) *lattice*

$$L = \mathcal{L}(\mathbf{b}_1, \ldots, \mathbf{b}_n) = \left\{ \sum_{i=1}^{n} v_i \mathbf{b}_i : v_i \in \mathbb{Z} \text{ for all } 1 \le i \le n \right\}$$

of dimension n with basis $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n) \in \mathbb{R}^{n \times n}$. (A basis is regarded not only as a set of vectors, but also as a matrix whose column vectors span a lattice.) Every lattice has infinitely many bases if $n \ge 2$; if two bases $\mathbf{B}_1$ and $\mathbf{B}_2$ span the same lattice, then there exists an $n \times n$ unimodular matrix $\mathbf{U} \in \mathrm{GL}_n(\mathbb{Z})$ with $\mathbf{B}_1 = \mathbf{B}_2\mathbf{U}$. The *volume* of L is defined as $\mathrm{vol}(L) = |\det(\mathbf{B})|$, independent of the choice of bases.

The *Gram–Schmidt orthogonalization* for an (ordered) basis $\mathbf{B}$ is the orthogonal family $\mathbf{B}^* = (\mathbf{b}_1^*, \ldots, \mathbf{b}_n^*) \in \mathbb{R}^{n \times n}$, recursively defined by $\mathbf{b}_1^* = \mathbf{b}_1$ and for $2 \le i \le n$

$$\mathbf{b}_i^* = \mathbf{b}_i - \sum_{j=1}^{i-1} \mu_{i,j} \mathbf{b}_j^*, \text{ where } \mu_{i,j} = \frac{\langle \mathbf{b}_i, \mathbf{b}_j^* \rangle}{\|\mathbf{b}_j^*\|^2} \text{ for } 1 \le j < i \le n.$$

Notice that the Gram–Schmidt vectors $\mathbf{b}_i^*$'s depend on the order of basis vectors in $\mathbf{B}$. For convenience, set $\mu = (\mu_{i,j}) \in \mathbb{R}^{n \times n}$ where let $\mu_{i,j} = 0$ for all $i < j$ and $\mu_{k,k} = 1$ for all k. Then $\mathbf{B} = \mathbf{B}^*\mu$, and thus $\mathrm{vol}(L) = \prod_{i=1}^{n} \|\mathbf{b}_i^*\|$ from the orthogonality of Gram–Schmidt vectors. For $2 \le \ell \le n$, let π_ℓ denote the orthogonal projection over the orthogonal supplement of the $\mathbb{R}$-vector space $\langle \mathbf{b}_1, \ldots, \mathbf{b}_{\ell-1} \rangle_{\mathbb{R}}$ as

$$\pi_\ell : \mathbb{R}^n \longrightarrow \langle \mathbf{b}_1, \ldots, \mathbf{b}_{\ell-1} \rangle_{\mathbb{R}}^{\perp} = \langle \mathbf{b}_\ell^*, \ldots, \mathbf{b}_n^* \rangle_{\mathbb{R}}, \ \pi_\ell(\mathbf{x}) = \sum_{i=\ell}^{n} \frac{\langle \mathbf{x}, \mathbf{b}_i^* \rangle}{\|\mathbf{b}_i^*\|^2} \mathbf{b}_i^*.$$

Every projection map depends on a basis. We also set $\pi_1 = \mathrm{id}$ for convenience.

2.2 Successive Minima, Hermite's Constants, and Gaussian Heuristic

For every $1 \le i \le n$, the ith *successive minimum* of an n-dimensional lattice L, denoted by $\lambda_i(L)$, is defined as the minimum of $\max_{1 \le j \le i} \|\mathbf{v}_j\|$ over all i linearly independent vectors $\mathbf{v}_1, \ldots, \mathbf{v}_i \in L$. In particular, the first minimum $\lambda_1(L)$ is the norm of the shortest non-zero vector in L. We clearly have $\lambda_1(L) \le \lambda_2(L) \le \cdots \le \lambda_n(L)$ by definition. Moreover, for any basis $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ of L, its Gram–Schmidt vectors satisfy $\lambda_i(L) \ge \min_{i \le j \le n} \|\mathbf{b}_j^*\|$ for every $1 \le i \le n$. (See Bremner 2011, Proposition 3.14 for proof.)

Hermite (1850) first proved that the quantity $\frac{\lambda_1(L)^2}{\mathrm{vol}(L)^{2/n}}$ is upper bounded over all lattices L of dimension n. Its supremum over all lattices of dimension n is called *Hermite's constant* of dimension n, denoted by γ_n. This implies $\lambda_1(L) \le \sqrt{\gamma_n}\mathrm{vol}(L)^{1/n}$ for any lattice L of dimension n. As its extension, it satisfies

$$\left(\prod_{i=1}^{r} \lambda_i(L)\right)^{1/r} \le \sqrt{\gamma_n}\mathrm{vol}(L)^{1/n} \text{ for } 1 \le r \le n.$$

This is known as Minkowski's second theorem. (See Martinet 2013, Chap. 2 for proof.) It is important to know the value of γ_n in order to obtain an upper bound of $\lambda_1(L)$; Minkowski's convex body theorem implies $\gamma_n \le 4\nu_n^{-2/n}$. (See Martinet 2013, Chap. 2 for proof.) This shows that

$$\lambda_1(L) \le 2\nu_n^{-1/n}\mathrm{vol}(L)^{1/n} \tag{1}$$

for any lattice L of dimension n. Moreover, it satisfies $\gamma_n \le 1 + \frac{n}{4}$ from well-known formulas for ν_n. It is very difficult to find the exact value of γ_n, and such values are known for only a few integers n. However, every γ_n is known as essentially linear in n. It also satisfies Mordell's inequality $\gamma_n \le \gamma_k^{(n-1)/(k-1)}$ for any $n \ge k \ge 2$. (See Nguyen 2009 for more details on Hermite's constants.)

Given a lattice L of dimension n and a measurable set S in $\mathbb{R}^n$, the *Gaussian Heuristic* predicts that the number of vectors in $L \cap S$ is roughly equal to $\mathrm{vol}(S)/\mathrm{vol}(L)$. By applying the ball of radius $\lambda_1(L)$ centered at the origin in $\mathbb{R}^n$, it leads to the prediction of the norm of the shortest non-zero vector in L. Specifically, the expectation of $\lambda_1(L)$ according to the Gaussian Heuristic is given by

$$\mathrm{GH}(L) = \nu_n^{-1/n}\mathrm{vol}(L)^{1/n} \sim \sqrt{\frac{n}{2\pi e}}\mathrm{vol}(L)^{1/n}.$$

This is tight compared to Eq. (1). Note that this is only a heuristic. But for "random" lattices, $\lambda_1(L)$ is asymptotically equal to $\mathrm{GH}(L)$ with overwhelming probability Ajtai (1996).

2.3 *Introduction to Lattice Problems*

The most famous lattice problem is given below.

? The Shortest Vector Problem (SVP)

Given a basis $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ of a lattice L, find the shortest non-zero vector in L, that is, a vector $\mathbf{s} \in L$ such that $\|\mathbf{s}\| = \lambda_1(L)$.

It was proven by Ajtai (1996) that SVP is NP-hard under randomized reductions. SVP can be relaxed by an approximate factor: *Given a basis of a lattice L and an approximation factor $f \geq 1$, find a non-zero vector* $\mathbf{v}$ *in L such that* $\|\mathbf{v}\| \leq f\lambda_1(L)$. Approximate-SVP is exactly SVP when $f = 1$. It is unlikely that one can efficiently solve approximate-SVP within quasi-polynomial factors in n, while approximate-SVP within a factor $\sqrt{n/\log(n)}$ is unlikely to be NP-hard. (See Nguyen 2009 for more details.)

Another famous lattice problem is given below.

? The Closest Vector Problem (CVP)

Given a basis $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ of a lattice L and a target vector $\mathbf{t}$, find a vector in L closest to $\mathbf{t}$, that is, a vector $\mathbf{v} \in L$ such that the distance $\|\mathbf{t} - \mathbf{v}\|$ is minimized.

CVP is at least as hard as SVP. As in the case of SVP, we can define an approximate variant of CVP by an approximate factor. Approximate-CVP is also at least as hard as approximate-SVP with the same factor. From a practical point of view, both are considered equally hard, due to Kannan's embedding technique Kannan (1987) which can transform approximate-CVP into approximate-SVP. (See also Galbraith 2012 for the embedding.)

The security of modern lattice-based cryptosystems is based on the hardness of cryptographic lattice problems, such as the LWE and the NTRU problems. (For example, see NIST 2016 for NIST post-quantum candidates.) Such lattice problems are reduced to approximate-SVP or approximate-CVP. (For example, see Albrecht et al. 2018 for details.)

3 Solving SVP Algorithms

In this section, we present typical algorithms for solving SVP. These algorithms can be classified into two categories, depending on whether they solve SVP *exactly* or *approximately*. However, both categories are *complementary*; exact algorithms first apply an approximation algorithm as a preprocessing to reduce their cost, while blockwise algorithms (e.g., the BKZ algorithm presented below) call many times an exact algorithm in low dimension as a subroutine to find a very short lattice vector.

3.1 Exact-SVP Algorithms

Exact-SVP algorithms find the non-zero shortest lattice vector, but they are expensive. These algorithms perform an exhaustive search of all short vectors, whose number is exponential in the dimension (in the worst case). These algorithms can be split in two categories; polynomial-space algorithms and exponential-space algorithms.

3.1.1 Polynomial-Space Exact Algorithms: Enumeration

They are based on *enumeration*, which dates back to the early 1980s with work by Pohst (1981), Kannan (1983), and Fincke–Pohst (1985). Enumeration is simply an exhaustive search for an integer combination of the basis vectors such that the lattice vector is the shortest. An enumeration algorithm takes as input an enumeration radius $R > 0$ and a basis $\mathbf{B} = (\mathbf{b}_1, \dots, \mathbf{b}_n)$ of a lattice L, and outputs all non-zero vectors $\mathbf{s}$ in L such that $\|\mathbf{s}\| \le R$ (if exists). The radius R is taken as an upper bound of $\lambda_1(L)$, like $\sqrt{\gamma_n}\mathrm{vol}(L)^{1/n}$, to find the shortest non-zero lattice vector. It goes through the enumeration tree formed by all vectors in the projected lattices $\pi_n(L)$, $\pi_{n-1}(L), \cdots, \pi_1(L) = L$ with norm at most R. More precisely, the enumeration tree is a tree of depth n, and for each $1 \le k \le n+1$, the nodes at depth $n+1-k$ are all the vectors in the projected lattice $\pi_k(L)$ with norm at most R. In particular, the root of the tree is the zero vector because $\pi_{n+1}(L) = \{\mathbf{0}\}$. The parent of a node $\mathbf{u} \in \pi_k(L)$ at depth $n+1-k$ is the node $\pi_{k+1}(\mathbf{u})$ at depth $n-k$. The child nodes are arranged in order of norms.

Here we introduce the basic idea of the Schnorr–Euchner algorithm Schnorr and Euchner (1994), which is a depth first search of the enumeration tree to find all leaves in practice. (cf. Kannan's algorithm 1983 is asymptotically superior in the running time, but it is not competitive in practice due to a substantial overhead of recursive procedures. See also Micciancio and Walter 2014 for such discussion.) We represent the shortest non-zero vector as $\mathbf{s} = v_1\mathbf{b}_1 + \cdots + v_n\mathbf{b}_n \in L$ for some unknown integers v_i's. With Gram–Schmidt information of $\mathbf{B}$, it is rewritten as

$$\mathbf{s} = \sum_{i=1}^{n} v_i \left(\mathbf{b}_i^* + \sum_{j=1}^{i-1} \mu_{i,j}\mathbf{b}_j^* \right) = \sum_{j=1}^{n} \left(v_j + \sum_{i=j+1}^{n} \mu_{i,j} v_i \right) \mathbf{b}_j^*.$$

Due to the orthogonality of Gram–Schmidt vectors $\mathbf{b}_j^*$'s, the squared norms of projections of the vector $\mathbf{s}$ are given as for every $1 \le k \le n$

$$\|\pi_k(\mathbf{s})\|^2 = \sum_{j=k}^{n} \left(v_j + \sum_{i=j+1}^{n} \mu_{i,j} v_i \right)^2 \|\mathbf{b}_j^*\|^2.$$

If $\mathbf{s}$ is a leaf of the enumeration tree, then its projections all satisfy $\|\pi_k(\mathbf{s})\|^2 \le R^2$ for all $1 \le k \le n$. These n inequalities together with above equations enable to perform an exhaustive search for the integral coordinates $v_n, v_{n-1}, \dots, v_1$ of $\mathbf{s}$:

$$\left(v_k + \sum_{i=k+1}^{n} \mu_{i,k} v_i \right)^2 \le \frac{R^2 - \sum_{j=k+1}^{n} \left(v_j + \sum_{i=j+1}^{n} \mu_{i,j} v_i \right)^2 \|\mathbf{b}_j^*\|^2}{\|\mathbf{b}_k^*\|^2} \qquad (2)$$

for every $1 \leq k \leq n$. We start with $k = n$ in Eq. (2), that is, $0 \leq v_n \leq \frac{R}{\|\mathbf{b}_n^*\|}$, because we can restrict to "positive" nodes due to the symmetry of the enumeration tree. Choosing a candidate of v_n, we move to the next index $k = n - 1$ in Eq. (2), that is, $(v_{n-1} + \mu_{n,n-1}v_n)^2 \leq \frac{R^2 - v_n^2\|\mathbf{b}_n^*\|^2}{\|\mathbf{b}_{n-1}^*\|^2}$ to find a candidate of v_{n-1}. By repeating this procedure, assume that the integers $v_n, \ldots, v_{k+1}$ are found for some $1 < k < n$. Then Eq. (2) enables to compute an interval I_k such that $v_k \in I_k$, and thus to perform an exhaustive search for the integer v_k. A depth first search of the tree corresponds to enumerating the interval from its middle, namely, a zig-zag search like

$$v_k = \lfloor c_k \rceil, \quad \lfloor c_k \rceil \pm 1, \quad \lfloor c_k \rceil \pm 2, \quad \cdots,$$

where $c_k = -\sum_{i=k+1}^{n} \mu_{i,k} v_i$. The basic Schnorr–Euchner enumeration algorithm Schnorr and Euchner (1994) is as below (see Gama et al. 2010, Algorithm 2 for the algorithm with some improvements).

Algorithm: The basic Schnorr–Euchner enumeration Schnorr and Euchner (1994)

```
Input: A basis $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ of a lattice $L$ and a radius $R$ with $\lambda_1(L) \leq R$
Output: The shortest non-zero vector $\mathbf{s} = \sum_{i=1}^{n} v_i \mathbf{b}_i$ in $L$
 1: Compute Gram–Schmidt information $\mu_{i,j}$ and $\|\mathbf{b}_i^*\|^2$ of $\mathbf{B}$
 2: $(\rho_1, \ldots, \rho_{n+1}) = \mathbf{0}$, $(v_1, \ldots, v_n) = (1, 0, \ldots, 0)$, $(c_1, \ldots, c_n) = \mathbf{0}$, $(w_1, \ldots, w_n) = \mathbf{0}$
 3: $k = 1$, last_nonzero $= 1$  // largest $i$ for which $v_i \neq 0$
 4: while true do
 5:   $\rho_k \leftarrow \rho_{k+1} + (v_k - c_k)^2 \cdot \|\mathbf{b}_k^*\|^2$  // $\rho_k = \|\pi_k(\mathbf{s})\|^2$
 6:   if $\rho_k \leq R^2$ then
 7:     if $k = 1$ then $R^2 \leftarrow \rho_k$, $\mathbf{s} \leftarrow \sum_{i=1}^{n} v_i \mathbf{b}_i$;  // update the squared radius
 8:     else $k \leftarrow k - 1$, $c_k \leftarrow -\sum_{i=k+1}^{n} \mu_{i,k} v_i$, $v_k \leftarrow \lfloor c_k \rceil$, $w_k \leftarrow 1$;
 9:   else
10:     $k \leftarrow k + 1$  // going up the tree
11:     if $k = n + 1$ then return $\mathbf{s}$;
12:     if $k \geq$ last_nonzero then last_nonzero $\leftarrow k$, $v_k \leftarrow v_k + 1$;
13:     else
14:       if $v_k > c_k$ then $v_k \leftarrow v_k - w_k$; else $v_k \leftarrow v_k + w_k$;  // zig-zag search
15:       $w_k \leftarrow w_k + 1$
16:     end if
17:   end if
18: end while
```

The running time of the enumeration algorithm fully depends on the total number of tree nodes N. An estimate of N can be derived from the Gaussian Heuristic. More precisely, the number of nodes at level ℓ is exactly half the number of vectors in the projected lattice $\pi_{n+1-\ell}(L)$ with norm at most R. Since $\mathrm{vol}(\pi_{n+1-\ell}(L)) =$

$\prod_{i=n+1-\ell}^{n} \|\mathbf{b}_i^*\|$, the Gaussian Heuristic predicts the number of nodes at level ℓ scanned by the Schnorr–Euchner algorithm to be close to

$$H_\ell \approx \frac{1}{2} \cdot \frac{V_\ell(R)}{\prod_{i=n+1-\ell}^{n} \|\mathbf{b}_i^*\|}.$$

Then $N \approx \sum_{\ell=1}^{n} H_\ell$. For a "good" basis (reduced by LLL or BKZ, introduced in the next subsection), we have $\|\mathbf{b}_i^*\|/\|\mathbf{b}_{i+1}^*\| \approx q$ for some constant q. This is called the *geometric series assumption (GSA)*,[1] first introduced by Schnorr (2003). The constant q depends on the reduction algorithm. For example, we experimentally have $q \approx 1.04$ by LLL and $q \approx 1.025$ by BKZ with blocksize 20 for high-dimensional lattices (see Gama and Nguyen 2008 for details.) Now we take the enumeration radius $R = \sqrt{\gamma_n}\mathrm{vol}(L)^{1/n}$, which is optimal in the worst case. With the constant q, we estimate

$$H_\ell \approx \frac{q^{(n-\ell)(n-1)/2} V_\ell(\sqrt{\gamma_n})}{2q^{(n-\ell-1)(n-\ell)/2}} = q^{\ell(n-\ell)/2} 2^{O(n)}$$

since we can roughly estimate $V_\ell(\sqrt{\gamma_n}) = 2^{O(n)}$ from $\sqrt{\gamma_n} = \Theta\left(\sqrt{n}\right)$ Gama et al. (2010). The right-hand term is maximized for $\ell = \frac{n}{2}$, and it is less than $q^{n^2/8} 2^{O(n)}$. Thus the maximum of H_ℓ is super-exponential in n and is reached for $\ell \approx \frac{n}{2}$. (See Gama et al. 2010, Fig. 1 for the actual number of nodes, which is very close to this prediction.) Since smaller q is obtained for a more reduced basis, it shows that the more reduced the input basis is, the less are the nodes in the enumeration tree, and the cheaper the enumeration cost.

It is possible to obtain substantial speedups using *pruning* techniques by Gama et al. (2010). Their idea is tempting not to enumerate all the tree nodes, by discarding certain branches. (See Aono et al. 2018 for a lower bound of the time complexity of pruned enumeration.) However, it decreases the success probability to find the shortest non-zero lattice vector $\mathbf{s}$. For instance, one might intuitively hope that $\|\pi_{n/2}(\mathbf{s})\|^2 \lessapprox \|\mathbf{s}\|^2/2$, which is more restrictive than the inequality $\|\pi_{n/2}(\mathbf{s})\|^2 \leq \|\mathbf{s}\|^2$. Formally, pruning replaces each of the n inequalities $\|\pi_k(\mathbf{s})\|^2 \leq R^2$ by $\|\pi_k(\mathbf{s})\|^2 \leq R_{n+1-k}^2$, where $R_1 \leq \cdots \leq R_n = R$ are n real numbers defined by a pruning strategy. A pruning parameter is set in the fplll library The FPLLL development team (2016), and a pruning function for setting R_i's is implemented in the progressive BKZ library Aono et al. (2016).

3.1.2 Exponential-Space Exact Algorithms: Sieve

These algorithms have a better asymptotic running time, but they all require exponential space $2^{\Theta(n)}$. The first algorithm of this kind is the randomized sieve algorithm proposed by Ajtai, Kumar, and Sivakumar (AKS) Ajtai et al. (2001). The AKS

[1]This assumption states that for a reduced basis $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$, the plots of its Gram–Schmidt log-norms $\log \|\mathbf{b}_i^*\|$ for $1 \leq i \leq n$ are on a straight line. (For example, see Schnorr 2003, Fig. 1.)

algorithm outputs the shortest lattice vector with overwhelming probability, and its asymptotic complexity is much better than deterministic enumeration algorithms with $2^{O(n^2)}$ time complexity. The main idea is as follows (see also Nguyen 2008, Sect. 3 or Nguyen 2009): Given a lattice L of dimension n, consider a ball S centered at the origin and of radius r with $\lambda_1(L) \leq r \leq O(\lambda_1(L))$. Then $\#(L \cap S) = 2^{O(n)}$ based on the Gaussian Heuristic. If we could perform an exhaustive search for all vectors in $L \cap S$, we could find the shortest lattice vector within $2^{O(n)}$ polynomial-time operations. Enumeration enables to perform an exhaustive search of $L \cap S$, but it requires to go through all the vectors in the union set $\widetilde{S} = \bigcup_{k=1}^{n} (\pi_k(L) \cap S)$, whose total number is much larger than $\#(L \cap S)$. In contrast, the AKS algorithm performs a randomized sampling of $L \cap S$, without going through the set $\widetilde{S}$. If it was uniformly sampled over $L \cap S$, a short lattice vector would be included in N samples with probability close to 1 for $N \gg \#(L \cap S)$. Unfortunately, it is unclear whether the uniform property is satisfied by the AKS sampling. However, it can be shown that there exists a vector $\mathbf{w} \in L \cap S$ such that $\mathbf{w}$ and $\mathbf{w} + \mathbf{s}$ can be sampled with non-zero probability for some shortest lattice vector $\mathbf{s}$. Thus the shortest lattice vector is obtained by computing the shortest difference of any pairs of the N sampled vectors in $L \cap S$.

There are several heuristic variants of the AKS algorithm with time complexity $2^{O(n)}$ and space complexity exponential in n for an n-dimensional lattice L Baiet al. (2016), Herold and Kirshanova (2017), Micciancio and Voulgaris (2010), Nguyen (2008). Given a basis of L, these algorithms build databases of lattice vectors with norms at most $R \cdot \mathrm{GH}(L)$ for a small constant $R > 0$ such as $R^2 = \frac{4}{3}$. In generic sieves, it is checked whether the sum or the difference of any pair of vectors in databases becomes shorter. The basic sieve algorithm is as below.

Algorithm: The basic sieve

Input: A basis $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ of a lattice L and a size parameter $N = \left(\frac{4}{3}\right)^{n/2+o(n)}$
Output: A database of N short vectors in L

```
1: Take a set $D$ of $N$ random vectors in $L$ (with norm at most $2^n \mathrm{vol}(L)^{1/n}$)
2: while $\exists(\mathbf{v}, \mathbf{w}) \in D^2$ such that $\|\mathbf{v} + \mathbf{w}\| < \|\mathbf{v}\|$ (resp., $\|\mathbf{v} - \mathbf{w}\| < \|\mathbf{v}\|$) do
3:     $\mathbf{v} \leftarrow \mathbf{v} + \mathbf{w}$ (resp., $\mathbf{v} \leftarrow \mathbf{v} - \mathbf{w}$) // update vectors in the database $D$
4: end while
5: return $D$
```

In Step 1 of the above algorithm, the initialization of the database D can be performed by first computing an LLL-reduced basis (see the next subsection for the LLL reduction), and taking random small integral combinations of the basis vectors. (A natural idea is to use a stronger reduction algorithm such as BKZ in order to generate shorter initial vectors.) The Nguyen–Vidick sieve (2008) finds pairs of vectors $(\mathbf{v}_1, \mathbf{v}_2)$ from D, whose sum or difference gives a shorter vector, that is, $\|\mathbf{v}_1 \pm \mathbf{v}_2\| < \max_{\mathbf{v} \in D} \|\mathbf{v}\|$. Once such a pair is found, the longest vector from the database gets replaced by $\mathbf{v}_1 \pm \mathbf{v}_2$. The database size is a priori fixed to the asymptotic heuristic minimum $2^{0.2075n+O(n)}$ in order to find enough such pairs. The running time is quadratic in the database size. The Gauss sieve (2010) is a variant of the Nguyen–Vidick sieve with substantial improvements; the main improvement is to divide the database into two parts, the so-called "list " part and the "queue" part. Both parts are separately sorted by Euclidean norm in order to make early reduction likely. In updating vectors, the queue part enables to avoid considering the same pair several times. The running time and the database size for the Gauss sieve are asymptotically the same as for the Nguyen–Vidick sieve, but its performance is better in practice. The 3-sieve Baiet al. (2016), Herold and Kirshanova (2017) searches for triples of lattice vectors whose sum gives a shorter vector. (cf. the Nguyen–Vidick and the Gauss algorithms are a kind of 2-sieve.) There are more possible triples than pairs to shorten vectors in the database, but a search for such triples is more costly. (Filtering techniques Herold and Kirshanova 2017 are required to speed up such a search.) Several tricks and techniques have been proposed to improve sieve algorithms, such as the SimHash technique Charikar (2002), Ducas (2018), Fitzpatrick et al. (2014). Several practical sieve algorithms also have been implemented in the fplll library The FPLLL development team (2016).

3.2 Approximate-SVP Algorithms

These algorithms are much faster than exact algorithms, but they output short lattice vectors, not necessarily the shortest ones.

3.2.1 LLL Reduction

The first efficient approximate-SVP algorithm is the celebrated algorithm by Lenstra, Lenstra, and Lovász (LLL) Lenstra et al. (1982). Nowadays it is known as the most famous algorithm of *lattice basis reduction*, which finds a lattice basis with short and nearly orthogonal basis vectors. Such a basis is called *reduced* or *good*. We introduce the notion of LLL reduction. Let $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ be a basis of a lattice L, and $\mathbf{B}^* = (\mathbf{b}_1^*, \ldots, \mathbf{b}_n^*)$ its Gram–Schmidt vectors with coefficients $\mu_{i,j}$. For a parameter $\frac{1}{4} < \delta < 1$, the basis $\mathbf{B}$ is called δ*-LLL-reduced* if it satisfies two conditions: (i) (Size-reduction condition) $|\mu_{i,j}| \leq \frac{1}{2}$ for all $1 \leq j < i \leq n$. (ii) (Lovász' condition) $\delta \|\mathbf{b}_{k-1}^*\|^2 \leq \|\pi_{k-1}(\mathbf{b}_k)\|^2$ for all $2 \leq k \leq n$. This can be rewritten as $\|\mathbf{b}_k^*\|^2 \geq (\delta - \mu_{k,k-1}^2)\|\mathbf{b}_{k-1}^*\|^2$. Any δ-LLL-reduced basis satisfies the below properties (see Bremner 2011 for proof):

- $\|\mathbf{b}_1\| \leq \alpha^{(n-1)/4}\mathrm{vol}(L)^{1/n}$, where $\alpha = \frac{4}{4\delta-1} > \frac{4}{3}$.
- $\|\mathbf{b}_i\| \leq \alpha^{(n-1)/2}\lambda_i(L)$ for $1 \leq i \leq n$, and $\prod_{i=1}^n \|\mathbf{b}_i\| \leq \alpha^{n(n-1)/4}\mathrm{vol}(L)$.

Given any basis of L, the LLL algorithm finds a δ-LLL-reduced basis of L. As seen from the above second property, it can solve approximate-SVP with factor $\alpha^{(n-1)/2}$. The basic LLL algorithm is given below (see also Galbraith 2012, Chap. 17 or Nguyen 2009).

Algorithm: The basic LLL Lenstra et al. (1982)

```
Input: A basis $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ of a lattice $L$, and a reduction parameter $\frac{1}{4} < \delta < 1$
Output: A $\delta$-LLL-reduced basis $\mathbf{B}$ of $L$
1: Compute Gram–Schmidt information $\mu_{i,j}$ and $\|\mathbf{b}_i^*\|^2$ of the input basis $\mathbf{B}$
2: $k \leftarrow 2$
3: while $k \leq n$ do
4:   Size-reduce $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ // At each $k$, we recursively change $\mathbf{b}_k \leftarrow \mathbf{b}_k - \lfloor \mu_{k,j} \rceil \mathbf{b}_j$ for $1 \leq j \leq k-1$ (e.g., see Galbraith 2012, Algorithm 24)
5:   if $(\mathbf{b}_{k-1}, \mathbf{b}_k)$ satisfies Lovász' condition then
6:     $k \leftarrow k+1$
7:   else
8:     Swap $\mathbf{b}_k$ with $\mathbf{b}_{k-1}$, and update Gram–Schmidt information of $\mathbf{B}$
9:     $k \leftarrow \max(k-1, 2)$
10:  end if
11: end while
```

In the LLL algorithm, a pair of adjacent basis vectors $(\mathbf{b}_{k-1}, \mathbf{b}_k)$ is swapped if it does not satisfy Lovász' condition. Thus the output basis is δ-LLL-reduced if the algorithm terminates. The quantity $\mathrm{Pot}(\mathbf{B}) = \prod_{i=1}^{n-1} \|\mathbf{b}_i^*\|^{2(n-i)}$ is called the *potential* of a basis $\mathbf{B}$. Every swap in the LLL algorithm decreases the potential of an input basis by a factor at least $\delta < 1$. (cf. the size-reduction procedure does not change the potential.) This guarantees the termination of the LLL algorithm in polynomial time in n. Furthermore, the LLL algorithm is applicable also for linearly dependent vectors to remove their linear dependency. (See Bremner 2011, Chap. 6, Cohen 2013, Sect. 2.6.4, Pohst 1987 or Sims 1994, Sect. 8.7 for details.)

3.2.2 Variants of LLL

LLL with Deep Insertions (DeepLLL)

This variant is a straightforward generalization of LLL, in which *non-adjacent* basis vectors can be changed. Specifically, a basis vector $\mathbf{b}_k$ is inserted between $\mathbf{b}_{i-1}$ and $\mathbf{b}_i$ as $\sigma_{i,k}(\mathbf{B}) = (\ldots, \mathbf{b}_{i-1}, \mathbf{b}_k, \mathbf{b}_i, \ldots, \mathbf{b}_{k-1}, \mathbf{b}_{k+1}, \ldots)$, called a *deep insertion*, if the so-called deep exchange condition $\|\pi_i(\mathbf{b}_k)\|^2 < \delta \|\mathbf{b}_i^*\|^2$ is satisfied for $\frac{1}{4} < \delta < 1$. In this case, the new GSO vector at the ith position is given by $\pi_i(\mathbf{b}_k)$, strictly shorter than the old GSO vector $\mathbf{b}_i^*$. A basis $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ is called *δ-DeepLLL-reduced* if it satisfies two conditions: (i) it is size-reduced, (ii) $\|\pi_i(\mathbf{b}_k)\|^2 \geq \delta \|\mathbf{b}_i^*\|^2$ for all $1 \leq i < k \leq n$. (The case $i = k-1$ is just Lovász' condition.) Any δ-DeepLLL-reduced basis satisfies the below properties Yasuda and Yamaguchi (2019), Theorem 1:

- $\|\mathbf{b}_1\| \leq \alpha^{\frac{n-1}{2n}} \left(1 + \frac{\alpha}{4}\right)^{\frac{(n-1)(n-2)}{4n}} \mathrm{vol}(L)^{\frac{1}{n}}$, where α is the same as in LLL.
- $\|\mathbf{b}_i\| \leq \sqrt{\alpha} \left(1+\frac{\alpha}{4}\right)^{\frac{n-2}{2}} \lambda_i(L)$ for $1 \leq i \leq n$, and $\prod_{i=1}^n \|\mathbf{b}_i\| \leq \left(1 + \frac{\alpha}{4}\right)^{\frac{n(n-1)}{4}} \mathrm{vol}(L)$.

These properties are strictly stronger than the case of LLL. The basic DeepLLL algorithm Schnorr and Euchner (1994) is given below (see also Bremner 2011, Fig. 5.1 or Cohen 2013, Algorithm 2.6.4).

Algorithm: The basic DeepLLL Schnorr and Euchner (1994)

Input: A basis $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ of a lattice L, and a reduction parameter $\frac{1}{4} < \delta < 1$
Output: A δ-DeepLLL-reduced basis $\mathbf{B}$ of L

```
1: Compute Gram–Schmidt information $\mu_{i,j}$ and $\|\mathbf{b}_i^*\|^2$ of the input basis $\mathbf{B}$
2: $k \leftarrow 2$
3: while $k \le n$ do
4:   Size-reduce $\mathbf{B}$ as in LLL
5:   $C \leftarrow \|\mathbf{b}_k\|^2$, $i \leftarrow 1$
6:   while $i < k$ do
7:     if $C \ge \delta \|\mathbf{b}_i^*\|^2$ then
8:       Compute $C \leftarrow C - \mu_{k,i}^2 \|\mathbf{b}_i^*\|^2$ and $i \leftarrow i + 1$ // $C = \|\pi_i(\mathbf{b}_k)\|^2$
9:     else
10:      $\mathbf{B} \leftarrow \sigma_{i,k}(\mathbf{B})$ // a deep insertion
11:      Update the Gram–Schmidt information of $\mathbf{B}$, and $k \leftarrow \max(i, 2) - 1$
12:    end if
13:  end while
14:  $k \leftarrow k + 1$
15: end while
```

Compared with LLL, it is complicated to update the Gram–Schmidt information of $\mathbf{B}$ after every deep insertion. (See Yamaguchi and Yasuda 2017.) Every deep insertion does not always decrease the potential of an input basis, and thus the complexity of DeepLLL is no longer polynomial-time but potentially super-exponential in the lattice dimension. However, DeepLLL often finds much shorter lattice vectors than LLL in practice Gama and Nguyen (2008).

Block Korkine–Zolotarev (BKZ) Algorithm

Let us first introduce a strong notion of reduction: A basis $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ of a lattice L is called *HKZ-reduced* if it is size-reduced and it satisfies $\|\mathbf{b}_i^*\| = \lambda_1(\pi_i(L))$ for all $1 \le i \le n$. For $1 \le i \le j \le n$, denote by $\mathbf{B}_{[i,j]}$ the local projected block $(\pi_i(\mathbf{b}_i), \pi_i(\mathbf{b}_{i+1}), \ldots, \pi_i(\mathbf{b}_j))$, and by $L_{[i,j]}$ the lattice spanned by $\mathbf{B}_{[i,j]}$. The notion of BKZ-reduction is a local block version of HKZ-reduction Schnorr (1987), Schnorr (1992), Schnorr and Euchner (1994). For a blocksize $2 \le \beta \le n$, a basis $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ of a lattice L is called *β-BKZ-reduced* if it is size-reduced and every local block $\mathbf{B}_{[j,j+\beta-1]}$ is HKZ-reduced for $1 \le j \le n - \beta + 1$. The second condition means $\|\mathbf{b}_j^*\| = \lambda_1(L_{[j,k]})$ for $1 \le j \le n - 1$ with $k = \min(j + \beta - 1, n)$. Every β-BKZ-reduced basis satisfies $\|\mathbf{b}_1\| \le \gamma_\beta^{(n-1)/(\beta-1)} \lambda_1(L)$ Schnorr (1992). The

BKZ algorithm Schnorr and Euchner (1994) finds a β-BKZ-reduced basis, and it calls LLL to reduce every local block before finding the shortest vector over the block lattice. (As β increases, a shorter lattice vector can be found, but the running time is more costly.)

Algorithm: The basic BKZ Schnorr and Euchner (1994)

```
Input: A basis $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ of a lattice $L$, a blocksize $2 \leq \beta \leq n$, and a reduction parameter $\frac{1}{4} < \delta < 1$ of LLL
Output: A $\beta$-DeepBKZ-reduced basis $\mathbf{B}$ of $L$
1: $\mathbf{B} \leftarrow \mathrm{LLL}(\mathbf{B}, \delta)$ // Compute $\mu_{i,j}$ and $\|\mathbf{b}_j^*\|^2$ of the new basis $\mathbf{B}$ together
2: $z \leftarrow 0$, $j \leftarrow 0$
3: while $z < n - 1$ do
4:   $j \leftarrow (j \bmod (n-1)) + 1$, $k \leftarrow \min(j + \beta - 1, n)$, $h \leftarrow \min(k + 1, n)$
5:   Find $\mathbf{v} \in L$ such that $\|\pi_j(\mathbf{v})\| = \lambda_1(L_{[j,k]})$ by enumeration or sieve
6:   if $\|\pi_j(\mathbf{v})\|^2 < \|\mathbf{b}_j^*\|^2$ then
7:     $z \leftarrow 0$ and call $\mathrm{LLL}((\mathbf{b}_1, \ldots, \mathbf{b}_{j-1}, \mathbf{v}, \mathbf{b}_j, \ldots, \mathbf{b}_h), \delta)$ // Insert $\mathbf{v} \in L$ and remove the linear dependency to obtain a new basis
8:   else
9:     $z \leftarrow z + 1$ and call $\mathrm{LLL}((\mathbf{b}_1, \ldots, \mathbf{b}_h), \delta)$
10:  end if
11: end while
```

It is customary to terminate the BKZ algorithm after a selected number of calls to an exact-SVP algorithm over block lattices. (See Hanrot et al. 2011 for analysis.) Efficient variants such as BKZ 2.0 Chen (2011) have been proposed, and some of them have been implemented in The FPLLL development team (2016). The *Hermite factor* is a good index to measure the practical output quality of a reduction algorithm. (See Gama and Nguyen 2008 for their experiments.) It is defined by $\gamma = \frac{\|\mathbf{v}\|}{\mathrm{vol}(L)^{1/n}}$, where $\mathbf{v}$ is the shortest basis vector output by a reduction algorithm for a basis of a lattice L of dimension n. Under the Gaussian Heuristic and GSA, a limiting value of the root Hermite factor of BKZ with blocksize β is predicted in Chen (2013) as

$$\lim_{n\to\infty} \gamma^{\frac{1}{n}} = \left(\nu_\beta^{-\frac{1}{\beta}}\right)^{\frac{1}{\beta-1}} \sim \left(\frac{\beta}{2\pi e}(\pi\beta)^{\frac{1}{\beta}}\right)^{\frac{1}{2(\beta-1)}}.$$

There are experimental evidences to support this prediction for high blocksizes such as $\beta > 50$. (Note that the Gaussian Heuristic holds in practice for random lattices in high dimensions, but unfortunately it is violated in low dimensions.) In a simple form based on the Gaussian Heuristic, the GSA shape of a β-BKZ-reduced basis of volume 1 is predicted as $\|\mathbf{b}_i^*\| \approx \alpha_\beta^{\frac{n-1}{2}-i}$, where $\alpha_\beta = \left(\frac{\beta}{2\pi e}\right)^{1/\beta}$. This is reasonably accurate in practice for $\beta > 50$ and $\beta \ll n$. (See Chen 2013, 2011; Yu and Ducas 2017.) Other variants of BKZ have been proposed such as slide reduction Gama and Nguyen (2008), self-dual BKZ Micciancio and Walter (2016), and progressive-

BKZ Aono et al. (2016). As a mathematical improvement of BKZ, DeepBKZ was recently proposed in Yamaguchi and Yasuda (2017), in which DeepLLL is called a subroutine alternative to LLL. In particular, DeepBKZ finds a short lattice vector by smaller blocksizes than BKZ in practice. (Dual and self-dual variants of DeepBKZ were also proposed in Yasuda (2018), Yasuda et al. (2018).)

4 The SVP Challenge and Recent Strategies

To test algorithms solving SVP, sample lattice bases are presented in Darmstadt (2010) for dimensions from 40 up to 200. (The lattices are random in the sense of Goldstein and Mayer Goldstein and Mayer (2003).) For every lattice L, any non-zero lattice vector with (Euclidean) norm less than $1.05\mathrm{GH}(L)$ can be submitted to the hall of fame in the SVP challenge. To enter the hall of fame, the lattice vector is required to be shorter than a previous one in the same dimension (with possibly different seed). Note that not all lattice vectors in the hall of fame are necessarily the shortest. In this section, we introduce two recent strategies for solving the SVP challenge in high dimensions such as $n \geq 150$.

4.1 The Random Sampling Strategy

Early in 2017, a non-zero vector in a lattice L of dimension $n = 150$ with norm less than $1.05\mathrm{GH}(L)$ was first found by Teruya and Kashiwabara using many high-performance servers. (See Teruya et al. 2018 for their large-scale experiments.) Their strategy is based on the work of Fukase and Kashiwabara (2015), which is an extension of Schnorr's random sampling reduction (RSR) Schnorr (2003). Here we review random sampling (SA) and RSR. For a lattice L of dimension n, fix $1 \leq u < n$ to be a constant of search space bound. Given a basis $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ of L, SA samples a vector $\mathbf{v} = \sum_{i=1}^{n} \nu_i \mathbf{b}_i^*$ in L satisfying $\nu_i \in (-1/2, 1/2]$ for $1 \leq i < n - u$, $\nu_i \in (-1, 1]$ for $n - u \leq i < n$ and $\nu_n = 1$. Let $S_{u,\mathbf{B}}$ denote the set of such lattice vectors. Since the number of candidates for ν_i with $|\nu_i| \leq 1/2$ (resp. $|\nu_i| \leq 1$) is 1 (resp. 2), there are 2^u lattice vectors in $S_{u,\mathbf{B}}$. By calling SA up to 2^u times, RSR generates $\mathbf{v}$ satisfying $\|\mathbf{v}\|^2 < 0.99\|\mathbf{b}_1\|^2$ Schnorr (2003), Theorem 1. Two extensions are proposed in Fukase and Kashiwabara (2015) for solving the SVP challenge; the first one is to represent a lattice vector by a sequence of natural numbers via the Gram–Schmidt orthogonalization, and to sample lattice vectors on an appropriate distribution of the representation. The second one is to decrease the sum of the squared Gram–Schmidt lengths $\mathrm{SS}(\mathbf{B}) := \sum_{i=1}^{n} \|\mathbf{b}_i^*\|^2$ to make it easier to sample very short lattice vectors. The effectiveness of their extensions is guaranteed by their statistical analysis on lattices. Specifically, under the randomness assumption (RA),[2] they roughly estimate that the distribution of the squared length of a sampled vector $\|\mathbf{v}\|^2 = \sum_{i=1}^{n} \nu_i^2 \|\mathbf{b}_i^*\|^2$ follows the normal distribution $\mathcal{N}(\mu, \sigma^2)$ with

$$\mu = \frac{\sum_{i=1}^{n} \|\mathbf{b}_i^*\|^2}{12} \quad \text{and} \quad \sigma = \left(\frac{\sum_{i=1}^{n} \|\mathbf{b}_i^*\|^4}{180}\right)^{1/2}.$$

This implies that *shorter* lattice vectors are sampled as the squared-sum $\mathrm{SS}(\mathbf{B})$ becomes *smaller*. Then the basic strategy in Fukase and Kashiwabara (2015); Teruya et al. (2018) consists of the following two steps: (i) We reduce an input basis so that it decreases the sum of its squared Gram–Schmidt lengths as small as possible, by using LLL and insertion of sampled lattice vectors like BKZ. (See also Yasuda et al. 2017 for such procedure). (ii) With such reduced basis $\mathbf{B}$, we then find a short lattice vector by randomly sampling $\mathbf{v} = \sum_{i=1} \nu_i \mathbf{b}_i^*$.

As a sequential work, Aono and Nguyen (2017) introduced lattice enumeration with discrete pruning to generalize random sampling, and also provided a deep analysis of discrete pruning by using the volume of the intersection of a ball with a box. In particular, under RA, the expectation of the length of a short vector generated by lattice enumeration with discrete pruning from the so-called tag $\mathbf{t} = (t_1, \ldots, t_n) \in \mathbb{Z}^n$ is roughly given by $E(\mathbf{t}) = \sum_{i=1}^{n} \left(\frac{t_i^2}{4} + \frac{t_i}{4} + \frac{1}{12}\right) \|\mathbf{b}_i^*\|^2$, which is a generalization of the above mean μ. However, it is shown in Aono and Nguyen (2017) that the empirical correlation between $E(\mathbf{t})$ and the volume of ball-box intersection is negative. This is statistical evidence why decreasing $\mathrm{SS}(\mathbf{B})$ is important instead of increasing the volume of ball-box intersection. Furthermore, the calculation of the volume presented in Aono and Nguyen (2017) is much less efficient than the computation of $\mathrm{SS}(\mathbf{B})$. In 2018, Matsuda et al. (2018) investigated the strategy of Fukase and Kashiwabara (2015) by the Gram–Charlier approximation in order to precisely estimate the success probability of sampling short lattice vectors, and also discussed the effectiveness of decreasing $\mathrm{SS}(\mathbf{B})$ for sampling short lattice vectors.

4.2 *The Sub-Sieving Strategy*

Around the end of August 2018, many records for the SVP challenge in dimensions up to 155 had been found by the sub-sieving strategy of Ducas (2018). (See Albrecht et al. 2019 for their experiments report.) The basic idea is to reduce SVP in high dimensions to the *bounded distance decoding (BDD)* problem in low dimensions, a particular case of CVP, in which the target vector is known to be somewhat close to the lattice. It enforces us to find an enormous number of short vectors in projected

[2] RA states that the coefficients ν_i of $\mathbf{v} = \sum_{i=1}^{n} \nu_i \mathbf{b}_i^*$ sampled by SA are uniformly distributed in $[-1/2, 1/2]$ for $1 \le i < n - u$ and in $[-1, 1]$ for $n - u \le i < n$. It does not hold strictly in practice.

lattices, and the sieve is useful to collect such vectors. In particular, the sieve is performed in projected lattices instead of the full lattice.

The specific strategy is as follows Ducas (2018), Section 3. Given a basis $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ of a lattice L of high dimension n, we fix an integer d with $1 \leq d \leq n$, and perform the sieve in the projected lattice $\pi_d(L)$ to obtain a list of short lattice vectors

$$D := \left\{ \mathbf{v} \in \pi_d(L) \mid \mathbf{v} \neq \mathbf{0} \text{ and } \|\mathbf{v}\| \leq \sqrt{\frac{4}{3}} \mathrm{GH}\left(\pi_d(L)\right) \right\}.$$

We hope that the desired shortest non-zero vector $\mathbf{s}$ in the full lattice L projects to a vector in the above list D, that is, it satisfies $\pi_d(\mathbf{s}) \neq \mathbf{0}$ and $\|\pi_d(\mathbf{s})\| \leq \sqrt{\frac{4}{3}}\mathrm{GH}(\pi_d(L))$. (Note that $\pi_d(\mathbf{s}) = \mathbf{0}$ means that the vector $\mathbf{s}$ is in the sub-lattice $\mathcal{L}(\mathbf{b}_1, \ldots, \mathbf{b}_{d-1})$ of L. Here we do not care about the case.) Since $\|\pi_d(\mathbf{s})\| \leq \|\mathbf{s}\| \approx \mathrm{GH}(L)$, the condition

$$\mathrm{GH}(L) \leq \sqrt{\frac{4}{3}} \mathrm{GH}\left(\pi_d(L)\right) \tag{3}$$

is sufficient to satisfy our hope. This condition is not tight, since the projected vector $\pi_d(\mathbf{s})$ becomes shorter than the full vector $\mathbf{s}$ as the index d increases. By exhaustive search over the list D, we assume that the projected vector $\mathbf{s}_d := \pi_d(\mathbf{s}) \in D$ is known. We need to recover the full vector $\mathbf{s}$ from $\mathbf{s}_d$. Write $\mathbf{s} = \mathbf{B}\mathbf{x}$ for some $\mathbf{x} \in \mathbb{Z}^n$, and split $\mathbf{x}$ as $(\mathbf{x}_1 \mid \mathbf{x}_2)$ with $\mathbf{x}_1 \in \mathbb{Z}^{d-1}$ and $\mathbf{x}_2 \in \mathbb{Z}^{n-d+1}$. Then $\mathbf{s}_d = \pi_d(\mathbf{B}\mathbf{x}) = \mathbf{B}_d\mathbf{x}_2$ and hence $\mathbf{x}_2$ is known, where $\mathbf{B}_d = (\pi_d(\mathbf{b}_d), \ldots, \pi_d(\mathbf{b}_n))$. Now we need to recover $\mathbf{x}_1$ so that $\mathbf{s} = \mathbf{B}_1\mathbf{x}_1 + \mathbf{B}_2\mathbf{x}_2$ is small (or the shortest), where $\mathbf{B} = (\mathbf{B}_1 \mid \mathbf{B}_2)$. This is an easy BDD instance over the d-dimensional lattice spanned by $\mathbf{B}_1$ for the target vector $\mathbf{B}_2\mathbf{x}_2$. A sufficient condition to solve this problem using Babai's nearest plane algorithm Babai (1986) is that $|\langle \mathbf{b}_i^*, \mathbf{s}\rangle| \leq \frac{1}{2}\|\mathbf{b}_i^*\|^2$ for all $1 \leq i < d$. (See also Galbraith 2012, Chap. 18 for Babai's algorithms.) Since $|\langle \mathbf{b}_i^*, \mathbf{s}\rangle| \leq \|\mathbf{b}_i^*\|\|\mathbf{s}\|$, a further sufficient condition is that $\mathrm{GH}(L) \leq \frac{1}{2}\min_{i<d}\|\mathbf{b}_i^*\|$. This condition is far from tight, and it should not be a serious issue in practice. Indeed, even for a strongly reduced basis, the d first Gram–Schmidt lengths won't be much smaller than $\mathrm{GH}(L)$, say by more than a factor 2. (The BKZ-preprocessing with blocksize $\beta = \frac{n}{2}$ is assumed in Ducas (2018).) A concrete maximal value of d satisfying the condition (3) depends on the shape of a basis $\mathbf{B}$. It is estimated in Ducas (2018) that $d = \Theta(n/\log n)$ is suitable over a quasi-HKZ-reduced basis.

In 2019, Albrecht et al. (2019) proposed the General Sieve Kernel (G6K), an abstract stateful machine supporting a variety of advanced lattice reductions based on sieving algorithms. They have provided a highly optimized, multi-threaded, and tweakable implementation of G6K as an open-source C++ and Python library. A number of records in the hall of fame for the SVP challenge were found by the sub-sieving strategy on G6K. (In June 2019, the highest dimension to be solved in the SVP challenge is 157, using G6K.) Specifically, their experiments imply that in average $d = 11.46 + 0.0757n$ is a suitable free dimension of the sub-sieving strategy for the SVP challenge in high dimensions n. Furthermore, their solution for the SVP challenge in dimension 151 was found 400 times faster than the times reported for the SVP challenge in dimension 150, which was solved early in 2017 by the random sampling strategy.

Acknowledgements This work was supported by JST CREST Grant Number JPMJCR14D6, Japan. A part of this work was also supported by JSPS KAKENHI Grant Number JP16H02830.

References

M. Ajtai, Generating hard instances of lattice problems, in *Symposium on Theory of Computing (STOC 1996)* (ACM, 1996), pp. 99–108

M. Ajtai, R. Kumar, D. Sivakumar, A sieve algorithm for the shortest lattice vector problem, in *Symposium on Theory of Computing (STOC 2001)* (ACM, 2001), pp. 601–610

M. Albrecht, L. Ducas, G. Herold, E. Kirshanova, E.W. Postlethwaite, M. Stevens, The general sieve kernel and new records in lattice reduction. *Advances in Cryptology–EUROCRYPT 2019*, Lecture Notes in Computer Science, vol. 11477 (Springer, Berlin, 2019), pp. 717–746

M.R. Albrecht, B.R. Curtis, A. Deo, A. Davidson, R. Player, E.W. Postlethwaite, F. Virdia, T. Wunderer, Estimate all the LWE, NTRU schemes! *Security and Cryptography for Networks (SCN 2018)*, Lecture Notes in Computer Science, vol. 11035 (2018), pp. 351–367

Y. Aono, P.Q. Nguyen, Random sampling revisited: Lattice enumeration with discrete pruning. *Advances in Cryptology–EUROCRYPT 2017*, Lecture Notes in Computer Science, vol. 10211 (Springer, Berlin, 2017), pp. 65–102

Y. Aono, P.Q. Nguyen, T. Seito, J. Shikata, Lower bounds on lattice enumeration with extreme pruning. *Advances in Cryptology–CRYPTO 2018*, Lecture Notes in Computer Science, vol. 10992 (Springer, Berlin, 2018), pp. 608–637

Y. Aono, Y. Wang, T. Hayashi, T. Takagi, Improved progressive BKZ algorithms and their precise cost estimation by sharp simulator. *Advances in Cryptology–EUROCRYPT 2016*, Lecture Notes in Computer Science, vol. 9665 (Springer, Berlin, 2016), pp. 789–819. Progressive BKZ library is available from https://www2.nict.go.jp/security/pbkzcode/

L. Babai, On Lovász' lattice reduction and the nearest lattice point problem. Combinatorica **6**(1), 1–13 (1986)

S. Bai, T. Laarhoven, D. Stehlé, Tuple lattice sieving. LMS J. Comput. Math. **19**(A), 146–162 (2016)

M.R. Bremner, *Lattice Basis Reduction: An Introduction to the LLL Algorithm and Its Applications* (CRC Press, Bocca Raton, 2011)

M.S. Charikar, Similarity estimation techniques from rounding algorithms, in *Symposium on Theory of Computing (STOC 2002)* (ACM, 2002), pp. 380–388

Y. Chen, Réduction de réseau et sécurité concrete du chiffrement completement homomorphe. Ph.D. thesis, Paris 7 (2013)

Y. Chen, P.Q. Nguyen, BKZ 2.0: Better lattice security estimates. *Advances in Cryptology–ASIACRYPT 2011*, Lecture Notes in Computer Science, vol. 7073 (Springer, Berlin, 2011), pp. 1–20

H. Cohen, *A Course in Computational Algebraic Number Theory*, vol. 138, Graduate Texts in Mathematics (Springer Science & Business Media, Berlin, 2013)

T. Darmstadt, SVP challenge. (2010) https://www.latticechallenge.org/svp-challenge/

L. Ducas, Shortest vector from lattice sieving: a few dimensions for free. *Adavances in Cryptology–EUROCRYPT 2018*, Lecture Notes in Computer Science, , vol. 10820 (Springer, Berlin, 2018), pp. 125–145

U. Fincke, M. Pohst, Improved methods for calculating vectors of short length in a lattice, including a complexity analysis. Math. Comput. **44**(170), 463–471 (1985)

R. Fitzpatrick, C. Bischof, J. Buchmann, Ö. Dagdelen, F. Göpfert, A. Mariano, B.Y. Yang, Tuning Gauss Sieve for speed. *Progress in Cryptology–LATINCRYPT 2014*, Lecture Notes in Computer Science, vol. 8895 (Springer, 2014), pp. 288–305

M. Fukase, K. Kashiwabara, An accelerated algorithm for solving SVP based on statistical analysis. J. Inf. Process. (JIP) **23**(1), 67–80 (2015)

S.D. Galbraith, *Mathematics of Public Key Cryptography* (Cambridge University Press, Cambridge, 2012)

N. Gama, P.Q. Nguyen, Finding short lattice vectors within Mordell's inequality, in *Symposium on Theory of Computing (STOC 2008)* (ACM, 2008), pp. 207–216

N. Gama, P.Q. Nguyen, Predicting lattice reduction, *Advances in Cryptology–EUROCRYPT 2008*, Lecture Notes in Computer Science, vol. 4965 (Springer, Berlin, 2008), pp. 31–51

N. Gama, P.Q. Nguyen, O. Regev, Lattice enumeration using extreme pruning, *Advances in Cryptology–EUROCRYPT 2010*, Lecture Notes in Computer Science, vol. 6110 (Springer, Berlin, 2010), pp. 257–278

D. Goldstein, A. Mayer, *Forum Mathematicum*, vol. 15, On the equidistribution of Hecke points (De Gruyter, Berlin, 2003), pp. 165–190

G. Hanrot, X. Pujol, D. Stehlé, Analyzing blockwise lattice algorithms using dynamical systems, *Advances in Cryptology–CRYPTO 2011*, Lecture Notes in Computer Science, vol. 6841 (Springer, Berlin, 2011), pp. 447–464

C. Hermite, Extraits de lettres de M. Hermite à M. Jacobi sur différents objets de la théorie des nombres: Deuxième lettre. Journal für die Reine und Angewandte Mathematik (1850), pp. 279–315

G. Herold, E. Kirshanova, Improved algorithms for the approximate k-list problem in Euclidean norm, *Public Key Cryptography (PKC 2017)*, Lecture Notes in Computer Science, vol. 10174 (Springer, Berlin, 2017), pp. 16–40

R. Kannan, Improved algorithms for integer programming and related lattice problems, in *Symposium on Theory of Computing (STOC 1983)* (ACM, 1983), pp. 193–206

R. Kannan, Minkowski's convex body theorem and integer programming. Math. Oper. Res. **12**(3), 415–440 (1987)

A.K. Lenstra, H.W. Lenstra, L. Lovász, Factoring polynomials with rational coefficients. Mathematische Annalen **261**(4), 515–534 (1982)

J. Martinet, *Comprehensive Studies in Mathematics*, vol. 327, Perfect lattices in Euclidean spaces (Springer Science & Business Media, Berlin, 2013)

Y. Matsuda, T. Teruya, K. Kashiwabara, Estimation of the success probability of random sampling by the Gram-Charlier approximation. IACR ePrint 2018/815 (2018)

D. Micciancio, P. Voulgaris, Faster exponential time algorithms for the shortest vector problem, in *Symposium on Discrete Algorithms (SODA 2010)* (ACM-SIAM, 2010), pp. 1468–1480

D. Micciancio, M. Walter, Fast lattice point enumeration with minimal overhead, in *Symposium on Discrete algorithms (SODA 2014)* (ACM-SIAM, 2014), pp. 276–294

D. Micciancio, M. Walter, Practical, predictable lattice basis reduction, *Advances in Cryptology–EUROCRYPT 2016*, Lecture Notes in Computer Science, vol. 9665 (Springer, Berlin, 2016), pp. 820–849

P.Q. Nguyen, *The LLL Algorithm*, Hermite's constant and lattice algorithms (Springer, Berlin, 2009), pp. 19–69

P.Q. Nguyen, T. Vidick, Sieve algorithms for the shortest vector problem are practical. J. Math. Cryptol. **2**(2), 181–207 (2008)

M. Pohst, On the computation of lattice vectors of minimal length, successive minima and reduced bases with applications. ACM Sigsam Bull. **15**(1), 37–44 (1981)

M. Pohst, A modification of the LLL reduction algorithm. J. Symb. Comput. **4**(1), 123–127 (1987)

C.P. Schnorr, A hierarchy of polynomial time lattice basis reduction algorithms. Theor. Comput. Sci. **53**(2–3), 201–224 (1987)

C.P. Schnorr, Block Korkin-Zolotarev bases and successive minima. International Computer Science Institute (1992)

C.P. Schnorr, Lattice reduction by random sampling and birthday methods, *Symposium on Theoretical Aspects of Computer Science (STACS 2003)*, Lecture Notes in Computer Science, vol. 2607 (Springer, Berlin, 2003), pp. 145–156

C.P. Schnorr, M. Euchner, Lattice basis reduction: Improved practical algorithms and solving subset sum problems. Math. Program. **66**, 181–199 (1994)

Shor, P.W.: Algorithms for quantum computation: discrete logarithms and factoring, in *Symposium on Foundations of Computer Science (FOCS 1994)* (IEEE, 1994), pp. 124–134

C.C. Sims, *Computation with Finitely Presented Groups*, vol. 48 (Cambridge University Press, Cambridge, 1994)

T. Teruya, K. Kashiwabara, G. Hanaoka, Fast lattice basis reduction suitable for massive parallelization and its application to the shortest vector problem, *Public Key Cryptography (PKC 2018)*, Lecture Notes in Computer Science, vol. 10769 (Springer, Berlin, 018), pp. 437–460

The FPLLL development team: fplll, a lattice reduction library (2016), https://github.com/fplll/fplll

The National Institute of Standards and Technology (NIST): Post-quantum cryptography. (2016) https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization

J. Yamaguchi, M. Yasuda, Explicit formula for Gram-Schmidt vectors in LLL with deep insertions and its applications, *Number-Theoretic Methods in Cryptology (NuTMiC 2017)*, Lecture Notes in Computer Science, vol. 10737 (Springer, Berlin, 2017), pp. 142–160

M. Yasuda, Self-dual DeepBKZ for finding short lattice vectors. J. Math. Cryptol. **14**(1), 84–94 (2020)

M. Yasuda, J. Yamaguchi, A new polynomial-time variant of LLL with deep insertions for decreasing the squared-sum of Gram-Schmidt lengths. Des. Codes Cryptogr. **87**, 2489–2505 (2019)

M. Yasuda, J. Yamaguchi, M. Ooka, S. Nakamura, Development of a dual version of DeepBKZ and its application to solving the LWE challenge, *Progress in Cryptology–AFRICACRYPT 2018*, vol. 10831, Lecture Notes in Computer Science (Springer, Berlin, 2018), pp. 162–182

M. Yasuda, K. Yokoyama, T. Shimoyama, J. Kogure, T. Koshiba, Analysis of decreasing squared-sum of Gram-Schmidt lengths for short lattice vectors. J. Math. Cryptol. **11**(1), 1–24 (2017)

Y. Yu, L. Ducas, Second order statistical behavior of LLL and BKZ, *Selected Areas in Cryptography (SAC 2017)*, Lecture Notes in Computer Science, vol. 10719 (Springer, Berlin, 2017), pp. 3–22