Designs, Codes and Cryptography (2019) 87:2489–2505
https://doi.org/10.1007/s10623-019-00634-9

A new polynomial-time variant of LLL with deep insertions for decreasing the squared-sum of Gram–Schmidt lengths

Masaya Yasuda[1] · **Junpei Yamaguchi**[2]

Received: 15 March 2018 / Revised: 23 March 2019 / Accepted: 6 April 2019 / Published online: 19 April 2019

Abstract
Lattice basis reduction algorithms have been used in cryptanalysis. The most famous algorithm is LLL, proposed by Lenstra, Lenstra, Lovász, and one of its typical improvements is LLL with deep insertions (DeepLLL). A DeepLLL-reduced basis is LLL-reduced, and hence its quality is at least as good as LLL. In practice, DeepLLL often outputs a more reduced basis than LLL, but no theoretical result is known. First, we show provable output quality of DeepLLL, strictly better than that of LLL. Second, as a main work of this paper, we propose a new variant of DeepLLL. The squared-sum of Gram–Schmidt lengths of a basis is related with the computational hardness of lattice problems such as the shortest vector problem (SVP). Given an input basis, our variant monotonically decreases the squared-sum by a given factor at every deep insertion. This guarantees that our variant runs in polynomial-time.

Keywords Lattice basis reduction · LLL with deep insertions · Shortest vector problem (SVP) · Shortest diagonal problem (SDP)

Mathematics Subject Classification 68R01 · 06B99

1 Introduction

For a positive integer n, let $\mathbf{b}_1, \ldots, \mathbf{b}_n \in \mathbb{R}^n$ be n linearly independent column vectors. The set of integral linear combinations of $\mathbf{b}_i$ is called a (full-rank) *lattice* of dimension n. The $n \times n$ matrix $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_n]$ is called a *basis* of the lattice. From a given basis, lattice

Communicated by T. Iwata.

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.

✉ Masaya Yasuda
yasuda@imi.kyushu-u.ac.jp

Junpei Yamaguchi
ma216010@math.kyushu-u.ac.jp

[1] Institute of Mathematics for Industry, Kyushu University, Fukuoka, Japan

[2] Graduate School of Mathematics, Kyushu University, Fukuoka, Japan

basis reduction aims to find a new basis with short and nearly orthogonal basis vectors. The LLL algorithm [17] is the most famous, and it computes a reduced basis with provable output quality in polynomial-time (see [20] for details). A blockwise generalization of LLL is the block Korkine-Zolotarev (BKZ) reduction algorithm proposed by Schnorr and Euchner [22]. Recently, BKZ 2.0 [6], terminating-BKZ [15], and progressive-BKZ [3] have been developed as efficient variants of BKZ. Another improvement of LLL was suggested also in [22], whose idea is called a *deep insertion*. While only adjacent basis vectors are swapped in LLL, *non-adjacent vectors* can be swapped in DeepLLL. Although the output quality of DeepLLL is better than LLL in practice (see [13] for experimental results), the complexity of DeepLLL is potentially super-exponential. In order to address this problem, Fontein, Schneider and Wagner [11] proposed variants of DeepLLL, which are proven to run in polynomial-time. As a recent work, Yamaguchi and Yasuda [25] embed DeepLLL into BKZ as a subroutine alternative to LLL, called DeepBKZ, and have found a number of new solutions for the Darmstadt SVP challenge [8] of dimensions from 102 to 127 over a single thread of a general-purpose server with an Intel Xeon CPU E5-2670@2.60GHz.

Since any DeepLLL-reduced basis is LLL-reduced, it satisfies the provable output quality of LLL. In this paper, we show the following provable output quality of DeepLLL, better than that of LLL (since $1 + \frac{\alpha}{4} < \alpha$ due to $\alpha > \frac{4}{3}$, the following properties are stronger than [20, Theorem 9 in Chapter 2]):

Theorem 1 *For a reduction parameter $\frac{1}{4} < \delta < 1$, set $\alpha = \frac{4}{4\delta - 1}$. Let $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_n]$ be a δ-DeepLLL-reduced basis of a lattice L. Then*

1. $\|\mathbf{b}_1\| \leq \alpha^{(n-1)/2n} \left(1 + \frac{\alpha}{4}\right)^{(n-1)(n-2)/4n} \mathrm{vol}(L)^{1/n}$.
2. *For all* $1 \leq i \leq n$, $\|\mathbf{b}_i\| \leq \alpha^{1/2} \left(1 + \frac{\alpha}{4}\right)^{(n-2)/2} \lambda_i(L)$.
3. $\prod_{i=1}^{n} \|\mathbf{b}_i\| \leq \left(1 + \frac{\alpha}{4}\right)^{n(n-1)/4} \mathrm{vol}(L)$.

Here $\mathrm{vol}(L)$ *is the volume of* L *and* $\lambda_i(L)$ *the* i*-th successive minimum of* L *for* $1 \leq i \leq n$ (*see* [20, *Definition* 13 *in Chapter* 2] *for definition*).

For a basis $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_n]$, let $\mathbf{B}^* = [\mathbf{b}_1^*, \ldots, \mathbf{b}_n^*]$ denote its Gram-Schmidt orthogonalization (GSO). The squared-sum of the lengths of the GSO vectors is defined as

$$\mathrm{SS}(\mathbf{B}) := \sum_{i=1}^{n} \|\mathbf{b}_i^*\|^2.$$

Let $\sigma(L)$ denote the smallest value of $\mathrm{SS}(\mathbf{B})^{1/2}$ when $\mathbf{B}$ ranges over all bases for a lattice L, and $\rho(L)$ the covering radius of L. By [19, Theorem 7.9], we have $\lambda_n(L) \leq 2\rho(L) \leq \sigma(L) \leq \sqrt{n}\lambda_n(L)$. Given a basis of L, the shortest diagonal problem (SDP) is to find a new basis $\mathbf{B}$ with $\mathrm{SS}(\mathbf{B}) = \sigma(L)^2$. This is related with various lattice problems [19, Figure 7.1]. For example, the size of $\mathrm{SS}(\mathbf{B})$ is related with the closest vector problem (CVP) since one can find a lattice vector within distance $\frac{1}{2}\mathrm{SS}(\mathbf{B})^{1/2}$ from a target vector by Babai's algorithm [4]. Recently, Kashiwabara and Teruya have found new solutions for the Darmstadt SVP challenge of dimensions from 134 to 150 with significant computational power over massively parallelized servers [24]. (cf., Around September 2018, many records had been updated by the sub-sieving technique [9] for dimensions up to 155 over less 80 threads with 256 or 512 GByte RAM.) Kashiwabara-Teruya's implementation relies on the strategy of [10], based on Schnorr's random sampling [21]. In their preprocessing, they manage to decrease $\mathrm{SS}(\mathbf{B})$

as much as possible to sample shorter lattice vectors (see [26] for analysis of [10]). It is mentioned also in [2, Section 6.1] that more short lattice vectors could be sampled over a basis $\mathbf{B}$ with smaller $\mathrm{SS}(\mathbf{B})$. (In a recent paper [18], Matsuda et al. gave a deep investigation for the strategy of [10] using the Gram-Charlier approximation, in order to precisely estimate the success probability of sampling short lattice vectors over a reduced basis.) In this paper, we propose a new variant of DeepLLL, which decreases $\mathrm{SS}(\mathbf{B})$ by a given factor $0 < \eta \leq 1$ at every deep insertion. In particular, the GSO formula [25, Theorem 1] enables us to efficiently compute the difference between $\mathrm{SS}(\mathbf{B})$ and $\mathrm{SS}(\mathbf{C})$ before updating the GSO of the new basis $\mathbf{C}$ obtained by a deep insertion. The complexity of our variant is bounded by the size of $\mathrm{SS}(\mathbf{B})$, and it runs in polynomial-time for $0 < \eta < 1$.

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. In this paper, we represent all vectors in *column format*. For $\mathbf{a} = (a_1, \ldots, a_n)^\top \in \mathbb{R}^n$, let $\|\mathbf{a}\|$ denote its Euclidean norm. For $\mathbf{a} = (a_1, \ldots, a_n)^\top$ and $\mathbf{b} = (b_1, \ldots, b_n)^\top \in \mathbb{R}^n$, let $\langle \mathbf{a}, \mathbf{b} \rangle$ denote the inner product $\sum_{i=1}^n a_i b_i$.

2 Preliminaries: lattices, LLL and DeepLLL

In this section, we review lattices and the LLL algorithm [17]. We also present the DeepLLL algorithm [22] and its variants.

2.1 Lattices and GSO

For a positive integer n, linearly independent vectors $\mathbf{b}_1, \ldots, \mathbf{b}_n \in \mathbb{Z}^n$ define a (full-rank) *lattice* (we consider only integral lattices for simplicity)

$$L = \left\{ \sum_{i=1}^n x_i \mathbf{b}_i \in \mathbb{Z}^n : x_i \in \mathbb{Z} \ (1 \leq \forall i \leq n) \right\}$$

of dimension n with basis $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_n] \in \mathbb{Z}^{n \times n}$. Every lattice has infinitely many bases; If $\mathbf{B}_1$ and $\mathbf{B}_2$ are two bases of the same lattice, then there exists a unimodular matrix $\mathbf{V} \in \mathrm{GL}_n(\mathbb{Z})$ such that $\mathbf{B}_1 = \mathbf{B}_2 \mathbf{V}$. Given a basis $\mathbf{B}$ of L, the *volume* of L is defined as $\mathrm{vol}(L) := |\det(\mathbf{B})|$, independent of the choice of bases. The *GSO* of $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_n]$ is the orthogonal family $[\mathbf{b}_1^*, \ldots, \mathbf{b}_n^*]$, recursively defined by

$$\begin{cases} \mathbf{b}_1^* := \mathbf{b}_1, \\ \mathbf{b}_i^* := \mathbf{b}_i - \sum_{j=1}^{i-1} \mu_{i,j} \mathbf{b}_j^*, \quad \mu_{i,j} := \dfrac{\langle \mathbf{b}_i, \mathbf{b}_j^* \rangle}{\|\mathbf{b}_j^*\|^2} \quad \text{for } 1 \leq j < i \leq n. \end{cases}$$

Every basis should be regarded as an *ordered* set for its GSO since the GSO vectors depend on the order of basis vectors. Let $\mathbf{B}^* = [\mathbf{b}_1^*, \ldots, \mathbf{b}_n^*] \in \mathbb{Q}^{n \times n}$ and $\mathbf{U} = (\mu_{i,j}) \in \mathbb{Q}^{n \times n}$, where we set $\mu_{i,i} = 1$ for all i and $\mu_{i,j} = 0$ for all $j > i$. Then $\mathbf{B} = \mathbf{B}^* \mathbf{U}^\top$ and $\mathrm{vol}(L) = \prod_{i=1}^n \|\mathbf{b}_i^*\|$. For $2 \leq \ell \leq n$, let π_ℓ denote the orthogonal projection from $\mathbb{R}^n$ over the orthogonal supplement of the $\mathbb{R}$-vector space $\langle \mathbf{b}_1, \ldots, \mathbf{b}_{\ell-1} \rangle_{\mathbb{R}}$. We set $\pi_1 = \mathrm{id}$ (the identity map). Specifically, we have $\pi_\ell(\mathbf{x}) = \sum_{i=\ell}^n \frac{\langle \mathbf{x}, \mathbf{b}_i^* \rangle}{\|\mathbf{b}_i^*\|^2}$ for any vector $\mathbf{x} \in \mathbb{R}^n$.

2.2 LLL-reduction

Definition 1 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 GSO with coefficients $\mu_{i,j}$. For a parameter $\frac{1}{4} < \delta < 1$, the basis $\mathbf{B}$ is called δ-*LLL-reduced* if the following two conditions are satisfied:

(i) (Size-reduced) $|\mu_{i,j}| \leq \frac{1}{2}$ 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 $2 \leq k \leq n$. Since $\pi_{k-1}(\mathbf{b}_k) = \mathbf{b}_k^* + \mu_{k,k-1}\mathbf{b}_{k-1}^*$, this condition can be rewritten as

$$\|\mathbf{b}_k^*\|^2 \geq (\delta - \mu_{k,k-1}^2)\|\mathbf{b}_{k-1}^*\|^2. \tag{1}$$

Any LLL-reduced basis satisfies the following properties (e.g., see [20, Theorem 9 in Chapter 2] or [5, Theorems 4.7 and 4.8] for details):

Theorem 2 *For a reduction parameter $\frac{1}{4} < \delta < 1$, set $\alpha = \frac{4}{4\delta - 1}$ as Theorem 1. Let $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_n]$ be a δ-LLL-reduced basis of a lattice L. Then*

1. $\|\mathbf{b}_1\| \leq \alpha^{(n-1)/4}\mathrm{vol}(L)^{1/n}$.
2. *For all* $1 \leq i \leq n$, $\|\mathbf{b}_i\| \leq \alpha^{(n-1)/2}\lambda_i(L)$.
3. $\prod_{i=1}^n \|\mathbf{b}_i\| \leq \alpha^{n(n-1)/4}\mathrm{vol}(L)$.

Note that these properties are strictly weaker than that of Theorem 1.

2.3 DeepLLL-reduction and the DeepLLL algorithm

In LLL [17], only adjacent basis vectors $\mathbf{b}_{k-1}$ and $\mathbf{b}_k$ can be swapped. In the DeepLLL algorithm [22], *non-adjacent* basis vectors can be changed; For a reduction parameter $\frac{1}{4} < \delta < 1$, a basis vector $\mathbf{b}_k$ is inserted between $\mathbf{b}_{i-1}$ and $\mathbf{b}_i$ for $1 \leq i < k \leq n$ if the *deep exchange condition* $\|\pi_i(\mathbf{b}_k)\|^2 < \delta\|\mathbf{b}_i^*\|^2$ is satisfied. In this case, the new GSO vector at the i-th position is given by $\pi_i(\mathbf{b}_k)$, which is strictly shorter than the old GSO vector $\mathbf{b}_i^*$. The notion of DeepLLL-reduction is defined as follows:

Definition 2 For a reduction parameter $\frac{1}{4} < \delta < 1$, a basis $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_n]$ is called δ-*DeepLLL-reduced* if the following two conditions are satisfied:

(i) The basis $\mathbf{B}$ is size-reduced.
(ii) We have $\|\pi_i(\mathbf{b}_k)\|^2 \geq \delta\|\mathbf{b}_i^*\|^2$ for all $1 \leq i < k \leq n$ [(in particular, the case $i = k - 1$ is equivalent to Lovász' condition (1)].

Let $\mathfrak{S}_n$ denote the group of permutations among n elements. Given an element $\sigma \in \mathfrak{S}_n$ and a basis $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_n]$, let $\sigma(\mathbf{B}) := [\mathbf{b}_{\sigma(1)}, \ldots, \mathbf{b}_{\sigma(n)}]$ denote the reordered basis. For $1 \leq i < k \leq n$, we define $\sigma_{i,k} \in \mathfrak{S}_n$ as $\sigma_{i,k}(\ell) = \ell$ for $\ell < i$ or $\ell > k$, $\sigma_{i,k}(i) = k$, and $\sigma_{i,k}(\ell) = \ell - 1$ for $i + 1 \leq \ell \leq k$. In other words, the reordered basis is given by

$$\sigma_{i,k}(\mathbf{B}) = [\mathbf{b}_1, \ldots, \mathbf{b}_{i-1}, \mathbf{b}_k, \mathbf{b}_i, \ldots, \mathbf{b}_{k-1}, \mathbf{b}_{k+1}, \ldots, \mathbf{b}_n],$$

which is obtained by inserting $\mathbf{b}_k$ between $\mathbf{b}_{i-1}$ and $\mathbf{b}_i$ (i.e., a deep insertion). In Algorithm 1, we present the DeepLLL algorithm to find a DeepLLL-reduced basis [22] (see also [5, Figure 5.1] or [7, Algorithm 2.6.4] for more details). Here we present the explicit GSO formula [25, Theorem 1] for the reordered basis:

Algorithm 1 DeepLLL [22]

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: Set $k \leftarrow 2$
2: **while** $k \leq n$ **do**
3: Size-reduce $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_n]$ /* At each k, we recursively change $\mathbf{b}_k \leftarrow \mathbf{b}_k - \lceil \mu_{k,j} \rfloor \mathbf{b}_j$ for $1 \leq j \leq k-1$ (e.g., see [12, Algorithm 24] for details) */
4: Set $C \leftarrow \|\mathbf{b}_k\|^2$ and $i \leftarrow 1$
5: **while** $i < k$ **do**
6: **if** $C \geq \delta \|\mathbf{b}_i^*\|^2$ **then**
7: Compute $C \leftarrow C - \mu_{k,i}^2 \|\mathbf{b}_i^*\|^2$ and set $i \leftarrow i+1$ /* $C = \|\pi_i(\mathbf{b}_k)\|^2$ */
8: **else**
9: Set $\mathbf{B} \leftarrow \sigma_{i,k}(\mathbf{B})$ and update the GSO of $\mathbf{B}$ /* A deep insertion */
10: Set $k \leftarrow \max(i, 2)$ and go back to step 3
11: **end if**
12: **end while**
13: Set $k \leftarrow k+1$
14: **end while**

Theorem 3 *Let $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_n]$ be a basis, and $\mathbf{B}^* = [\mathbf{b}_1^*, \ldots, \mathbf{b}_n^*]$ its GSO with coefficients $\mu_{i,j}$ and $B_j = \|\mathbf{b}_j^*\|^2$. For $1 \leq i < k \leq n$, let $\mathbf{C} = \sigma_{i,k}(\mathbf{B})$, and $\mathbf{C}^* = [\mathbf{c}_1^*, \ldots, \mathbf{c}_n^*]$ its GSO with $C_j = \|\mathbf{c}_j^*\|^2$. Then $C_i = D_i^{(k)}$ and*

$$C_j = \frac{D_j^{(k)}}{D_{j-1}^{(k)}} B_{j-1}$$

for $i+1 \leq j \leq k$, where set $D_\ell^{(k)} = \|\pi_\ell(\mathbf{b}_k)\|^2 = \sum_{j=\ell}^{k} \mu_{k,j}^2 B_j$ for $1 \leq \ell \leq k$.

2.4 Variants of the DeepLLL algorithm

While the complexity of LLL is polynomial-time in the dimension of an input basis, DeepLLL has potentially super-exponential complexity and no provable upper bound is known. In order to address the problem, Schnorr and Euchner in [22, Comments in Section 3] proposed insertion restriction; In step 9 of Algorithm 1, deep insertions $\mathbf{B} \leftarrow \sigma_{i,k}(\mathbf{B})$ are performed only in case of either $i < \varepsilon$ or $k - i \leq \varepsilon$ for fixed ε. DeepLLL with such restriction seems to run in polynomial-time in practice. However, such restriction does not guarantee polynomial-time complexity. In contrast, Fontein, Schneider and Wagner [11] proposed two polynomial-time variants of DeepLLL, called "PotLLL" and "PotLLL2". To introduce their variants, let us define the following value:

Definition 3 The *potential* of a basis $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_n]$ is defined as

$$\mathrm{Pot}(\mathbf{B}) := \prod_{i=1}^{n} \mathrm{vol}(L_i)^2 = \prod_{i=1}^{n} \|\mathbf{b}_i^*\|^{2(n-i+1)},$$

where L_i is the lattice spanned by $[\mathbf{b}_1, \ldots, \mathbf{b}_i]$ for $1 \leq i \leq n$.

The potential plays an important role in showing that LLL is a polynomial-time algorithm (e.g., see [5, Theorem 4.19]). Fontein et al. give the following key lemma [11, Lemma 1] for complexity analysis of their variants, and define the following notion of reduction [11, Definition 4]:

Lemma 1 *Let* $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_n]$ *be a basis. For* $1 \le i < k \le n$, *let* $\mathbf{C} = \sigma_{i,k}(\mathbf{B})$ *denote the reordered basis by a deep insertion. Then we have*

$$\mathrm{Pot}(\mathbf{C}) = \mathrm{Pot}(\mathbf{B}) \prod_{j=i}^{k-1} \frac{\|\pi_j(\mathbf{b}_k)\|^2}{\|\mathbf{b}_j^*\|^2}.$$

Proof Fixing k, it is proved in [11] by induction on i from $k-1$ to 1. With Theorem 3, we can *directly* prove it; Here we simply write D_ℓ for $D_\ell^{(k)}$. Then

$$\begin{aligned}
\frac{\mathrm{Pot}(\mathbf{C})}{\mathrm{Pot}(\mathbf{B})} &= \frac{D_i^{n-i+1} \times \prod_{j=i+1}^{k} \left(\frac{D_j}{D_{j-1}} B_{j-1}\right)^{n-j+1}}{\prod_{j=i}^{k} B_j^{n-j+1}} \\
&= \frac{D_i^{n-i+1} \times \left(\frac{D_{i+1}}{D_i}\right)^{n-i} \left(\frac{D_{i+2}}{D_{i+1}}\right)^{n-i-1} \cdots \left(\frac{D_k}{D_{k-1}}\right)^{n-k+1}}{B_i \cdots B_{k-1} \cdot B_k^{n-k+1}} \\
&= \frac{D_i \cdots D_{k-1} \cdot D_k^{n-k+1}}{B_i \cdots B_{k-1} \cdot B_k^{n-k+1}} = \frac{D_i \cdots D_{k-1}}{B_i \cdots B_{k-1}}
\end{aligned}$$

since $D_k = B_k$. This completes the proof. □

Definition 4 For a reduction parameter $\frac{1}{4} < \delta < 1$, a basis $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_n]$ is called *δ-PotLLL-reduced* if the following two conditions are satisfied:

(i) The basis $\mathbf{B}$ is size-reduced.
(ii) We have $\delta\mathrm{Pot}(\mathbf{B}) \le \mathrm{Pot}(\sigma_{i,k}(\mathbf{B}))$ for all $1 \le i < k \le n$.

For $\frac{1}{4} < \delta < 1$, any δ-PotLLL-reduced basis $\mathbf{B}$ is also δ-LLL-reduced since $\delta\mathrm{Pot}(\mathbf{B}) \le \mathrm{Pot}(\sigma_{k-1,k}(\mathbf{B}))$ if and only if $\delta\|\mathbf{b}_{k-1}^*\|^2 \le \|\pi_{k-1}(\mathbf{b}_k)\|^2$ by Lemma 1, which is just Lovász' condition (1). The PotLLL algorithm is shown in [11, Algorithm 1] to obtain a δ-PotLLL-reduced basis. Specifically, it computes the index $i = \mathrm{argmin}_{1 \le j < k} \mathrm{Pot}(\sigma_{j,k}(\mathbf{B}))$ for $2 \le k \le n$, and it performs a deep insertion if $\delta\mathrm{Pot}(\mathbf{B}) > \mathrm{Pot}(\sigma_{i,k}(\mathbf{B}))$. In PotLLL, every deep insertion decreases $\mathrm{Pot}(\mathbf{B})$ by a factor of at least δ for an input basis $\mathbf{B}$. This is same as in LLL, and hence the complexity of PotLLL is polynomial-time [11, Proposition 1].

3 Proof of Theorem 1

Here we prove Theorem 1. Let $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_n]$ be a δ-DeepLLL-reduced basis for $\frac{1}{4} < \delta < 1$. Let $\mathbf{B}^* = [\mathbf{b}_1^*, \ldots, \mathbf{b}_n^*]$ denote its GSO with coefficients $\mu_{i,j}$. For $\alpha = \frac{4}{4\delta - 1}$, we first show that

$$\|\mathbf{b}_i^*\|^2 \le \alpha \left(1 + \frac{\alpha}{4}\right)^{k-i-1} \|\mathbf{b}_k^*\|^2 \tag{2}$$

for all $1 \le i < k \le n$. Fix $2 \le k \le n$. We shall prove (2) by induction on index i from $k-1$ to 1; The case $i = k-1$ is equivalent to $\|\mathbf{b}_{k-1}^*\|^2 \le \alpha\|\mathbf{b}_k^*\|^2$, which holds for any δ-DeepLLL-reduced basis (even for any δ-LLL-reduced basis). For $2 \le j \le k-1$, we assume inequality (2) holds for $i = j, \ldots, k-1$, and consider the case of $i = j-1$. It follows from two conditions (i) and (ii) in Definition 2 that we have

$$\delta \|\mathbf{b}_{j-1}^*\|^2 \le \|\pi_{j-1}(\mathbf{b}_k)\|^2 \le \|\mathbf{b}_k^*\|^2 + \frac{1}{4}\sum_{h=j-1}^{k-1} \|\mathbf{b}_h^*\|^2.$$

Combining this with inequality (2) for $j \le i \le k-1$, we have

$$\left(\delta - \frac{1}{4}\right)\|\mathbf{b}_{j-1}^*\|^2 \le \|\mathbf{b}_k^*\|^2 + \frac{1}{4}\sum_{h=j}^{k-1}\|\mathbf{b}_h^*\|^2 \le \left(1 + \frac{\alpha}{4}\sum_{h=j}^{k-1}\left(1+\frac{\alpha}{4}\right)^{k-h-1}\right)\|\mathbf{b}_k^*\|^2 = \left(1+\frac{\alpha}{4}\right)^{k-j}\|\mathbf{b}_k^*\|^2.$$

This shows that inequality (2) holds for the case $i = j-1$, and thus it completes the proof of (2) by induction.

From inequality (2), we can easily prove Theorem 1 as follows:

1. From inequality (2), we have $\|\mathbf{b}_1\|^2 \le \alpha\left(1+\frac{\alpha}{4}\right)^{k-2}\|\mathbf{b}_k^*\|^2$ for all $2 \le k \le n$. By multiplying the inequalities, we have

$$\|\mathbf{b}_1\|^{2n} \le \|\mathbf{b}_1^*\|^2 \prod_{k=2}^{n} \alpha\left(1+\frac{\alpha}{4}\right)^{k-2}\|\mathbf{b}_k^*\|^2 \le \alpha^{n-1}\left(1+\frac{\alpha}{4}\right)^{(n-1)(n-2)/2}\mathrm{vol}(L)^2.$$

   This implies $\|\mathbf{b}_1\| \le \alpha^{(n-1)/2n}\left(1+\frac{\alpha}{4}\right)^{(n-1)(n-2)/4n}\mathrm{vol}(L)^{1/n}$.

2. From condition (i) in Definition 2 and inequality (2), we have

$$\|\mathbf{b}_i\|^2 \le \|\mathbf{b}_i^*\|^2 + \frac{1}{4}\sum_{j=1}^{i-1}\|\mathbf{b}_j^*\|^2 \le \left(1+\frac{1}{4}\sum_{j=1}^{i-1}\alpha\left(1+\frac{\alpha}{4}\right)^{i-j-1}\right)\|\mathbf{b}_i^*\|^2 = \left(1+\frac{\alpha}{4}\right)^{i-1}\|\mathbf{b}_i^*\|^2 \tag{3}$$

   for all $1 \le i \le n$. On the other hand, we see from (2) that

$$\|\mathbf{b}_i^*\|^2 \le \alpha\left(1+\frac{\alpha}{4}\right)^{n-i-1}\min_{i\le j\le n}\|\mathbf{b}_j^*\|^2$$

   for all $1 \le i \le n$. Furthermore, since $\lambda_i(L) \ge \min_{i\le j\le n}\|\mathbf{b}_j^*\|$ [20, Lemma 7 in Chapter 2], we have $\|\mathbf{b}_i^*\|^2 \le \alpha\left(1+\frac{\alpha}{4}\right)^{n-i-1}\lambda_i(L)^2$. Combining this with (3), we obtain

$$\|\mathbf{b}_i\|^2 \le \alpha\left(1+\frac{\alpha}{4}\right)^{n-2}\lambda_i(L)^2.$$

3. By multiplying inequalities (3) for $1 \leq i \leq n$, we have

$$\prod_{i=1}^{n} \|\mathbf{b}_i\|^2 \leq \mathrm{vol}(L)^2 \prod_{i=1}^{n} \left(1+\frac{\alpha}{4}\right)^{i-1} = \left(1+\frac{\alpha}{4}\right)^{n(n-1)/2} \mathrm{vol}(L)^2.$$

This completes the proof of Theorem 1. □

4 New polynomial-time variant of DeepLLL

In this section, we propose a new polynomial-time variant of DeepLLL, and show experimental results on our variant.

4.1 S^{2}LLL-reduction

Given a basis $\mathbf{B}$, our variant aims to decrease $\mathrm{SS}(\mathbf{B})$ by every deep insertion. Since LLL decreases $\mathrm{SS}(\mathbf{B})$ by every swap [26, Lemma 5.2], ours can be regarded as a generalization of LLL in terms of decreasing $\mathrm{SS}(\mathbf{B})$. While the complexity of LLL and PotLLL for an input basis $\mathbf{B}$ is bounded by the size of $\mathrm{Pot}(\mathbf{B})$, the complexity of our variant is bounded by the size of $\mathrm{SS}(\mathbf{B})$. We define a new notion of reduction as follows:

Definition 5 For $0 < \eta \leq 1$, a basis $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_n]$ is called *η-S^{2}LLL-reduced* if the following two conditions are satisfied:

(i) The basis $\mathbf{B}$ is size-reduced.
(ii) We have $\eta \mathrm{SS}(\mathbf{B}) \leq \mathrm{SS}(\sigma_{i,k}(\mathbf{B}))$ for all $1 \leq i < k \leq n$.

In particular, when $\eta = 1$, we simply call the basis $\mathbf{B}$ to be *S^{2}LLL-reduced*.

S^{2}LLL-reduced bases have a local property; If $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_n]$ is S^{2}LLL-reduced, then $\mathbf{B}' = [\pi_i(\mathbf{b}_i), \pi_i(\mathbf{b}_{i+1}), \ldots, \pi_i(\mathbf{b}_j)]$ is also S^{2}LLL-reduced for all $1 \leq i \leq j \leq n$. We remark that any orthogonal basis $\mathbf{B}$ is η-S^{2}LLL-reduced for any $0 < \eta \leq 1$ since $\mathrm{SS}(\mathbf{B}) = \mathrm{SS}(\sigma_{i,k}(\mathbf{B})) = \sum_{i=1}^{n} \|\mathbf{b}_i\|^2$ for any $1 \leq i < k \leq n$. Therefore it is clear that any η-S^{2}LLL reduced basis is not always LLL-reduced (e.g., for $\mathbf{b}_1 = (3, 0)^\top$ and $\mathbf{b}_2 = (0, 1)^\top$, the basis $\mathbf{B} = [\mathbf{b}_1, \mathbf{b}_2]$ is η-S^{2}LLL-reduced for any $0 < \eta \leq 1$, but it is not δ-LLL-reduced for any $\frac{1}{4} < \delta < 1$). In contrast, for non-orthogonal bases, we have the following relation:

Proposition 1 *For $0 < \eta \leq 1$, let $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_n]$ be an η-S^{2}LLL-reduced basis, and $\mathbf{B}^* = [\mathbf{b}_1^*, \ldots, \mathbf{b}_n^*]$ its GSO with coefficients $\mu_{i,j}$ and $B_j = \|\mathbf{b}_j^*\|^2$. Assume $\mu_{j+1,j} \neq 0$ for all $1 \leq j \leq n-1$. Set $N = \min_{1 \leq j \leq n-1} \left(\mu_{j+1,j}^2 B_j\right) > 0$ and*

$$M = \frac{N}{(1-\eta)\mathrm{SS}(\mathbf{B}) + N}.$$

If $M > \frac{1}{4}$, then $\mathbf{B}$ is δ-LLL-reduced for any $\frac{1}{4} < \delta < M$. In particular, any S^{2}LLL-reduced basis (i.e., $\eta = 1$) is also δ-LLL-reduced for any $\frac{1}{4} < \delta < 1$.

Proof Suppose that a pair $(\mathbf{b}_{k-1}, \mathbf{b}_k)$ does not satisfy Lovász' condition (1) for some $2 \leq k \leq n$ and $\frac{1}{4} < \delta < M$. Let $\mathbf{C} = \sigma_{k-1,k}(\mathbf{B}) = [\mathbf{c}_1, \ldots, \mathbf{c}_n]$ denote the basis obtained by swapping $\mathbf{b}_k$ with $\mathbf{b}_{k-1}$. By [26, Lemma 5.2], we have

$$\mathrm{SS}(\mathbf{B}) - \mathrm{SS}(\mathbf{C}) = \frac{\mu_{k,k-1}^2(1-\delta_{k-1})}{\delta_{k-1}} B_{k-1}, \tag{4}$$

Algorithm 2 S^{2}LLL

Input: A basis $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_n]$ of a lattice L, and a factor $0 < \eta \leq 1$
Output: An η-S^{2}LLL-reduced basis of L
1: Set $k \leftarrow 2$.
2: **while** $k \leq n$ **do**
3: Size-reduce $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_n]$ as well as in Algorithm 1
4: Compute $i \leftarrow \mathrm{argmax}_{1 \leq \ell \leq k-1}(S_{\ell,k})$ for fixed k /∗ See (5) for $S_{\ell,k}$ ∗/
5: **if** $S_{i,k} \leq (1-\eta)\mathrm{SS}(\mathbf{B})$ **then**
6: Set $k \leftarrow k+1$
7: **else**
8: Set $\mathbf{B} \leftarrow \sigma_{i,k}(\mathbf{B})$, and update the GSO of $\mathbf{B}$ /∗ A deep insertion ∗/
9: Set $k \leftarrow \max(i, 2)$ and go back to step 3
10: **end if**
11: **end while**

where we set $\delta_{k-1} := \dfrac{B_k + \mu_{k,k-1}^2 B_{k-1}}{B_{k-1}}$. Since $(\mathbf{b}_{k-1}, \mathbf{b}_k)$ does not satisfy Lovász' condition (1), we have $\delta_{k-1} < \delta$. Then

$$\mathrm{SS}(\mathbf{B}) - \mathrm{SS}(\mathbf{C}) > \frac{\mu_{k,k-1}^2(1-\delta)}{\delta} B_{k-1} \geq \frac{(1-\delta)}{\delta} N.$$

Since $\mathbf{B}$ is η-S^{2}LLL-reduced, we have $\mathrm{SS}(\mathbf{B}) - \mathrm{SS}(\mathbf{C}) \leq (1-\eta)\mathrm{SS}(\mathbf{B})$, and hence $\dfrac{(1-\delta)}{\delta} N < (1-\eta)\mathrm{SS}(\mathbf{B})$. This implies $\delta > M$, which is a contradiction. □

Remark 1 In the above proposition, the factor η is required to be very close to 1 for $M > \frac{1}{4}$. For example, let $\mathbf{b}_1 = (3,1)^\top$ and $\mathbf{b}_2 = (0,1)^\top$. The basis $\mathbf{B} = [\mathbf{b}_1, \mathbf{b}_2]$ is η-S^{2}LLL-reduced for $\eta = \frac{100}{109} \approx 0.917$ since $\mathrm{SS}(\mathbf{B}) = \frac{109}{10}$ and $\mathrm{SS}([\mathbf{b}_2, \mathbf{b}_1]) = 10$. Since $\mu_{2,1} = \frac{1}{10}$ and $N = \frac{1}{10}$, we have $M = \frac{1}{10} < \frac{1}{4}$. However, we see that $\mathbf{B}$ is not δ-LLL-reduced for any $\frac{1}{4} < \delta < 1$.

4.2 S^{2}LLL algorithm

Let $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_n]$ be a basis, and $\mathbf{B}^* = [\mathbf{b}_1^*, \ldots, \mathbf{b}_n^*]$ its GSO with coefficients $\mu_{i,j}$ and $B_j = \|\mathbf{b}_j^*\|^2$. For $1 \leq i < k \leq n$, let $\mathbf{C} = \sigma_{i,k}(\mathbf{B})$ denote the reordered basis. If the case $i = k-1$, the difference $\mathrm{SS}(\mathbf{B}) - \mathrm{SS}(\mathbf{C})$ is given by (4). For general $i < k$, by Theorem 3, the difference is calculated as

$$S_{i,k} := \mathrm{SS}(\mathbf{B}) - \mathrm{SS}(\sigma_{i,k}(\mathbf{B})) = \sum_{j=i}^{k-1} \mu_{k,j}^2 B_j \left(\frac{B_j}{D_j^{(k)}} - 1 \right). \tag{5}$$

Algorithm 2 is our variant of DeepLLL, which we call the *S^{2}LLL algorithm*. In step 4 of Algorithm 2, we can compute the difference $S_{\ell,k}$ from (5) *without computing the GSO of the reordered basis* $\sigma_{\ell,k}(\mathbf{B})$. This makes S^{2}LLL practical. Since deep insertions are performed only when

$$S_{i,k} > (1-\eta)\mathrm{SS}(\mathbf{B}) \Longleftrightarrow \mathrm{SS}(\sigma_{i,k}(\mathbf{B})) < \eta \mathrm{SS}(\mathbf{B}),$$

S^{2}LLL with factor η decreases $\mathrm{SS}(\mathbf{B})$ by a factor of at least η by every deep insertion. Then we obtain the following proposition on the complexity, which guarantees that S^{2}LLL with

$0 < \eta < 1$ runs in polynomial-time (this argument is the same as in [11, Proposition 1] for the complexity analysis of PotLLL).

Proposition 2 *If $0 < \eta < 1$, the number of deep insertions in S^{2}LLL (Algorithm 2) for an input basis $\mathbf{B}$ is at most $O\left(\log_{1/\eta}(\mathrm{SS}(\mathbf{B}))\right)$.*

We remark that both PotLLL and S^{2}LLL have polynomial-time complexity by construction, but their output quality cannot be covered by Theorem 1 since their output bases are no longer DeepLLL-reduced.

Remark 2 Under the randomness assumption in [21], Fukase and Kashiwabara [10] gave an analytic form of the distribution of random sampling vectors over a lattice basis $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_n]$. in particular, the mean μ and the variance σ^2 of the expected distribution are given as

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

This implies that more short lattice vectors could be sampled over a basis with smaller $\mathrm{SS}(\mathbf{B})$, as mentioned in Sect. 1. This is the origin to consider $\mathrm{SS}(\mathbf{B})$ in solving the SVP. In 2017, Aono and Nguyen [2] 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 the randomness assumption, the expectation of the length of a short vector generated by lattice enumeration with discrete pruning from so-called a 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 mean μ. However, it is shown in [2] that 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 [2] is much less efficient than the computation of $\mathrm{SS}(\mathbf{B})$. In 2018, Matsuda et al. [18] investigated the strategy of [10] 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. Hence we focus on $\mathrm{SS}(\mathbf{B})$ to develop a reduction algorithm in this paper.

4.3 Experimental results on S^{2}LLL

In this subsection, we show experimental results to compare DeepLLL, PotLLL and S^{2}LLL. In our experiments, we used NTL library [23] of C++ programs. In our implementation, we used the `int` data type for the lattice basis $\mathbf{B}$, and the `long double` for its GSO information. We also used the `gcc 6.4.0` compiler with option `O3 -std=c++11`. Furthermore, we used a single thread of a 64-bit PC with Intel Xeon CPU E3-1225 v5@3.30GHz and 4.0GB RAM. We took several bases from the Darmstadt SVP challenge [8] of dimensions $n = 100$, 110, 120, 130, 140, and 150 (these bases are random lattices in the sense of Goldstein and Mayer [14]), and reduced them by LLL with reduction parameter $\delta = 0.99$ for input bases. We also used $\delta = 0.99$ for DeepLLL and PotLLL.

4.3.1 Choosing suitable factors of S^{2}LLL

Here we discuss how to set a factor η of S^{2}LLL, corresponding to $\delta = 0.99$ of DeepLLL and PotLLL. Fix a basis $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_n]$ with $B_j = \|\mathbf{b}_j^*\|^2$. From Proposition 1, we roughly expect

$$\delta \approx \frac{N}{(1-\eta)\mathrm{SS}(\mathbf{B}) + N} \Longleftrightarrow 1 - \eta \approx \frac{1-\delta}{\delta} \cdot \frac{N}{\mathrm{SS}(\mathbf{B})},$$

where $N = \min_{1 \le j \le n-1}(\mu_{j+1,j}^2 B_j)$ denotes the same as in Proposition 1 (we expect $N \neq 0$ for all bases in the Darmstadt SVP challenge). In our experiments, input bases $\mathbf{B}$ are LLL-reduced with $\delta = 0.99$. Under the *geometric series assumption (GSA)* [21], we roughly have $B_i / B_{i+1} \approx q^2$ for $1 \le i \le n-1$, where the constant q depends on lattice basis reduction algorithms. According to [20, p. 52 in Chapter 2], we have $q \approx 1.02^2 \approx 1.04$ in practice for random lattices. Then we can roughly obtain

$$1 - \eta \approx \frac{1-\delta}{\delta} \cdot \frac{N}{\mathrm{SS}(\mathbf{B})} \approx \frac{1-\delta}{12\delta} \cdot \frac{q^2-1}{q^{2n}-1}$$

where we simply estimate $N = \mu_{n,n-1}^2 B_{n-1} \approx \frac{B_{n-1}}{12}$ since the expected value of $\mu_{n,n-1}^2$ with $|\mu_{n,n-1}| \le \frac{1}{2}$ is $\frac{1}{12}$. For example, by taking $q = 1.04$ and $n = 100$, we have $1 - \eta = 2.69 \times 10^{-8}$ and we may take $\eta = 1 - 10^{-8}$. Similarly, we obtain $\eta = 1 - 10^{-9}$ and $1 - 10^{-10}$ for $n = 120$ and 140, respectively. However, from the below experimental results, such $1 - \eta$ seems too small, and $\eta = 1 - 10^{-6}$ might be sufficient for $n = 100$–150 since the output quality of S^{2}LLL with $\eta = 1 - 10^{-6}$ is almost equal to that with $\eta = 1$.

4.3.2 Comparison of DeepLLL, PotLLL and S^{2}LLL

In our experiments, we performed DeepLLL with insertion restriction of blocksizes $\varepsilon = 5, 10, 20$ (which we write DeepLLL-5, -10, -20 for short), PotLLL, and S^{2}LLL with factors $\eta = 1, 1 - 10^{-8}, 1 - 10^{-6}$ and $1 - 10^{-4}$. In Figures 1, 2, 3, 4, 5, and 6, we show transition of SS($\mathbf{B}$) in DeepLLL, PotLLL and S^{2}LLL for bases $\mathbf{B}$ of dimensions $n = 100$–150, where every $\mathbf{B}$ is an LLL-reduced basis of the basis generated by the generator in [8] with seed 0. Specifically, in the figures, we show a relation between transition of SS($\mathbf{B}$) and the number of deep insertions required in algorithms. (Data of DeepLLL-20 is not drawn because it requires about 10 times more deep insertions than DeepLLL-10.)

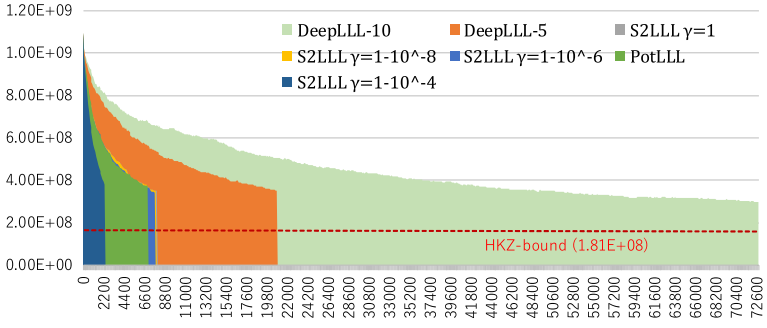

Fig. 1 Transition of SS($\mathbf{B}$) in DeepLLL, S^{2}LLL and PotLLL for an LLL-reduced basis $\mathbf{B}$ of a lattice L of dimension $n = 100$ (x-axis: the number of deep insertions, y-axis: the value of SS($\mathbf{B}$)) (Color figure online)

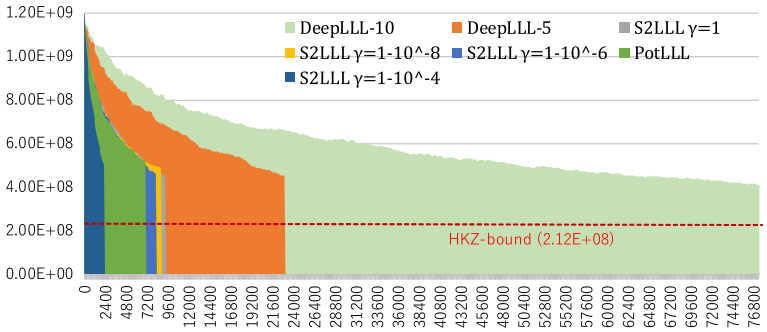

Fig. 2 Same as Table 1, but $n = 110$ (Color figure online)

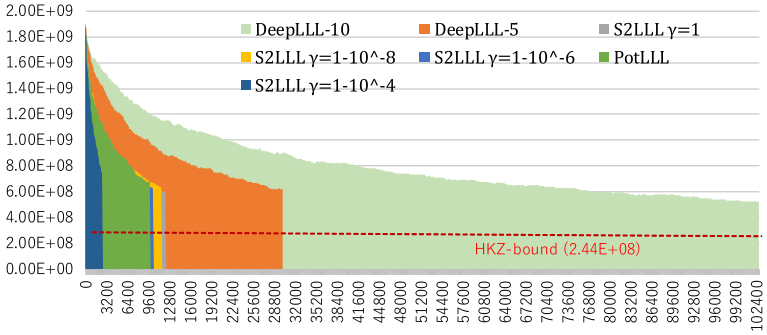

Fig. 3 Same as Table 1, but $n = 120$ (Color figure online)

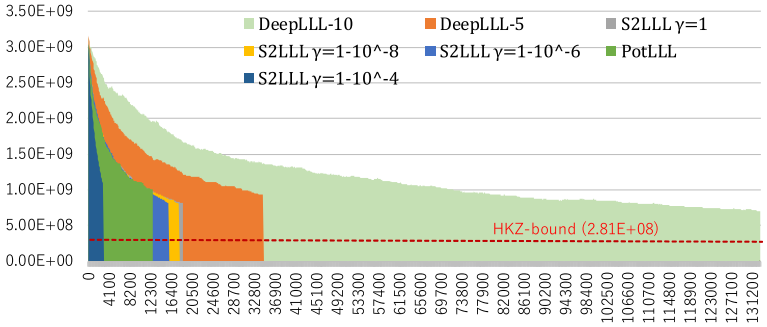

Fig. 4 Same as Table 1, but $n = 130$ (Color figure online)

(a) S^2LLL can decrease SS(**B**) with much less deep insertions than DeepLLL. While S^2LLL decreases SS(**B**) monotonically, the size of SS(**B**) sometimes increases during execution of DeepLLL. More specifically, S^2LLL with factors $1 - 10^{-6} \leq \eta \leq 1$ decreases SS(**B**) by almost the same size as DeepLLL-5, but DeepLLL-5 requires about 2–3 times more deep insertions. Compared to S^2LLL, DeepLLL-10 and -20 output smaller SS(**B**), but they require much more deep insertions (e.g., in $n = 100$, DeepLLL-20 requires 618,162 deep insertions, about 100 times more than S^2LLL).

(b) For factors $1-10^{-6} \leq \eta \leq 1$, S^2LLL outputs almost same size of SS(**B**) with almost same number of deep insertions. In some cases, smaller η (e.g., $\eta = 1 - 10^{-6}$) outputs slightly

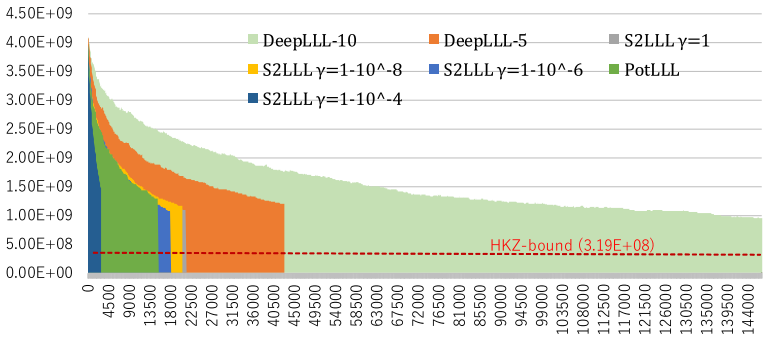

Fig. 5 Same as Table 1, but $n = 140$ (Color figure online)

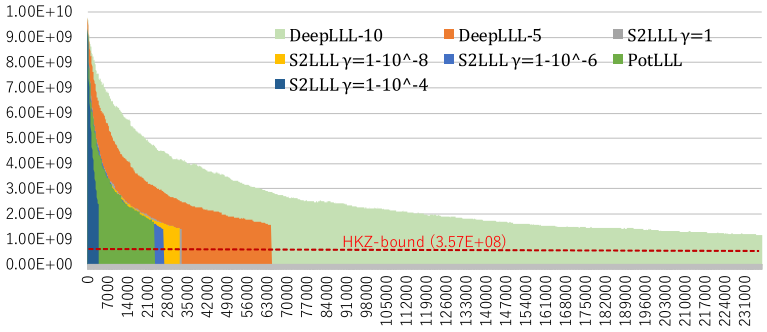

Fig. 6 Same as Table 1, but $n = 150$ (Color figure online)

smaller SS(**B**) than $\eta = 1$. This is due to the *order of deep insertions* during execution; certain sequences of deep insertions enable to decrease SS(**B**) more significantly (e.g., DeepLLL sometimes increases SS(**B**) during execution, but DeepLLL-ε with $\varepsilon \geq 10$ decreases SS(**B**) more than S^{2}LLL in total).

(c) As seen from Figures 1, 2, 3, 4, 5, and 6, the number of deep insertions required in PotLLL is little less than S^{2}LLL with factors $1 - 10^{-6} \leq \eta \leq 1$, but S^{2}LLL decreases SS(**B**) more than PotLLL.

Remark 3 The HKZ-reduction [16] is an ideal reduction for SVP (see also [19, Chapter 7]); A basis $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_n]$ of a lattice L is called *HKZ-reduced* if the following two conditions are satisfied; (i) The basis **B** is size-reduced. (ii) We have $\|\mathbf{b}_i^*\| = \lambda_1(\pi_i(L))$ for all $1 \leq i \leq n$. On the other hand, a random lattice L of dimension n satisfies asymptotically $\lambda_1(L) \approx \sqrt{n/(2\pi e)}\mathrm{vol}(L)^{1/n}$ with overwhelming probability (see [1] for details). For an HKZ-reduced basis $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_n]$, we can roughly estimate every length $\|\mathbf{b}_i^*\|$ as

$$\|\mathbf{b}_i^*\| = \lambda_1(\pi_i(L)) \approx \sqrt{\frac{n-i+1}{2\pi e}}\mathrm{vol}(\pi_i(L))^{1/(n-i+1)}.$$

Since $\mathrm{vol}(\pi_i(L)) = \mathrm{vol}(L)/\prod_{j=1}^{i-1}\|\mathbf{b}_j^*\|$ for $2 \leq i \leq n$, we estimate

$$\mathrm{SS}(\mathbf{B}) \approx \frac{1}{2\pi e}\sum_{i=1}^{n}(n-i+1)\left(\frac{\mathrm{vol}(L)}{\prod_{j=1}^{i-1}\|\mathbf{b}_j^*\|}\right)^{2/(n-i+1)}. \tag{6}$$

Table 1 The average of the Hermite factor constant $\gamma^{1/n}$ of S^{2}LLL with factor η in dimensions $n = 100$–150

n	$\eta = 1$	$\eta = 1 - 10^{-8}$	$\eta = 1 - 10^{-6}$	$\eta = 1 - 10^{-4}$
100	1.01420	1.01408	1.01424	1.01457
110	1.01430	1.01425	1.01422	1.01483
120	1.01418	1.01420	1.01425	1.01494
130	1.01421	1.01418	1.01418	1.01520
140	1.01415	1.01419	1.01412	1.01544
150	1.01419	1.01417	1.01420	1.01568

In Figures 1, 2, 3, 4, 5, and 6, we draw a line of the value (6) as an HKZ-bound of SS($\mathbf{B}$). Compared to the HKZ-bound, the value of SS($\mathbf{B}$) obtained by S^{2}LLL becomes greater for higher dimensions n. This shows that SS($\mathbf{B}$) should be decreased much more for solving SVP or SDP (shortest diagonal problem).

4.3.3 Hermite factor of S^{2}LLL

Let L be a lattice of dimension n. The *Hermite factor* of a lattice basis reduction algorithm for a basis of L is defined by

$$\gamma := \frac{\|\mathbf{b}_1\|}{\mathrm{vol}(L)^{1/n}} \tag{7}$$

with the output basis $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_n]$ (assume that $\mathbf{b}_1$ is shorter than the other $\mathbf{b}_j$). This factor is experimentally investigated in [13], and it is shown that the factor gives a good index to measure the practical output quality of a lattice basis reduction algorithm. The output quality becomes better as γ is smaller. According to experimental results [13, Figure 5], the value $\gamma^{1/n}$ converges a constant in practical algorithms such as LLL, DeepLLL and BKZ for large n. The limiting constant is called the *Hermite factor constant*.

In Table 1, we summarize experimental results of the average of the Hermite factor constant $\gamma^{1/n}$ of S^{2}LLL with several factors η for lattice dimensions $n = 100$–150. In our experiments, for each dimension n, we generated 100 lattice bases by using the generator in the Darmstadt SVP challenge [8] with seeds 0–99. We reduced every basis by LLL with $\delta = 0.99$ in preprocessing, and took it as an input for S^{2}LLL. We see from Table 1 that the average of $\gamma^{1/n}$ of S^{2}LLL is almost equal for factors $1 - 10^{-6} \leq \eta \leq 1$. Moreover, the average with each factor of $1 - 10^{-6} \leq \eta \leq 1$ is about $\gamma^{1/n} = 1.01420$ for $n = 100$–150. In contrast, from experimental results [13, Table 1], the average of the Hermite factor constant of LLL (resp., BKZ-20, BKZ-28, and DeepLLL-50) for random lattices is $\gamma^{1/n} = 1.0219$ (resp., 1.0128, 1.0109, and 1.011). Moreover, from [11, Table 1], the average of the Hermite factor constant of BKZ-5 (resp., PotLLL, DeepLLL-5, DeepLLL-10, and BKZ-10) is $\gamma^{1/n} = 1.0152$ (resp., 1.0146, 1.0137, 1.0129, and 1.0139) in dimension 100 (default parameter of [11] is set as $\delta = 0.99$ for all algorithms). From these results, we estimate that S^{2}LLL with factors $1 - 10^{-6} \leq \eta \leq 1$ has almost the same output quality as BKZ-10, PotLLL and DeepLLL-5 in terms of the Hermite factor. (cf., According to [11, Figure 2], for dimensions $n = 40$–400, DeepLLL-5 and PotLLL have almost same performance, and BKZ-10 is slower.)

4.3.4 Summary on the practical output quality of S^{2}LLL

From our experimental results, S^{2}LLL with factors $1 - 10^{-6} \leq \eta \leq 1$ has almost the same output quality as DeepLLL-5 with $\delta = 0.99$ in terms of both decreasing SS($\mathbf{B}$) and the Hermite factor γ. Different from DeepLLL, S^{2}LLL is proven to run in polynomial-time for $0 < \eta < 1$, and it decreases SS($\mathbf{B}$) with about 2 or 3 times less deep insertions than DeepLLL-5. In fact, S^{2}LLL is much faster than DeepLLL in our experiments. Moreover, since DeepLLL-5 has almost the same implementation performance as the polynomial-time variant PotLLL from [11, Figure 2], S^{2}LLL is more efficient than PotLLL for decreasing SS($\mathbf{B}$) (S^{2}LLL also decreases SS($\mathbf{B}$) more than PotLLL).

4.3.5 Preliminary experiments of S^{2}LLL with random sampling

Here we give preliminary experiments of S^{2}LLL with random sampling for the Darmstadt SVP challenge [8]. For simplicity, we used Schnorr's random sampling [21] in our experiments. Given a basis $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_n]$ of a lattice L and a parameter $1 \leq u < n$ of search space, it samples a short lattice vector $\mathbf{v} = \sum_{i=1}^{n} v_i \mathbf{b}_i^* \in L$ satisfying

$$v_i \in \begin{cases} (-1/2, 1/2] & \text{if } 1 \leq i < n-u, \\ (-1, 1] & \text{if } n-u \leq i < n, \\ \{1\} & \text{if } i = n. \end{cases}$$

Let $S_{u,\mathbf{B}}$ denote the set of lattice vectors $\mathbf{v} = \sum_{i=1}^{n} v_i \mathbf{b}_i^*$ satisfying the above condition. Since the number of candidates for v_i with $|v_i| \leq 1/2$ (resp. $|v_i| \leq 1$) is 1 (resp. 2), there are 2^u short lattice vectors in $S_{u,\mathbf{B}}$.

In Table 2, we show the average and standard deviation of norms of short lattice vectors, sampled by Schnorr's random sampling with $u = 25$ over LLL-reduced and S^{2}LLL-reduced bases of dimensions $n = 100$–150 for the SVP challenge [8] with seed 0. For every reduced basis, we sampled 100,000 different lattice vectors. In the SVP challenge, a vector over an n-dimensional lattice L with norm less than at least

$$1.05 \cdot \frac{\Gamma(n/2+1)^{1/n}}{\sqrt{\pi}} \cdot \mathrm{vol}(L)^{1/n}$$

can be submitted to enter the hall of frame, where Γ is the Gamma function. We write this norm as the target norm in Table 2. We see from Table 2 that an S^{2}LLL-reduced basis is much

Table 2 The average and standard deviation of norms of short lattice vectors, sampled by Schnorr's random sampling (with parameter $u = 25$) over LLL-reduced and S^{2}LLL-reduced bases of dimensions $n = 100$–150 for the Darmstadt SVP challenge [8] with seed 0

n	Target norm of the challenge [8]	LLL ($\delta = 0.99$)		S^{2}LLL ($\eta = 1 - 10^{-8}$)	
		Average	SD	Average	SD
100	2666.50	7892.13	689.99	5558.07	399.69
110	2789.44	10, 349.11	927.48	6302.49	463.38
120	2907.37	12, 613.96	1105.84	7209.72	526.94
130	3030.91	14, 925.14	1311.96	8440.73	645.13
140	3140.67	18, 945.05	1631.50	9444.47	707.72
150	3245.20	24, 317.82	2125.78	10, 898.33	817.56

more suitable than an LLL-reduced basis for sampling shorter lattice vectors. However, it is yet insufficient to achieve the target norm of the SVP challenge. Actually, even 5 times standard deviation around the average of S^2LLL does not reach the target norm, and hence the probability of sampling a lattice vector with norm less than the target norm is extremely low under the assumption that norms of sampled lattice vectors follow the normal distribution. For example, in order to reduce a basis much more, we need to embed the S^2LLL algorithm in BKZ as an alternative subroutine to LLL, like DeepBKZ [25].

5 Conclusion and future work

We showed provable output quality of DeepLLL (Theorem 1), strictly stronger than that of LLL (Theorem 2). We also proposed a polynomial-time variant of DeepLLL (S^2LLL in Sect. 4), which enables us to monotonically decrease the squared-sum $\mathrm{SS}(\mathbf{B})$ of Gram-Schmidt lengths of an input basis $\mathbf{B}$. (Note that provable output quality of S^2LLL cannot be covered by Theorem 1.) Thanks to the explicit GSO formula [25] (Theorem 3), we can keep track of $\mathrm{SS}(\mathbf{B})$ efficiently without computing the Gram-Schmidt vectors exactly. Furthermore, we showed practical output quality of S^2LLL by experiments.

As discussed in the previous subsection, S^2LLL is much more suitable than LLL for sampling shorter lattice vectors. But it is yet insufficient to find a solution of the Darmstadt SVP challenge [8] in dimensions $n \geq 100$. As a future work, we would like to embed S^2LLL in BKZ as an alternative subroutine to LLL, like DeepBKZ [25], in order to reduce an input basis much more for sampling very short lattice vectors. (We might need to modify the enumeration algorithm in BKZ to match with S^2LLL.)

References

1. Ajtai M.: Generating random lattices according to the invariant distribution, Draft of March (2006).
2. Aono Y., Nguyen P.Q.: Random sampling revisited: Lattice enumeration with discrete pruning. In: Advances in Cryptology—EUROCRYPT 2017, Lecture Notes in Computer Science. 10211, pp. 65–102 (2017).
3. Aono Y., Wang Y., Hayashi T., Takagi T.: Improved progressive BKZ algorithms and their precise cost estimation by sharp simulator. In: Advances in Cryptology—EUROCRYPT 2016, Lecture Notes in Computer Science 9665, pp. 789–819 (2016).
4. Babai L.: On Lovász' lattice reduction and the nearest lattice point problem. Combinatorica **6**(1), 1–13 (1986).
5. Bremner M.R.: Lattice Basis Reduction: An Introduction to the LLL Algorithm and Its Applications. CRC Press, Boca Raton (2011).
6. Chen Y., Nguyen P.Q.: BKZ 2.0: better lattice security estimates. In: Advances in Cryptology—ASIACRYPT 2011, Lecture Notes in Computer Science 7073, pp. 1–20 (2011).
7. Cohen H.: A Course in Computational Algebraic Number Theory, vol. 138. Graduate Texts in Math-Springer, Berlin (1993).
8. Darmstadt T.U.: SVP Challenge. http://www.latticechallenge.org/svp-challenge/.
9. Ducas L.: Shortest vector from lattice sieving: a few dimensions for free. In: Advances in Cryptology—EUROCRYPT 2018, Lecture Notes in Computer Science 10820, pp. 125–145 (2018).
10. Fukase M., Kashiwabara K.: An accelerated algorithm for solving SVP based on statistical analysis. J. Inf. Process. **23**(1), 1–15 (2015).
11. Fontein F., Schneider M., Wagner U.: PotLLL: a polynomial time version of LLL with deep insertions. Des. Codes Cryptogr. **73**, 355–368 (2014).
12. Galbraith S.D.: Mathematics of Public Key Cryptography. Cambridge University Press, Cambridge (2012).
13. Gama N., Nguyen P.Q.: Predicting lattice reduction. In: Advances in Cryptology—EUROCRYPT 2008, Lecture Notes in Computer Science 4965, pp. 31–51 (2008).

14. Goldstein D., Mayer A.: On the equidistribution of Hecke points. Forum Math. **15**, 165–189 (2003).
15. Hanrot G., Pujol X., Stehlé D.: Analyzing blockwise lattice algorithms using dynamical systems. In: Advances in Cryptology—CRYPTO 2011, Lecture Notes in Computer Science 6841, pp. 447–464 (2011).
16. Korkine A., Zolotarev G.: Sur les formes quadratiques. Math. Ann. **6**, 366–389 (1873).
17. Lenstra A.K., Lenstra H.W., Lovász L.: Factoring polynomials with rational coefficients. Math. Ann. **261**(4), 515–534 (1982).
18. Matsuda Y., Teruya T., Kashiwabara K.: Estimation of the success probability of random sampling by the Gram–Charlier approximation. IACR ePrint 2018/815 (2018).
19. Micciancio D., Goldwasser S.: Complexity of Lattice Problems: A Cryptographic Perspective. Springer, New York (2012).
20. Nguyen Q., Vallée B.: The LLL Algorithm. Information Security Cryptography (2010).
21. Schnorr C.P.: Lattice reduction by random sampling and birthday methods. In: International Symposium on Theoretical Aspects of Computer Science—STACS 2003, Lecture Notes in Computer Science 2606, pp. 145–156 (2003).
22. Schnorr C.P., Euchner M.: Lattice basis reduction: Improved practical algorithms and solving subset sum problems. Math. Program. **66**, 181–199 (1994).
23. Shoup V.: NTL: a library for doing number theory. http://www.shoup.net/ntl/.
24. Teruya T., Kashiwabara K., Hanaoka G.: Fast lattice basis reduction suitable for massive parallelization and its application to the shortest vector problem. In: Public-Key Cryptography—PKC 2018, Lecture Notes in Computer Science 10769, pp. 437–460 (2018).
25. Yamaguchi J., Yasuda M.: Explicit formula for Gram–Schmidt vectors in LLL with deep insertions and its applications. In: Number-Theoretic Methods in Cryptology—NuTMiC 2017, Lecture Notes in Computer Science 10737, pp. 142–160 (2018).
26. Yasuda M., Yokoyama K., Shimoyama T., Kogure J., Koshiba T.: Analysis of decreasing squared-sum of Gram–Schmidt lengths for short lattice vectors. J. Math. Cryptol. **11**(1), 1–24 (2017).