Research Article

Masaya Yasuda*

Self-dual DeepBKZ for finding short lattice vectors

https://doi.org/10.1515/jmc-2015-0053
Received Feb 05, 2020; accepted Feb 10, 2020

Abstract: In recent years, the block Korkine-Zolotarev (BKZ) and its variants such as BKZ 2.0 have been used as de facto algorithms to estimate the security of a lattice-based cryptosystem. In 2017, DeepBKZ was proposed as a mathematical improvement of BKZ, which calls LLL with deep insertions (DeepLLL) as a subroutine alternative to LLL. DeepBKZ can find a short lattice vector by smaller blocksizes than BKZ. In this paper, we develop a self-dual variant of DeepBKZ, as in the work of Micciancio and Walter for self-dual BKZ. Like DeepBKZ, our self-dual DeepBKZ calls both DeepLLL and its dual variant as main subroutines in order to accelerate to find a very short lattice vector. We also report experimental results of DeepBKZ and our self-dual DeepBKZ for random bases on the Darmstadt SVP challenge.

Keywords: Lattice basis reduction, SVP, BKZ, DeepLLL

2010 Mathematics Subject Classification: Primary: 68R01, Secondary: 06B99

1 Introduction

Since the US National Institute of Standards and Technology (NIST) began a process to develop new standards for post-quantum cryptography (PQC) in 2015 and called for proposals in 2016, it has rapidly accelerated to research lattice-based cryptography as a candidate of PQC. At the submission deadline of November 30, 2017 for the call, NIST received more than 20 proposals of lattice-based cryptosystems (see the web page of "Round 1 Submissions" of [13]). The security of such proposals relies on the hardness of lattice problems such as LWE and NTRU, and it is becoming more important to precisely evaluate the hardness.

Lattice basis reduction is a strong tool in cryptanalysis, and it has been used to estimate the security of lattice-based cryptosystems. In particular, BKZ [16] and its variants such as BKZ 2.0 [3] are de facto algorithms to estimate the security level (see [1]). Given β, BKZ repeatedly calls an SVP oracle in a β-dimensional lattice to find a short lattice vector. In security estimation, it is discussed which blocksizes β are required for BKZ to find a short lattice vector of target norm. A new improvement of BKZ, called DeepBKZ [18], was proposed, which calls DeepLLL [16, Section 3] before every SVP oracle to find a short lattice vector by smaller blocksizes than BKZ. In fact, DeepBKZ with around $\beta = 40$ had found new solutions for the SVP challenge [4] in most dimensions from 102 to 127.

In this paper, we develop a self-dual variant of DeepBKZ, emulating the self-dual BKZ by Micciancio and Walter [11]. The original self-dual BKZ calls SVP and dual SVP oracles in forward and backward tours, respectively, to find a reduced basis $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_n]$ with short $\mathbf{b}_1$ and long $\mathbf{b}_n^\star$, where $[\mathbf{b}_1^\star, \ldots, \mathbf{b}_n^\star]$ denotes the Gram-Schmidt orthogonalization of $\mathbf{B}$. In our self-dual DeepBKZ, DeepLLL and dual DeepLLL [19] are respectively called to reduce $[\mathbf{b}_1, \ldots, \mathbf{b}_{n-1}]$ and $[\pi_2(\mathbf{b}_1), \ldots, \pi_2(\mathbf{b}_n)]$ before every SVP and dual SVP oracles, where π_2 denotes the orthogonal projection. Our construction is similar to [11], but its mathematical background is based on classical proofs of Mordell's inequality as well as the slide reduction algorithm [6] (see also [12]).

***Corresponding Author: Masaya Yasuda:** Institute of Mathematics for Industry, Kyushu University, 744 Motooka, Nishi-ku, Fukuoka 819-0395, Japan; Email: yasuda@imi.kyushu-u.ac.jp

Like DeepBKZ [18], we hope that DeepLLL and its dual variant could help to find a basis with *shorter* $\mathbf{b}_1$ and *longer* $\mathbf{b}_n^\star$ than the original self-dual BKZ [11] (as a result, we expect that a very short lattice vector could be found). To show practicality, we report experimental results of both DeepBKZ and our self-dual variant for random bases on [4].

Notation 1.1. The symbols $\mathbb{Z}$ and $\mathbb{R}$ denote the ring of integers and the field of real numbers, respectively. We represent all vectors in *column* format. For a vector $\mathbf{a} = (a_1, \ldots, a_n)^\top \in \mathbb{R}^n$, let $\|\mathbf{a}\|$ denote its Euclidean norm. For $\mathbf{a} = (a_1, \ldots, a_n)^\top$ and $\mathbf{b} = (b_1, \ldots, b_n)^\top$, let $\langle \mathbf{a}, \mathbf{b} \rangle$ denote the inner product $\sum_{i=1}^n a_i b_i$.

2 Preliminaries

In this section, we review some definitions on lattices. We also introduce typical reduction algorithms and DeepBKZ [18], an improvement of BKZ.

2.1 Lattices

(Primal) lattices and bases

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

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

of dimension n with basis $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_n] \in \mathbb{R}^{n \times n}$. Every lattice has infinitely many bases if $n \ge 2$; If two bases $\mathbf{B}_1$ and $\mathbf{B}_2$ span the same lattice, there exists a unimodular matrix $\mathbf{V} \in \mathrm{GL}_n(\mathbb{Z})$ with $\mathbf{B}_1 = \mathbf{B}_2\mathbf{V}$. The *volume* of L is defined as $\mathrm{vol}(L) = |\det(\mathbf{B})| > 0$, which is independent of the choice of bases. The Gram-Schmidt orthogonalization (GSO) for an *ordered* basis $\mathbf{B}$ is the orthogonal family $\mathbf{B}^\star = [\mathbf{b}_1^\star, \ldots, \mathbf{b}_n^\star]$, recursively defined by $\mathbf{b}_1^\star := \mathbf{b}_1$ and

$$\mathbf{b}_i^\star := \mathbf{b}_i - \sum_{j=1}^{i-1} \mu_{i,j} \mathbf{b}_j^\star \text{ with } \mu_{i,j} := \frac{\langle \mathbf{b}_i, \mathbf{b}_j^\star \rangle}{\|\mathbf{b}_j^\star\|^2}$$

for $2 \le i \le n$. Then $\mathrm{vol}(L) = \prod_{i=1}^n \|\mathbf{b}_i^\star\|$. For $2 \le \ell \le n$, let π_ℓ denote the orthogonal projection over the orthogonal supplement of the $\mathbb{R}$-vector space $\langle \mathbf{b}_1, \ldots, \mathbf{b}_{\ell-1} \rangle_{\mathbb{R}}$ (note that π_ℓ depends on a basis, and set $\pi_1 = \mathrm{id}$). For $1 \le i \le j \le n$, we 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 *first successive minimum* is the length of a shortest non-zero vector in a lattice L, denoted by $\lambda_1(L)$.

Dual lattices and dual bases

The *dual* of a lattice L is defined as

$$\widehat{L} = \left\{ \mathbf{x} \in \mathrm{span}_{\mathbb{R}}(L) : \langle \mathbf{x}, \mathbf{y} \rangle \in \mathbb{Z} \text{ for all } \mathbf{y} \in L \right\},$$

where $\mathrm{span}_{\mathbb{R}}(L)$ denotes the $\mathbb{R}$-vector space spanned by the vectors in L. The dual of a full-rank lattice with basis $\mathbf{B}$ has a basis $\mathbf{D} = \left(\mathbf{B}^{-1}\right)^\top$. In other words, the relation $\mathbf{D}^\top \mathbf{B} = \mathbf{I}$ is always maintained, where $\mathbf{I}$ is the identity matrix. This tells how the dual basis $\mathbf{D}$ changes with respect to changes of the primal basis $\mathbf{B}$.

2.2 Lattice Basis Reduction

Here we introduce some notions of reduction and algorithms to achieve them.

LLL

For a parameter $\frac{1}{4} < \delta < 1$, a basis $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_n]$ is called *δ-LLL-reduced* if it satisfies the two conditions; (i) It is *size-reduced*, namely, the GSO coefficients satisfy $|\mu_{i,j}| \leq \frac{1}{2}$ for all $j < i$. (ii) It satisfies *Lovász' condition*, namely, $\delta\|\mathbf{b}_{k-1}^{\star}\|^2 \leq \|\pi_{k-1}(\mathbf{b}_k)\|^2$ for all k. An LLL-reduced basis can be found by the LLL algorithm [10], in which adjacent basis vectors $\mathbf{b}_{k-1}$ and $\mathbf{b}_k$ are swapped if Lovász' condition does not hold.

DeepLLL

It is a straightforward generalization of LLL, in which *non-adjacent* vectors can be changed; If the *deep exchange condition* $\|\pi_i(\mathbf{b}_k)\|^2 < \delta\|\mathbf{b}_i^{\star}\|^2$ is satisfied for some $i < k$, the vector $\mathbf{b}_k$ is inserted between $\mathbf{b}_{i-1}$ and $\mathbf{b}_i$ as

$$\mathbf{B} \leftarrow [\mathbf{b}_1, \ldots, \mathbf{b}_{i-1}, \mathbf{b}_k, \mathbf{b}_i, \ldots, \mathbf{b}_{k-1}, \mathbf{b}_{k+1}, \ldots, \mathbf{b}_n], \tag{1}$$

This is called a *deep insertion*. Every output basis of DeepLLL satisfies the following condition; For $\frac{1}{4} < \delta < 1$, a basis $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_n]$ is called *δ-DeepLLL-reduced* if it is size-reduced and it also satisfies $\delta\|\mathbf{b}_i^{\star}\|^2 \leq \|\pi_i(\mathbf{b}_k)\|^2$ for all $i < k$.

BKZ

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 $\|\mathbf{b}_i^{\star}\| = \lambda_1(\pi_i(L))$ for every $1 \leq i \leq n$. The notion of BKZ-reduction is a local block version of HKZ-reduction, defined as follows [14–16]; For $\beta \geq 2$, a basis $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_n]$ of a lattice L is called *β-BKZ-reduced* (simply called *β-reduced* in [14, 16]) if it is size-reduced and every local block $\mathbf{B}_{[j,j+\beta-1]}$ is HKZ-reduced for $1 \leq j \leq n-\beta+1$. The second condition means $\|\mathbf{b}_j^{\star}\| = \lambda_1(L_{[j,k]})$ for every $1 \leq j \leq n-1$ with $k = \min(j+\beta-1, n)$. The original BKZ algorithm [16] finds an almost β-BKZ-reduced basis, and it calls LLL to reduce every local block before enumeration of a shortest vector over the block lattice. Efficient variants of BKZ have been proposed such as BKZ 2.0 [3], and some of them have been implemented in software (e.g., [5]).

Self-dual BKZ

Motivated from the slide reduction algorithm [6], an elegant generalization of LLL (see also Section 3.3 below), the self-dual BKZ algorithm was proposed by Micciancio and Walter [11]. Their algorithm is based on a new notion [11, Definition 1] of block reduction using lattice duality (cf., the slide reduction algorithm is based on classical proofs of Mordell's inequality, see also [12] for details). The output quality of self-dual BKZ in the worst case is proven to be at least as the worst case behavior of BKZ [11, Theorem 1]. The self-dual BKZ algorithm calls primal SVP and dual SVP oracles over local blocks in forward and backward tours, respectively, and it calls forward and backward tours alternately. Like BKZ, self-dual BKZ is a proper block generalization of the LLL algorithm.

DeepBKZ

It was proposed in [18], which calls DeepLLL as a subroutine alternative to LLL in BKZ. Every output basis of DeepBKZ satisfies the following condition of reduction; For $\frac{1}{4} < \delta < 1$ and $\beta \geq 2$, a basis is called *(δ, β)-DeepBKZ-reduced* if it is both δ-DeepLLL-reduced and β-BKZ-reduced. In Algorithm 1, we give a detailed algorithm of DeepBKZ with early-abort strategy, adopted in BKZ 2.0; For an input basis $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_n]$ of a lattice L, we terminate DeepBKZ with blocksize β after

$$N = C\left(\frac{n}{\beta}\right)^2 \left(\log n + \log\log \max_{1\leq i\leq n} \frac{\|\mathbf{b}_i^{\star}\|}{\mathrm{vol}(L)^{1/n}}\right) \tag{2}$$

tours, where C is a small constant. Since it is proven in [9] that the output basis of BKZ after N tours has an enough quality, we expect that a similar result would hold for DeepBKZ (we took different values for C in our experiments).

Dual DeepBKZ
It is a dual version of DeepBKZ proposed in [19]. It consists of the dual enumeration in self-dual BKZ [11] and a dual version of DeepLLL to reduce the dual basis of an input basis. In particular, in *dual DeepLLL* [19, Algorithm 1], a basis transformation is performed as

$$\mathbf{B} \leftarrow [\mathbf{b}_1, \ldots, \mathbf{b}_{i-1}, \mathbf{b}_{i+1}, \ldots, \mathbf{b}_k, \mathbf{b}_i, \mathbf{b}_{k+1}, \ldots, \mathbf{b}_n], \tag{3}$$

called a *dual deep insertion* (this is opposite to the primal deep insertion (1)).

3 Self-dual DeepBKZ

In this section, we develop a self-dual variant of DeepBKZ.

3.1 Overview of algorithm

Algorithm 2 is our self-dual DeepBKZ. It consists of two parts of a *forward tour* and a *backward tour*, as in self-dual BKZ by Micciancio and Walter [11]. Let $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_n]$ be an input basis of self-dual DeepBKZ with blocksize $3 \le \beta \le n - 1$. We describe an overview for each part as follows:

Forward tour
As in the original self-dual BKZ [11], we call SVP oracles in dimension β to reduce every local block $\mathbf{B}_{[j,j+\beta-1]}$ from $j = 1$ to $n - \beta$. In our self-dual DeepBKZ, we additionally call DeepLLL for the sub-basis $[\mathbf{b}_1, \ldots, \mathbf{b}_{n-1}]$ before enumeration to find a shortest vector over every block lattice $L_{[j,j+\beta-1]}$. Note that this part does not change the last basis vector $\mathbf{b}_n$.

Backward tour
As in [11], we call dual SVP oracles in dimension β to reduce the dual basis of every local block $\mathbf{B}_{[j-\beta+1,j]}$ from $j = n$ down to $\beta + 1$. Similarly to the above part, we call *dual DeepLLL* [19, Algorithm 1], a dual variant of DeepLLL, before the dual enumeration [11, Algorithm 2] (see also Appendix A for the dual enumeration). As in the above part, we restrict the index i in every dual deep insertion (3) from $i = n - 1$ to 2 in order not to change the first basis vector $\mathbf{b}_1$ by dual DeepLLL. This means that it reduces the local block $\mathbf{B}_{[2,n]}$ by dual DeepLLL, equivalently, it reduces its dual basis by (primal) DeepLLL.

3.2 Terminating condition

A mathematical terminating condition is given in [11, Lemma 1] for the original self-dual BKZ, but it is mentioned in [11] that such condition might never be met in practice. As in [11], we use the early-abort strategy (cf., the termination of the original self-dual BKZ depends on the behavior of GSA slopes). We count a pair of forward and backward tours as one tour of our self-dual DeepBKZ. After

$$M = C \left(\frac{n}{2\beta}\right)^2 \left(\log n + \log\log \max_{1 \le i \le n} \frac{\|\mathbf{b}_i^\star\|}{\mathrm{vol}(L)^{1/n}}\right) \tag{4}$$

tours with the same constant C as in (2), we terminate self-dual DeepBKZ with blocksize β for an input basis $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_n]$ of a lattice L (we replace the denominator of (2) by 2β, see [11, Section 4] for details).

3.3 Mathematical background and our motivation

Inspired by classical proofs of Mordell's inequality, a reduction notion is introduced in [12]; A basis $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_n]$ of a lattice L is called *Mordell-reduced* with factor $\varepsilon \geq 0$ if it satisfies both

$$\|\mathbf{b}_1\| = \lambda_1\left(\mathcal{L}\left(\mathbf{b}_1, \ldots, \mathbf{b}_{n-1}\right)\right) \text{ and } \frac{1}{\|\mathbf{b}_n^\star\|} \leq (1+\varepsilon)\lambda_1(\widehat{\pi_2(L)}).$$

Mordell's reduction satisfies some important properties such as [12, Lemma 10], and the slide reduction algorithm [6] is designed to achieve a blockwise version of Mordell's reduction. In particular, the output quality of the slide reduction algorithm in the worst case is proven to be slightly better than that of BKZ for a fixed blocksize β. (However, from [11, Figure 2], the output quality of the slide reduction algorithm is worse than both BKZ and self-dual BKZ in practice.)

In order to describe the basic idea of our self-dual DeepBKZ, consider

$$\begin{array}{ccccccc} \mathbf{b}_1 & \mathbf{b}_2 & \cdots & \cdots & \mathbf{b}_{n-1} & & \\ & \pi_2(\mathbf{b}_2) & \pi_2(\mathbf{b}_3) & \cdots & \cdots & \pi_2(\mathbf{b}_n). \end{array}$$

for a basis $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_n]$ of a lattice L. Our self-dual DeepBKZ reduces the sub-basis $[\mathbf{b}_1, \ldots, \mathbf{b}_{n-1}]$ by DeepBKZ [18] in forward tours. It also reduces the basis $[\pi_2(\mathbf{b}_2), \ldots, \pi_2(\mathbf{b}_n)]$ of the projected lattice $\pi_2(L)$ by dual DeepBKZ [19] in backward tours, equivalently, it reduces the dual basis by DeepBKZ. With this construction, we expect that our self-dual DeepBKZ could find a basis satisfying a condition close to Mordell's reduction. In particular, like DeepBKZ [18], we hope that our self-dual DeepBKZ could find a *shorter* $\mathbf{b}_1$ (resp., *longer* $\mathbf{b}_n^\star$) with help of DeepLLL (resp., dual DeepLLL [19]) than the original self-dual BKZ [11].

3.4 Implementation

We implemented DeepBKZ (Algorithm 1) and self-dual DeepBKZ (Algorithm 2) in C++ programs with the NTL library [17]. We used the `g++` complier with `-O3 -std=c++11` option. We set a triple of $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_n]$, $\mu = (\mu_{i,j})_{1\leq j<i\leq n}$ and $(B_i)_{1\leq i\leq n}$ as a class, where $\mathbf{B}$ is a basis, μ its GSO coefficients, and $B_i = \|\mathbf{b}_i^\star\|^2$. We used the `int` data type for $\mathbf{B}$, and `long double` for both μ and $(B_i)_{1\leq i\leq n}$. To keep track of the GSO information in DeepLLL and dual DeepLLL, we implemented pseudo-codes of [18, Algorithm 4] and [19, Algorithm 2], respectively. We also implemented [8, Algorithm 2] and Algorithm 3 in Appendix A for primal and dual enumerations, respectively. We took a full enumeration setting as in [18] for both enumerations. Our experiments ran on an Intel Xeon CPU E3-1225 v6@3.30GHz with 32.0 GB RAM (we run every reduction algorithm over a single thread).

3.5 Experimental results

Here we report experimental results of DeepBKZ and self-dual DeepBKZ for random bases on [4] in terms of both the output quality and performance.

Output quality

The *Hermite factor* is a good index to measure the (practical) output quality of a reduction algorithm [7]. The factor of an algorithm for a basis of an n-dimensional lattice L is defined as

$$\gamma = \frac{\|\mathbf{b}\|}{\mathrm{vol}(L)^{1/n}},$$

where $\mathbf{b}$ is a shortest non-zero basis vector output by the algorithm. Smaller γ means that the algorithm can find a shorter lattice vector. For practical algorithms such as LLL and BKZ, Gama and Nguyen [7] experimentally showed that the root Hermite factor $\gamma^{\frac{1}{n}}$ converges a constant for high dimensions. In Table 1 (resp., Table 2), we show the average of the root Hermite factor of DeepBKZ (resp., self-dual DeepBKZ), for random

Table 1: The average of the root Hermite factor $\gamma^{\frac{1}{n}}$ of DeepBKZ with early-abort for some constants C, for the SVP challenge in dimensions n with seeds 0–9 ($C = \infty$ means the average factor *without early-abort*, which data are from [18])

n	C	$\beta = 20$	$\beta = 25$	$\beta = 30$	$\beta = 35$	$\beta = 40$	$\beta = 45$
100	1.0	1.01077	1.01051	1.01039	1.01033	1.01022	1.01008
	4.0	1.01077	1.01035	1.01011	1.01005	1.00994	1.00987
	∞	–	1.01043	1.00999	1.00958	1.00949	–
105	1.0	1.01071	1.01028	1.01022	1.01014	1.01005	1.01003
	4.0	1.01071	1.01023	1.00994	1.00994	1.00991	1.00979
	∞	–	1.01041	1.00983	1.00937	1.00924	–
110	1.0	1.01057	1.01031	1.01023	1.01012	1.01017	1.01004
	4.0	1.01057	1.01024	1.01009	1.01004	1.00998	1.00983
	∞	–	1.01010	1.00975	1.00941	1.00916	–
115	1.0	1.01065	1.01039	1.01027	1.01020	1.01003	1.00997
	4.0	1.01065	1.01032	1.01010	1.00995	1.00993	1.00980
	∞	–	1.01021	1.00964	1.00917	1.00899	–

Table 2: Same as Table 1, but the root Hermite factor of self-dual DeepBKZ

n	C	$\beta = 20$	$\beta = 25$	$\beta = 30$	$\beta = 35$	$\beta = 40$	$\beta = 45$
100	1.0	1.01078	1.01065	1.01060	1.01056	1.01051	1.01049
	4.0	1.01038	1.01035	1.01031	1.01025	1.01024	1.01016
105	1.0	1.01053	1.01053	1.01045	1.01042	1.01036	1.01031
	4.0	1.01033	1.01016	1.01008	1.01008	1.01007	1.01007
110	1.0	1.01060	1.01048	1.01038	1.01036	1.01036	1.01029
	4.0	1.01038	1.01033	1.01030	1.01015	1.01009	1.01002
115	1.0	1.01054	1.01036	1.01036	1.01036	1.01022	1.01017
	4.0	1.01026	1.01021	1.01016	1.01014	1.01013	1.01012

bases on the SVP challenge [4] in dimensions from $n = 100$ to 115 with seeds 0–9. We took $C = 1.0$ and 4.0 for two different early-abort constants in (2) and (4) (cf., $C = 0.25$, 2.0 and 8.0 were taken in [20]). As in [18, 19], we set $\delta = 0.99$ as reduction parameters of both DeepLLL and its dual variant. We took as input bases reduced by BKZ with blocksize 20, implemented in the fplll library [5]. We increased blocksizes β by every 5 from 20 up to 45 for both DeepBKZ and its self-dual variant (it becomes very slow for $\beta \geq 50$ since we did not use any pruning in [8]). We see from Tables 1 and 2 that the root Hermite factor (i.e., the output quality) of DeepBKZ is slightly better than that of self-dual DeepBKZ for $\beta \geq 30$. In Table 1, we also show the average of the root Hermite factor of DeepBKZ *without early-abort*, which data are from [18, Table 2]. As seen from Table 1, the root Hermite factor of DeepBKZ with early-abort is pretty worse than without early-abort for $\beta \geq 30$.

Performance

In Tables 3 (resp., Table 4), we show the average of the running time of DeepBKZ (resp., self-dual DeepBKZ) for the SVP challenge in dimensions from $n = 100$ to 115 with seeds 0–9. From Tables 3 and 4, our self-dual DeepBKZ is at least 3 times slower than DeepBKZ for $\beta \geq 30$. In particular, DeepBKZ is much faster for $20 \leq \beta \leq 30$ due to that a DeepBKZ-reduced basis can be found by tours less than the terminating condition (2). Moreover, DeepBKZ with early-abort is much faster than without early-abort; For example, it took 5242 seconds ≈ about 1.5 hours to run DeepBKZ with early-abort constant $C = 4.0$ in $n = 115$ for blocksizes up to $\beta = 45$, while a few days are required to run DeepBKZ without early-abort.

Table 3: The average of the total running time (seconds) of DeepBKZ with blocksizes β for the SVP challenge in dimensions n with seeds 0–9 (each time was counted from blocksize 20 to given β)

n	C	$\beta = 20$	$\beta = 25$	$\beta = 30$	$\beta = 35$	$\beta = 40$	$\beta = 45$
100	1.0	9	21	52	112	258	439
	4.0	9	43	138	323	750	1330
105	1.0	10	34	71	193	433	801
	4.0	10	53	202	510	1177	2032
110	1.0	27	55	122	298	637	1006
	4.0	28	59	273	803	1985	3529
115	1.0	76	85	173	424	882	1470
	4.0	78	91	437	1329	2916	5242

Table 4: Same as Table 3, but the total running time (seconds) of self-dual DeepBKZ

n	C	$\beta = 20$	$\beta = 25$	$\beta = 30$	$\beta = 35$	$\beta = 40$	$\beta = 45$
100	1.0	231	340	392	484	664	911
	4.0	539	943	1293	1559	2196	3198
105	1.0	315	410	599	785	1104	1714
	4.0	1075	1735	2437	3012	4758	7548
110	1.0	777	1182	1460	1861	2647	4109
	4.0	1151	2795	4443	6043	8335	12154
115	1.0	1254	1817	2565	3295	4622	6606
	4.0	3263	6886	9702	12899	17325	24118

4 Concluding remarks

In this section, we first compare reduction algorithms in terms of both the output quality and performance. We then conclude this work and give our future work.

4.1 Comparison of algorithms

Output quality

A prediction of limiting value of the root Hermite factor $\gamma^{\frac{1}{n}}$ achieved by BKZ is given in Chen's thesis [2] (it is based on Gaussian Heuristic, and it seems to hold for $\beta \geq 40$ in practice);

$$\lim_{n\to\infty} \gamma^{\frac{1}{n}} \approx \left(\frac{\beta}{2\pi e}(\pi\beta)^{\frac{1}{\beta}}\right)^{\frac{1}{2(\beta-1)}}. \tag{5}$$

Actually, as seen from [11, Figure 2], the output quality of both BKZ and self-dual BKZ approximately follows the prediction (5) for $\beta \geq 40$. As discussed in Subsection 3.5, the output quality of our self-dual DeepBKZ is slightly worse than that of DeepBKZ for every $\beta \geq 30$. This is the same as the relation between BKZ and self-dual BKZ, shown in [11]. While the prediction (5) implies that $\beta \geq 85$ is required for BKZ to achieve $\gamma^{\frac{1}{n}} = 1.01$, it requires only $\beta \geq 40$ for both DeepBKZ and our self-dual DeepBKZ with early-abort from Tables 1 and 2 (cf., only $\beta \geq 30$ is required for DeepBKZ without early-abort).

Performance

Since DeepLLL and dual DeepLLL are somewhat costly, DeepBKZ and our self-dual DeepBKZ are more costly than BKZ and self-dual BKZ. However, for blocksizes $\beta \geq 30$, SVP and dual SVP oracles (i.e., enumerations) are

dominant in DeepBKZ and our self-dual DeepBKZ. Hence DeepBKZ and its variants have comparable running time to BKZ and its variants for high blocksizes.

4.2 Conclusion and future work

We showed by experiments that DeepLLL [16] and dual DeepBKZ [19] can accelerate to find a short lattice vector in the framework of self-dual BKZ [11] even for small blocksizes. Furthermore, our self-dual DeepBKZ has comparable running time to self-dual BKZ for high blocksizes since the enumeration cost is dominant for $\beta \geq 30$. However, DeepBKZ [18] is more efficient than our self-dual DeepBKZ in both the output quality and performance. Therefore, as a future work, we would like to improve DeepBKZ with pruning as in BKZ 2.0 [3], and analyze its practical output quality for high blocksizes $\beta \geq 50$.

Acknowledgement: This work was supported by JST CREST Grant Number JPMJCR14D6, Japan. A part of this work was also supported by JSPS KAKENHI Grant Number 16H02830, Japan.

References

[1] Martin R Albrecht, Benjamin R Curtis, Amit Deo, Alex Davidson, Rachel Player, Eamonn W Postlethwaite, Fernando Virdia and Thomas Wunderer, Estimate all the {LWE, NTRU} schemes!, *Cryptology ePrint Archive: Report 2018/331* (2018).

[2] Yuanmi Chen, *Réduction de réseau et sécurité concrete du chiffrement completement homomorphe*, Ph.D. thesis, Paris 7, 2013.

[3] Yuanmi Chen and Phong Q Nguyen, BKZ 2.0: Better lattice security estimates, in: *Advances in Cryptology–ASIACRYPT 2011*, Lecture Notes in Computer Science 7073, Springer, pp. 1–20, 2011.

[4] TU Darmstadt, *SVP Challenge*, Available at https://www.latticechallenge.org/svp-challenge/, 2018.

[5] The FPLLL development team, *fplll, a lattice reduction library*, Available at https://github.com/fplll/fplll, 2016.

[6] Nicolas Gama and Phong Q Nguyen, Finding short lattice vectors within mordell's inequality, in: *Proceedings of the fortieth annual ACM symposium on Theory of computing*, ACM, pp. 207–216, 2008.

[7] Nicolas Gama and Phong Q Nguyen, Predicting lattice reduction, in: *Advances in Cryptology–EUROCRYPT 2008*, Lecture Notes in Computer Science 4965, Springer, pp. 31–51, 2008.

[8] Nicolas Gama, Phong Q Nguyen and Oded Regev, Lattice enumeration using extreme pruning, in: *Advances in Cryptology–EUROCRYPT 2010*, Lecture Notes in Computer Science 6110, Springer, pp. 257–278, 2010.

[9] Guillaume Hanrot, Xavier Pujol and Damien Stehlé, Analyzing blockwise lattice algorithms using dynamical systems, in: *Advances in Cryptology–CRYPTO 2011*, Lecture Notes in Computer Science 6841, Springer, pp. 447–464, 2011.

[10] Arjen Klaas Lenstra, Hendrik Willem Lenstra and László Lovász, Factoring polynomials with rational coefficients, *Mathematische Annalen* **261** (1982), 515–534.

[11] Daniele Micciancio and Michael Walter, Practical, predictable lattice basis reduction, in: *Advances in Cryptology–EUROCRYPT 2016*, Lecture Notes in Computer Science 9665, Springer, pp. 820–849, 2016.

[12] Phong Q Nguyen, *Hermite's constant and lattice algorithms*, The LLL Algorithm, Springer, 2009, pp. 19–69.

[13] The National Institute of Standards and Technology (NIST), *Post-Quantum Cryptography Standardization*, Available at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization.

[14] Claus-Peter Schnorr, A hierarchy of polynomial time lattice basis reduction algorithms, *Theoretical computer science* **53** (1987), 201–224.

[15] Claus-Peter Schnorr, *Block Korkin-Zolotarev bases and successive minima*, International Computer Science Institute, 1992.

[16] Claus-Peter Schnorr and Martin Euchner, Lattice basis reduction: Improved practical algorithms and solving subset sum problems, *Mathematical programming* **66** (1994), 181–199.

[17] Victor Shoup, *NTL: A Library for doing Number Theory*, Available at http://www.shoup.net/ntl/.

[18] Junpei Yamaguchi and Masaya Yasuda, Explicit formula for Gram-Schmidt vectors in LLL with deep insertions and its applications, in: *International Conference on Number-Theoretic Methods in Cryptology–NuTMiC 2017*, Lecture Notes in Computer Science 10737, Springer, pp. 142–160, 2017.

[19] Masaya Yasuda, Junpei Yamaguchi, Michiko Ooka and Satoshi Nakamura, Development of a dual version of DeepBKZ and its application to solving the LWE challenge, in: *Progress in Cryptology–AFRICACRYPT 2018*, Lecture Notes in Computer Science 10831, Springer, pp. 162–182, 2018.

[20] Yang Yu and Léo Ducas, Second order statistical behavior of LLL and BKZ, in: *Selected Areas in Cryptography–SAC 2017*, Lecture Notes in Computer Science 10719, Springer, pp. 3–22, 2017.

A Dual enumeration with modifications

In Algorithm 3, we show an algorithm of dual enumeration by [11, Algorithm 2] with modifications for efficiency. The modifications are based on [8, Appendix B] for primal enumeration. The strategy of dual enumeration is as follows; Let $\mathbf{B} = [\mathbf{b}_1, \dots, \mathbf{b}_n]$ be a basis of a lattice L, and $\mathbf{v}$ a short vector over the dual lattice $\widehat{L}$. By definition of $\widehat{L}$, we have $x_i := \langle \mathbf{v}, \mathbf{b}_i \rangle \in \mathbb{Z}$ for all $1 \le i \le n$, that is, we have $\mathbf{v} = \sum_{i=1}^{n} x_i \mathbf{d}_i \in \widehat{L}$ for the dual basis $\mathbf{D} = [\mathbf{d}_1, \dots, \mathbf{d}_n]$ of $\mathbf{B}$. Since $x_i = \langle \mathbf{v}, \mathbf{b}_i^\star \rangle - \sum_{j=1}^{i-1} \mu_{i,j} \langle \mathbf{v}, \mathbf{b}_j^\star \rangle$ for each $1 \le i \le n$, we can compute $x_i^\star := \langle \mathbf{v}, \mathbf{b}_i^\star \rangle$ from $(x_1, \dots, x_i) \in \mathbb{Z}^i$ as $x_1 = x_1^\star$ and $x_i^\star = x_i + \sum_{j=1}^{i-1} \mu_{i,j} x_j^\star$ for $2 \le i \le n$. On the other hand, for a search bound $A > 0$ with $\|\mathbf{v}\|^2 \le A$, we have

$$\left| x_i + \sum_{j=1}^{i-1} \mu_{i,j} x_j^\star \right|^2 \le \|\mathbf{v}\|^2 \cdot \|\mathbf{b}_i^\star\|^2 \le A \cdot \|\mathbf{b}_i^\star\|^2 \text{ for } 1 \le i \le n.$$

With these equations, we enumerate all coefficient vectors $\mathbf{x} = (x_1, \dots, x_n)^\top \in \mathbb{Z}^n$ of dual lattice vectors $\mathbf{v} \in \widehat{L}$ with $\|\mathbf{v}\|^2 \le A$.

Algorithm 1 DeepBKZ [18] with early-abort

Input: A basis $\mathbf{B} = [\mathbf{b}_1, \dots, \mathbf{b}_n]$ of a lattice L, a reduction parameter $\frac{1}{4} < \delta < 1$, a blocksize $\beta \in \mathbb{Z}$ with $2 \le \beta \le n$, and the maximum number of tours $N > 0$

Output: A (δ, β)-DeepBKZ-reduced basis $\mathbf{B}$ of L (if N is sufficiently large)

1: $\mathbf{B} \leftarrow \texttt{DeepLLL}(\mathbf{B}, \delta)$ /* Compute $\mu := (\mu_{i,j})$ and $B_i := \|\mathbf{b}_i^\star\|^2$ together */
2: $z \leftarrow 0$, $j \leftarrow 0$, $t \leftarrow 0$ /* t is the current number of tours */
3: **while** $z < n - 1$ **do**
4: **if** $j = n - 1$ **then**
5: $j \leftarrow 0$ and $t \leftarrow t + 1$
6: **if** $t \ge N$ **then**
7: **break** /* Early-abort */
8: **end if**
9: **end if**
10: $j \leftarrow j + 1$ /* j mod $n - 1$ */
11: $k \leftarrow \min(j + \beta - 1, n)$ and $h \leftarrow \min(k + 1, n)$
12: $\mathbf{v} = (v_j, \dots, v_k) \leftarrow \texttt{Enum}(\mu_{[j,k]}, (B_i)_{j \le i \le k}, R)$ /* Find $\mathbf{v} \in \mathbb{Z}^{k-j+1}$ such that $\left\| \pi_j \left(\sum_{i=j}^{k} v_i \mathbf{b}_i \right) \right\| = \lambda_1 \left(L_{[j,k]} \right)$ by enumeration for a search bound R (see [8, Algorithm 2], and we took $R = \delta B_j$ for our experiments) */
13: **if** $\mathbf{v} \neq (\pm 1, 0, \dots, 0)$ **then**
14: $z \leftarrow 0$
15: $\mathbf{w} \leftarrow \sum_{i=j}^{k} v_i \mathbf{b}_i \in L$
16: $[\mathbf{b}_1, \dots, \mathbf{b}_h, \mathbf{0}] \leftarrow \texttt{MLLL}([\mathbf{b}_1, \dots, \mathbf{b}_{j-1}, \mathbf{w}, \mathbf{b}_j, \dots, \mathbf{b}_h], \delta)$ /* Remove the linear dependency after insertion of $\mathbf{w}$ at the j-th position */
17: $[\mathbf{b}_1, \dots, \mathbf{b}_h] \leftarrow \texttt{DeepLLL}([\mathbf{b}_1, \dots, \mathbf{b}_h], \delta)$ at stage j
18: **else**
19: $z \leftarrow z + 1$
20: $[\mathbf{b}_1, \dots, \mathbf{b}_h] \leftarrow \texttt{DeepLLL}([\mathbf{b}_1, \dots, \mathbf{b}_h], \delta)$ at stage $h - 1$
21: **end if**
22: **end while**

Algorithm 2 Self-dual DeepBKZ

Input: A basis $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_n]$ of a lattice L, a reduction parameter $\frac{1}{4} < \delta < 1$, a blocksize $3 \le \beta \le n-1$, and the maximum number of tours $M > 0$

Output: A reduced basis $\mathbf{B}$ of L

$t \leftarrow 0$
Compute the GSO information $\mu_{i,j}$ and $B_i = \|\mathbf{b}_i^*\|^2$
while $t \le M$ **do**
  /* Part of a forward tour */
  $[\mathbf{b}_1, \ldots, \mathbf{b}_{n-1}] \leftarrow \texttt{DeepLLL}([\mathbf{b}_1, \ldots, \mathbf{b}_{n-1}], \delta)$
  for $j = 1$ to $n - \beta$ **do**
    $k \leftarrow j + \beta - 1$
    $h \leftarrow \min(k+1, n-1)$
    Same as steps from 8 to 21 in Algorithm 1 for DeepBKZ
  end for
  /* Part of a backward tour */
  $\mathbf{B} \leftarrow \texttt{Dual\_DeepLLL}(\mathbf{B}, \delta)$ at stages from $n-1$ downto 2
  for $j = n$ downto $\beta + 1$ **do**
    $k \leftarrow j - \beta + 1$
    $\mathbf{x} \leftarrow \texttt{Dual\_Enum}(\mu_{[k,j]}, (B_i)_{k \le i \le j}, A)$ /* Enumerate coefficient vectors $\mathbf{x} \in \mathbb{Z}^{j-k+1}$ of short vectors $\mathbf{v} \in \widehat{L}$ (Appendix A) */
    if $\mathbf{x} \neq (0, \ldots, 0, \pm 1)$ **then**
      Insert $\mathbf{v} \in \widehat{L}$ into the dual basis of $\mathbf{B}$ at the j-th position to obtain a new basis $\mathbf{B}$ /* it can be achieved by LLL, see [11, Section 7] */
      $\mathbf{B} \leftarrow \texttt{Dual\_DeepLLL}(\mathbf{B}, \delta)$ at stages from $n-1$ downto 2
    end if
  end for
  $t \leftarrow t + 1$ /* current number of tours */
end while

Algorithm 3 `Dual_Enum`: Dual-Enumeration [11] with modifications

Input: GSO information $(\mu_{i,j})$ and $B_i = \|\mathbf{b}_i^\star\|^2$ of a basis $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_n]$ of a lattice L, and a search bound $A > 0$

Output: The coordinate vector $\mathbf{x} = (x_1, \ldots, x_n)^\top \in \mathbb{Z}^n$ of a dual lattice vector $\mathbf{v} \in \widehat{L}$ satisfying $\|\mathbf{v}\|^2 \le A$ (if such $\mathbf{v}$ exists)

for $k = 1$ to n **do**
  $C_k = 1/B_k$ /* inverse of B_k */
  $\widehat{\mu}_{k,k} = 1$; **for** $j = k+1$ **to** n: $\widehat{\mu}_{k,j} \leftarrow -\sum_{h=k}^{j-1} \mu_{j,h}\widehat{\mu}_{k,h}$;
end for
$\sigma \leftarrow (0)_{(n+1)\times n}$ /* $(n+1) \times n$ matrix with all entries 0 */
$r_1 = 0, r_2 = 1, \ldots, r_n = n-1, r_{n+1} = n$
$\rho_0 = \rho_1 = \cdots = \rho_n = 0$ /* partial norm */
$x_1 = 1, x_2 = \cdots = x_n = 0$ /* current coordinate vector $\mathbf{x}$ */
$c_1 = \cdots = c_n = 0$; $w_1 = \cdots = w_n = 0$ /* centers and jumps */
$k \leftarrow 1$, last_nonzero $\leftarrow 1$ /* largest i for which $x_i \neq 0$ */
while true **do**
  $\rho_k \leftarrow \rho_{k-1} + (x_k - c_k)^2 C_k$
  if $\rho_k \le A$ **then**
    if $k = n$ **then**
      return $\mathbf{x} = (x_1, \ldots, x_n)^\top \in \mathbb{Z}^n$ /* solution found; program ends */
    else
      $k \leftarrow k+1$; $r_{k+1} \leftarrow \min(r_k, r_{k+1})$
      for $i = r_k$ to $k-1$ **do**
        $\sigma_{i+2,k} \leftarrow \sigma_{i+1,k} + x_i\widehat{\mu}_{i,k}$
      end for
      $c_k \leftarrow -\sigma_{k+1,k}$; $x_k \leftarrow \lfloor c_k \rceil$; $w_k \leftarrow 1$
    end if
  else
    $k \leftarrow k-1$
    if $k = 0$ **then return** $\emptyset$ /* there is no solution */
    $r_{k+1} \leftarrow k$
    if $k \ge$ last_nonzero **then**
      last_nonzero $\leftarrow k$; $x_k \leftarrow x_k + 1$
    else
      if $x_k > c_k$ **then** $x_k \leftarrow x_k - w_k$; **else** $x_k \leftarrow x_k + w_k$;
      $w_k \leftarrow w_k + 1$
    end if
  end if
end while