The ISC Int'l Journal of Information Security

ISeCure

January 2025, Volume 17, Number 1 (pp. 75–106)

http://www.isecure-journal.org

Design of an Accurate BKZ Simulation

Gholam Reza Moghissi [1,*] and Ali Payandeh [1]

[1] *Department of ICT, Malek-Ashtar University of Technology, Tehran, Iran*

ARTICLE INFO.

Article history:
Received: January 30, 2024
Revised: September 5, 2024
Accepted: December 30, 2024
Published Online: January 1, 2025

Keywords:
Provable Emulation, Gram-Schmidt Orthogonalization (GSO), Updating GSO Norms, Updating GSO Coefficients, LLL Function, GNR Enumeration

Type: Research Article

doi: 10.22042/isecure.2025.440859.1086

ABSTRACT

The primary role of BKZ simulations focuses on showing the behavior of BKZ algorithm for high block sizes, therefore, current lattice security analysis (e.g., bit-security estimations and selection of efficient/secure parameter-set for current LWE/NTRU-based schemes) needs for these simulations. This paper claims that current BKZ simulations are not necessarily accurate enough for exact lattice security analysis, so for the first time, this study introduces two provable tools of "Emulation of updating GSO norms/coefficients" and "Emulation of LLL function" to be used in designing an accurate BKZ simulation. This paper proves that for a typical SVP solver "Z" (e.g., GNR-enumeration, Sieving, discrete pruning), if there is a simulation of "Z_emulate" which provably emulates the behavior of practical running of "Z", then Our BKZ Simulation by using "emulate_SVPSolver"="Z_emulate" can provably emulate the BKZ algorithm using SVP solver "Z". Our BKZ Simulation solves different problems and weaknesses in former BKZ simulations. Our tests show that, altogether, the shape of GSO norms $\|b_i^*\|^2$, the root-Hermite factor of basis, estimated total-cost and the running-time in "Experimental Running of Original BKZ algorithm" are closer to the corresponding test results in "Our BKZ Simulation" than to the test results in "Chen-Nguyen's BKZ simulation", "BKZ simulation by Shi Bai *et al.*" and some other BKZ models and approximations. Moreover, the wrong strategy in updating GSO norms/coefficients of Chen-Nguyen's BKZ simulation causes many GSO violation errors in lattice blocks, on the other hand, our test results verify that all these errors are eliminated automatically in Our BKZ Simulation.

1 Introduction

The concrete security estimations in lattice-based cryptography has been researched in the long term [1]. Lattice reduction is one of the primary techniques in lattice security attacks. Significant progress has been made to improve the efficiency of lattice-reduction algorithms and to introduce better sense of their behaviors [1]. In this scope, the BKZ algorithm is a main lattice reduction algorithm in security analysis of lattice-based cryptographic primitives [2, 3]. Therefore, the total-cost of running BKZ algorithm and the quality of output basis from it (BKZ) should be predicted accurately in order to be used in bit-security estimation and parameter selection of these cryptographic primitives. The original BKZ algorithm is a simple algorithm, however, it is still poorly under-

* Corresponding author.
Email addresses: fumoghissi@chmail.ir, payandeh@mut.ac.ir
ISSN: 2008-2045

stood from a theoretical point of view [4]. Moreover, since the runtime of "Experimental Running of BKZ algorithm" cannot be tractable for big block sizes, the BKZ simulation is usually used for predicting the behavior of BKZ algorithm in high block sizes.

Related work. There are many former studies on the BKZ simulations that show the determinative role and the importance of them (BKZ simulations) in lattice security analysis and estimating the bit-security of lattice-based cryptographic primitives, such as:

- Chen and Nguyen [2] introduce an efficient BKZ simulation based on Gaussian Heuristic (see Heuristic 1) in 2011. They use their BKZ simulation to predict the approximate required block sizes in BKZ for reaching to some specific target Hermite factors. Also, they illustrate how their BKZ simulation can be used to better security estimates on NTRU Lattices and Gentry-Halevi's main challenges.
- Aono *et al.* [3] introduce a sharp simulator based on Geometric Series Assumption (GSA). Also, they give some simulation results for solving the Darmstadt SVP Challenge;
- The BKZ simulation by Shi Bai *et al.* [5] use more robust facts including the head concavity phenomenon in BKZ and the better sampling enumeration solution norm.
- Shi Bai *et al.* [5] adapt their proposed simulation (see Section 4 in [5]) in a direct way to simulate their proposed algorithm of pressed-BKZ.
- Some lattice-based cryptographic primitives use the BKZ simulations in their bit-security estimations, such as NTRU Prime submission (in Round 1) [6], which uses the Chen-Nguyen's BKZ simulation [2] to determine the necessary block size and number of rounds to achieve a specified root Hermite factor for their BKZ cost model. Also, Hoffstein *et al.* [7] use the Chen-Nguyen's BKZ simulation [2] to determine the block size and the number of rounds of BKZ algorithm required for reaching to specific root Hermite factor in choosing parameters of NTRUEncrypt;
- Postlethwaite *et al.* [8] implement their u-SVP simulator for BKZ by considering some former BKZ simulations [2, 5] in their work, and they try to re-estimate the cost of attacking three lattice-based KEMs in the NIST Post Quantum Standardization Process;
- Wang *et al.* [9] show their practical cost estimations using a progressive-BKZ simulation on the LWE challenge cases.
- Van de Pol *et al.* [10] use Chen-Nguyen's BKZ simulation [2] to identify the best lattice attack that can be applied using the BKZ algorithm for a given dimension at a given security level.
- Dachman *et al.* [11] present a refined strategy using the BKZ simulation and a probabilistic model to have an accurate security prediction even for small block sizes of BKZ algorithm in the security estimation of u-SVP.
- Albrecht *et al.* [12] design an improved lattice-reduction algorithm that achieves the root-Hermite factor of $\beta^{1/(2\beta)}$ in the time of $\beta^{\beta/8+o(\beta)}$ with a polynomial memory, and they use their simulation to attest their claims.
- Albrecht *et al.* [13] make a study on the use of approximate enumeration function in the BKZ algorithm, and they use their simulation to validate their claims and assess the concrete time/quality performance.
- The designers of CRYSTALS use BKZ simulators due to its inaccuracy caused by the "tail" phenomenon in the primal attacks in its official documents of CRYSTALS [14], while in dual lattice attack from [14], the "head" phenomenon of BKZ reduction is the most important.
- Wang *et al.* propose a Pump and Jump BKZ (pnj-BKZ) simulator by using the properties of HKZ reduction basis, based on BKZ 2.0 simulator [15].
- Also, Wang *et al.* optimize the pnj-BKZ simulator [16], then by this optimized pnj-BKZ simulator, they introduce a more accurate hardness estimation of LWE by considering technologies such as progressive-BKZ preprocessing and jump strategy;
- Wenwen *et al.* design a pnj-BKZ simulator, which gives a block size and jump strategy selection algorithm, in order to achieve the best simulated efficiency in solving u-SVP instances [17].
- Ziyu Zhao *et al.* [18] claim that they give several improvements on the BKZ algorithm, which can be used for different SVP-solvers based on enumeration and sieving, while these improvements lead to a speedup of $2^{3\sim4}$. Also, they introduce a simulation for this new BKZ algorithm.
- Zishen Zhao *et al.* [19] introduce a new simulation for predicting the Z-shape of random q-ary lattices by using some results on the distribution of the length of the shortest vector and some reasonable heuristics;
- Etc.

Albrecht *et al.* estimate the cost of primal and dual lattice attacks against LWE-based schemes and primal attacks against NTRU-based schemes by using LWE-estimator from [20] and some proposed cost models for BKZ (see the user-friendly scripts for these estimations in [21]). The cost model is a way to show the total nodes of processing in SVP-solvers (e.g., GNR-enumeration, sieving algorithm and dis-

crete pruning); Definition of an exact formula for the cost model of BKZ algorithm in achieving a specified value of the root-Hermite factor, simplifies the security analysis of lattice-based cryptographic primitives; Wang *et al.* introduce the following four steps in concrete security estimations of lattice-based primitives respectively [1]:

- *First Step*: Specifying lattice problems (e.g., LWE, SIS, NTRU).
- *Second Step*: Parametrising generic attacks.
- *Third Step*: Estimating the required block size β of BKZ.
- *Fourth Step*: Estimating the cost by using the cost model.

Problem. Unfortunately, this paper claims that these cost models are not necessarily exact enough for lattice-based security analysis, since, as shown in paper [22], some typical cryptographic scheme "A" under one cost model may be considered harder (to be broken) than another typical cryptographic scheme "B", in contrast, the scheme "A" under another cost model may appear weaker (to be broken) than the scheme "B". Here, some limited examples from Table 8, 9 and 10 in paper [22], show a surprising gap in estimations of bit-security for a specific parameter set of some cryptographic schemes:

- *Example 1*: The cost of primal attack against "Falcon-1024-2.87-12289" (with claimed bit-security of 230) by using enumeration with four different cost models in Table 10 from [22] is estimated as 2^{418}, 2^{474}, 2^{836} and 2^{1118}.
- *Example 2*: The cost of dual attack against "Titanium.PKE-2048-1.41-1198081" (with claimed bit-security of 256) by using enumeration with four different cost models in Table 10 from [22] is estimated as 2^{595}, 2^{652}, 2^{1096} and 2^{1474}.

These examples lead to two trivial questions: "How the security analysers can use these bit-security estimations to compare the efficiency and the security of different lattice cryptographic primitives;" Also "How the cryptographic designers can choose some efficient/secure parameter-sets for their cryptographic primitives by using these non-exact bit-security estimations."

Solution. "Design of an Accurate BKZ Simulation" mainly can be used to make these cost models and consequently the bit-security estimations of current LWE/NTRU-based schemes (e.g., [23–28]) more precise than before.

Our contributions. Our contributions in this paper are two provable tools of "Emulation of updating GSO norms/coefficients" and "Emulation of LLL function" as two main parts in designing an accurate BKZ simulation. This paper proves that for a typical SVP-solver "Z" (e.g., GNR-enumeration, Sieving and discrete pruning), if there is a simulation of "Z_emulate" which provably emulates the behavior of practical running of "Z", then Our BKZ Simulation (Algorithm 1) by using "emulate_SVPSolver"="Z_emulate" can provably emulate the Experimental Running of BKZ algorithm (Algorithm 6) using SVP-solver of "Z". By using our former contributions and achievements in [29–32] which try to simulate the behavior of GNR-enumeration (as an SVP-Solver in BKZ) with better accuracy, this paper assembles our two contributions (in this paper), together with our former contributions in [29–32] (as our simulation of GNR-enumeration), and introduces a claimant accurate BKZ simulation which is expected to make the estimations of the total cost of lattice attacks more accurate (for reaching to a specified value of root-Hermite factor or a specified degree of basis quality measure). Consequently, using our accurate BKZ simulation can lead to more exact bit-security estimations (and more efficient/secure parameter set) for lattice-based schemes (e.g., [6, 23–26, 33]) against any lattice attacks which use lattice reductions. There are, however, other applications which can be counted for using an accurate BKZ Simulation.

Privileges of our work. The use of our two contributions of "Emulation of updating GSO norms/coefficients" and "Emulation of LLL function" in designing an accurate BKZ simulation, introduce the following privileges over former proposed BKZ simulations:

- Against the former BKZ simulations which use only the "GSO norms" of basis vectors as input/output parameters, our BKZ simulation (in this paper) uses "GSO norms" together with "GSO coefficients" as input/output, and this makes the opportunity of taking more information about the output basis of our BKZ simulation, such as approximation of Euclidean norms of output basis vectors which may be used in further studies for some lattice attacks.
- The Chen-Nguyen's BKZ simulation [2] uses an error-prone strategy for updating GSO norms/coefficients (which is noted in [5] too), but this problem is solved in our BKZ simulation.
- Also, paper [3] does not explicitly introduce a process of updating GSO for its BKZ simulation under Heuristic 2 and Heuristic 3 from [34], instead only analyses its simulation under the assumption of GSA, while (to the best of our knowledge) the assumption of GSA is known as a non-exact approximate prediction only for

middle lattice blocks, however this weakness is solved in our BKZ simulation.
- Moreover, in this paper, it is proved that the process of updating GSO in [5] is not hold under Heuristic 2 and Heuristic 3 from [34] too. Also, this paper claims that the paper of [5] nearly tries to hold only the assumption of GSA to simulate the process of updating GSO norms/coefficients, but this weakness is solved in our BKZ simulation.
- By just using "Emulation of updating GSO norms/coefficients" and ignoring "Emulation of LLL function" after each enumeration over the current lattice block, the correctness of BKZ simulation is not violated, however, this paper shows that the quality of the next block is made worse, so the enumeration cost over the next block increases in an unpleasant way (to the best of our knowledge, the emulation of LLL algorithm is not at all considered in former corresponding studies [2, 3, 5]). This problem is solved in our BKZ simulation.
- Another main reason for using the "Emulation of LLL function" after each enumeration success in the design of our BKZ simulation is that an overflow error in calculations over real numbers can be observed (especially in the case of using float point precision). In contrast, this error is tensed for bigger block sizes (to the best of our knowledge, the emulation of LLL algorithm is not at all considered in papers of [2, 3, 5]). This problem is solved in our BKZ simulation.
- Moreover, Our BKZ Simulation is fully compatible with our former valuable contributions which simulate the behavior of GNR-enumeration (as an SVP-Solver of BKZ) with better accuracy (include: "Revised Estimations for Cost and Success Probability of GNR-Enumeration" [29], "Optimal bounding function for GNR-enumeration" [30], "Better Sampling Method of Enumeration Solution for BKZ Simulation" [31] and "Revised Method for Sampling Coefficient Vector of GNR-enumeration Solution" [32]). However, our BKZ simulation can use other SVP-Solvers of BKZ (instead of GNR-Enumeration).

Our test results. Our test results in this paper verify the value of our contributions in achieving more accuracy of BKZ simulation (and consequently, in better accuracy of lattice bit-security estimations). Briefly, our test results include the following cases:

- Our test results show that altogether the shape of GSO norms of $\|b_i^*\|^2$ in the "Experimental Running of Original BKZ algorithm" is more similar (and close) to the shape of $\|b_i^*\|^2$ in "Our BKZ Simulation", than to the GSO norms of $\|b_i^*\|^2$ in "Chen-Nguyen's BKZ simulation" and "BKZ Simulation by Shi Bai *et al.*".
- Also, our test results show that altogether the root-Hermite factor of basis after the "Experimental Running of Original BKZ algorithm" is nearly close to the root-Hermite factor of basis after applying "Our BKZ Simulation".
- Also, our test results show that altogether the total cost (and the running-time) in the "Experimental Running of Original BKZ algorithm" is nearly more similar (and close) to the total cost (and the running-time) in "Our BKZ Simulation", than to the total cost (and the running-time) in "Chen-Nguyen's BKZ simulation", "BKZ Simulation by Shi Bai *et al.*" and some other BKZ models in our test.
- Moreover, the wrong strategy of updating GSO norms/coefficients in Chen-Nguyen's BKZ simulation results in many GSO violation errors for lattice blocks, while our final test results in this paper verify that all these errors are eliminated automatically in our BKZ simulation.

The remainder of this paper is organized as follows. Section 2 is dedicated to the essential background for understanding our contributions in this paper. Our contributions of "Emulation of updating GSO norms/coefficients" and "Emulation of LLL function" in designing an accurate BKZ simulation (our emulation of BKZ algorithm) would be introduced in Section 3. Our simulation/experimental test results in Section 4 show the value of our contributions to lattice-based cryptography. Finally, the conclusions and the further studies of this work are expressed in Section 5.

2 Preliminaries

In this section, a sufficient background on the lattice theory and the BKZ algorithm is introduced to make this work easy to study. To have the traceability on the relations, propositions and algorithms, the similar notations are used in this paper (such as the notations of n and m for "rank" and "dimension" of lattice in entire the paper).

2.1 Essential Definitions, Notations, and Concepts

In this section, some basic concepts needed in this paper will be defined briefly.

Lattices. For given n-linearly independent vectors $b_1, \ldots, b_n \in \mathbb{R}^m$, a lattice can be generated as the following set of vectors [34]:

$$\mathcal{L}_{[b_1,\ldots,b_n]} = \sum_{i=1}^{n} x_i b_i : x_i \in \mathbb{Z} \quad (1)$$

In other words, a lattice is a set of points in the

n-dimensional space with a periodic structure [35]. The set of vectors $[b_1, \ldots, b_n]$ is known as the basis of lattice which is shown by the column of matrix B in this paper. Also, since the lattices discussed in this paper, are defined for cryptographic applications, this is assumed to consider $b_i \in \mathbb{Z}^m$. The rank and dimension of lattice $\mathcal{L}(B)$ are respectively n and m.

Euclidean Norm. The length of a lattice vector $v = (v_1, \ldots, v_m)$ is measured by $\|v\| = \sqrt[2]{v_1^2 + \cdots + v_m^2}$.

In this paper, the phrases of "norm" and "length" refer to the Euclidean norm.

Fundamental Domain. For a lattice $\mathcal{L}(B)$, the fundamental domain is defined as following set:

$$\mathcal{F}(\mathcal{L}) = t_1 b_1 + t_2 b_2 + \cdots + t_n b_n : 0 \leq t_i < 1 \quad (2)$$

Volume of Lattices. The volume of a lattice $\mathcal{L}(B)$ is defined by the volume of the parallelepiped of fundamental domain $\mathcal{F}(\mathcal{L})$ which is computed as follows:

$$\mathrm{vol}\left(\mathcal{L}(B)\right) = \mathrm{vol}\left(\mathcal{F}(\mathcal{L}(B))\right) = |\det B| \quad (3)$$

There are many hard problems in the lattice theory, where SVP (Shortest Vector Problem) is one of the basic ones. For a given lattice basis B, SVP solvers try to find the shortest nonzero vector in this lattice. In practice, SVP is discussed as an approximate variant, which is defined as follows.

Approximate-SVP (SVP_γ). For a lattice $\mathcal{L}$, the problem of finding a lattice vector whose length is at most some approximation factor γ times the length of the shortest nonzero vector.

Note: The norm of the shortest nonzero vector in lattice $\mathcal{L}$ is shown by $\lambda_1(\mathcal{L})$ (which is first successive-minima in lattice $\mathcal{L}$).

The volume of a n-dimensional sphere (ball) is another basic concept in this paper, which can be computed as follows:

$$\begin{aligned} \mathrm{V}_n(R) &= \mathrm{vol}\left(\mathrm{Ball}_n(R)\right) = (\pi^{n/2}/\Gamma(n/2+1))R^n \\ &\approx ((2\pi e/n)^{n/2}/\sqrt{n\pi})R^n \end{aligned} \quad (4)$$

In this paper, $\mathrm{V}_l(R)$ refers to the volume of a l-dimensional ball with radius R.

For an input basis B with dimension n, one of the main parameters for assessing the quality of the shortest vector returned by SVP-solvers is the "root-Hermite factor", which is defined as follows [36]:

$$\mathrm{RHF}(\mathcal{L}(B)) = \left(\frac{\|b_1\|}{\mathrm{vol}(\mathcal{L}(B))^{1/n}}\right)^{1/n} \quad (5)$$

One of the primary heuristic in lattice theory is Gaussian Heuristic which estimates the number of points in a set S. This heuristic is defined as follows.

Heuristic 1 (Gaussian Heuristic). *Given a lattice $\mathcal{L}$ and a set S, the number of points in $S \bigcap \mathcal{L}$ is approximated by $vol(S)/vol(\mathcal{L})$* [34].

If the lattice $\mathcal{L}$ is limited in a centered ball with radius length of $R = \lambda_1(\mathcal{L})$, then it is expected that there is at least one lattice vector in $\mathrm{Ball}_n(R)$ with radius R, which is the shortest vector and consequently, the value of $\lambda_1(\mathcal{L})$ can be estimated by Gaussian Heuristic of lattice $\mathrm{GH}(\mathcal{L})$ as follows (by using sterling approximation):

$$\mathrm{GH}(\mathcal{L}) = \left(\frac{\mathrm{vol}(\mathcal{L}(B))}{\mathrm{vol}(\mathrm{Ball}_n(1))}\right)^{1/n} \approx \sqrt{\tfrac{n}{2\pi e}}(\det B)^{1/n} \quad (6)$$

Gram-Schmidt Orthogonalization (GSO) and projected sub-lattices are other fundamental definitions in the study of BKZ-reduction, and they are used massively in our contributions of this paper.

Orthogonal projection (π_i). For a given lattice basis $B = (b_1, b_2, \ldots, b_n)$, the orthogonal projection $\pi_i(\cdot)$ is defined as follows:

$$\pi_i : \mathbb{R}^m \mapsto \mathrm{span}(b_1, \ldots, b_{i-1})^\perp, \quad 1 \leq i \leq n \quad (7)$$

Gram-Schmidt Orthogonal basis (GSO basis). For a given lattice basis $B = (b_1, b_2, \ldots, b_n)$, the Gram-Schmidt Orthogonal basis $B^* = (b_1^*, b_2^*, \ldots, b_n^*)$ is defined as follows:

$$\pi_i(b_i) = b_i^* = b_i - \textstyle\sum_{j=1}^{i-1} \mu_{i,j} b_j^*, \quad (8)$$
$$\text{where} \quad \mu_{i,j} = (b_i b_j^*)/\|b_j^*\|^2 \quad \text{and} \quad 1 \leq j < i \leq n$$

Consequently by using Equation (8):

$$\|b_i\|^2 = \textstyle\sum_{j=1}^{i} \mu_{i,j}^2 \|b_j^*\|^2, \quad \textit{where } 1 \leq j < i \leq n \quad (9)$$

Note: An "Orthonormal Basis" in this paper refers to $\|B^*\|'' = [\frac{b_1^*}{\|b_1^*\|}, \ldots, \frac{b_n^*}{\|b_n^*\|}]$.

Note: The parameter of $\|b_x^*\|$ in this paper is named as "GSO norm of basis vector of b_x" or "GSO norm of GSO (projected) vector of b_x^*".

The coefficient matrix μ is organized as the following matrix:

$$\mu = \begin{bmatrix} 1 & 0 & 0 & \ldots \\ \mu_{2,1} & 1 & 0 & \ldots \\ \mu_{3,1} & \mu_{3,2} & 1 & \ldots \\ \vdots & \vdots & \vdots & \ddots \\ \mu_{n,1} & \mu_{n,2} & \mu_{n,3} & \ldots \end{bmatrix} \quad (10)$$

The parameter of $\mu_{i,j} \in R$ is named a Gram-Schmidt coefficient (GSO coefficient), and b_i^* refers to i-th vector of GSO basis of B^*.

For an input lattice basis B, the volume of the lattice can be computed by the norm of GSO vectors as follows:

$$\mathrm{vol}\left(\mathcal{L}(B)\right) = \textstyle\prod_{i=1}^{n} \|b_i^*\| \quad (11)$$

In addition to Heuristic 1 (Gaussian Heuristic), an-

other important heuristic in Lattice theory is Geometric Series Assumption (GSA), which is defined as follows.

Geometric Series Assumption (GSA). Schnorr's GSA says that for a BKZ-reduced basis, the geometric series of $\|b_i^*\| = \mathfrak{r}^{i-1}\|b_1\|$ for the GSA constant $\mathfrak{r} \in [3/4, 1)$ can be assumed [3].

By using GSA assumption, the parameter of q-factor can be defined by Equation (12), which measures the quality of basis [37]:

$$q \approx 1/\mathfrak{r} = \|b_i^*\|/\|b_{i+1}^*\| \tag{12}$$

Moreover, some statistical distributions are used in our analysis, such as Gamma distribution and Exponential distribution. These distributions are defined as follows.

Gamma distribution. The Gamma distribution is a two-parameter and continuous probability distribution which is defined as follows (for input shape-parameter of k and scale-parameter of θ):

$$\text{Gamma}(x; k, \theta) = \frac{x^{k-1}e^{-x/\theta}}{\Gamma(k)\theta^k}, \text{ where } x > 0 \tag{13}$$

The mean and variance in Gamma distribution respectively are determined by $k\theta$ and $k\theta^2$.

Exponential distribution. The Exponential distribution is a one-parameter and continuous probability distribution which is defined as follows (for input parameter of ϑ):

$$\text{Expo}(x; \vartheta) = \vartheta e^{-\vartheta x}, \quad \text{where } x > 0 \text{ and } \vartheta > 0 \tag{14}$$

The mean and variance in Exponential distribution respectively are determined by $1/\vartheta$ and $1/\vartheta^2$.

Note: The notation of multiplication between two scalar parameters of a and a' can be shown by $a \times a'$ or aa'; Also the notation of multiplication between one scalar parameter of a and one vector of y can be shown by $a \times y$ or ay; Moreover the notation of multiplication between two vectors of y and w in this paper (which is known as inner product) is shown by $y \cdot w$.

Note: In this paper, the dimension of lattice basis as "m" is equal to the rank of basis as "n", however, the main parts of our analysis in this paper, which work on the lattice blocks, use the rank of lattice blocks as "d" or "β" with the dimension of $m = n$.

Note: In this paper, the notations of $\text{GH}(\mathcal{L})$, $\text{GH}(\mathcal{L}_\beta)$ and $\text{GH}(\mathcal{L}_{[j,k]})$ are respectively defined as Gaussian Heuristic expectation of λ_1 for an orthogonal projected lattice block $\mathcal{L}$, an orthogonal projected lattice block $\mathcal{L}$ with rank of β and an orthogonal projected lattice block $\mathcal{L} = (b_j^*, b_{j+1}^*, \ldots, b_k^*)$.

Note: The pre-phrases of "full-finished" or "fully-finished" before the functions of LLL or BKZ show that these functions (LLL or BKZ) cannot be aborted and should continue up to the end of their process.

Some other notations in this paper are defined as follows. The absolute value of a real number x is shown by $|x|$. The random function $\text{rand}_{(x,\ldots,y)}$ returns a random real number between x and y (except the numbers of x and y). In fact, the notations of $(x, \ldots, y)$, $(x, \ldots, y]$, $[x, \ldots, y)$ and $[x, \ldots, y]$ respectively represent the range of x to y except x and y, the range of x to y except x, the range of x to y except y and the full range of x to y. Also, the notation of $\lfloor x \rceil$ returns the nearest integer number to x.

2.2 The LLL Algorithm

The most well-known lattice reduction algorithm is LLL function which is developed by Lenstra (Arjen Klaas), Lenstra (Hendrik Willem), and Lovász in 1982 [38]. LLL reduction is a polynomial time algorithm for the problem of approximate-SVP within an approximation factor of $\gamma = 2^{O(n)}$ (where n is the dimension of the lattice basis).

LLL-reduced basis. For a given basis $B = [b_1, \ldots, b_n] \in \mathbb{Z}^{n \times m}$ and the parameter of $\delta \in [\frac{1}{4}, 1)$, LLL-reduced bases should satisfy the following conditions:

- Size-reduction: $|\mu_{i,j}| \leq \frac{1}{2}$ for every $1 \leq j < i \leq n$ (see lines of $2, 3, \ldots, 7$ in Algorithm 3 from Appendix A.1).
- Lovasz criterion: $\|b_{i+1}^*\|^2 \geq (\delta - \mu_{i+1,i}^2)\|b_i^*\|^2$ for $1 \leq i \leq n-1$ (see lines of $8, 9, \ldots, 12$ in Algorithm 3 from Appendix A.1).

The condition of Size-reduction makes the length of vectors smaller. The condition of Lovasz tries to relax the quasi-orthogonality condition (i.e., $\|b_{i+1}^*\| \geq \frac{\sqrt{3}}{2}\|b_i^*\|$), in order to close the vectors into some pleasant degree of orthogonality in polynomial time. The pseudo-code of the LLL algorithm is shown in the Appendix A.1. The first vector of a LLL_δ-reduced-basis B bounded by $\|b_1\| \leq (\frac{2}{\sqrt{4\delta-1}})^{n-1}\lambda_1(\mathcal{L}(B))$. Some useful remarks about the LLL-reduction are stated as follows.

Remark 1. The size-condition does not affect the GSO basis of B^*, therefore when the analysis and discussions only focus on the GSO basis of B^*, this is assumed that the corresponding original basis B is size-reduced.

Remark 2. Lovasz condition geometrically says that whether $\|\pi_i(b_{i+1})\|$ is more than/equal to $\sqrt{\delta} \times \|\pi_i(b_i)\|$ or not.

Remark 3. Based on "Size reduction" and "Lovasz criterion", the LLL algorithm cannot replace the first

GSO norm of basis with a bigger GSO norm (i.e., LLL always decreases or does not modify the first GSO norm).

2.3 The BKZ Algorithm

In 1987, the Blockwise Korkine-Zolotarev (BKZ) algorithm was introduced by Schnorr as an extension of the LLL algorithm. The main idea in BKZ is to replace the blocks of 2×2 (which are used in LLL) with the blocks of larger size. Increasing the block size improves the approximation factor at the expense of more running time. There are several variants of Schnorr's BKZ, but all these variants achieve nearly the same exponential approximation factor. The algorithms of BKZ and Hermite-Korkine-Zolotarev (HKZ) are formally defined as follows:

HKZ-reduced basis. For every block $\mathcal{L}_{[b_j \dots b_n]}$ of input lattice basis $\mathcal{L}_{[b_1 \dots b_n]}$ where $j = 1, \dots, n$, the basis should be size-reduced and satisfies $\pi_j(b_j) = \lambda_1(\pi_j(\mathcal{L}_{[b_j \dots b_n]}))$.

BKZ_β-reduced basis. For every block $\mathcal{L}_{[b_j \dots b_k]}$ of input lattice basis $\mathcal{L}_{[b_1 \dots b_n]}$ where $1 \leq j < k = \min(j + \beta - 1, n)$, then this basis should be size-reduced and satisfies $\pi_j(b_j) = \lambda_1(\pi_j(\mathcal{L}_{[b_j \dots b_k]}))$.

In an algorithmic view, the BKZ algorithm starts with LLL-reduction of basis, then it iteratively performs the following steps:

- For enumeration radius R, the solution vector $v = \sum_{l=j}^{k} y_l b_l$ is returned from SVP-oracle (e.g., lattice enumeration function) which is applied on the projected lattice block of $\pi_j(\mathcal{L}_{[b_j \dots b_k]})$ when $\|\pi_j(v)\| < R$. Next, v is inserted between the vectors of b_{j-1} and b_j. Because of the linear dependency between vectors, the resulting set of vectors is not a basis, so the LLL algorithm is performed on the partial set of $[b_1, \dots, b_{j-1}, v, b_j, \dots, b_{h=\min(k+1,n)}]$.
- Otherwise, the LLL algorithm is performed on the partial set of $[b_1, \dots, b_{h=\min(k+1,n)}]$.

The pseudo-code of Schnorr-Euchner's BKZ algorithm is shown in Algorithm 5 from Appendix A.2; In this paper, some variant of the LLL algorithm which is performed on the partial set of $[b_1, \dots, b_{h=\min(k+1,n)}]$ (see lines of $7, 8, \dots, 13$ in Algorithm 5 from Appendix A.2) is named as "partial-LLL", while against a "full-LLL" (see line 1 in Algorithm 5 from Appendix A.2), the function of partial-LLL starts the reduction process from the stage of j, instead of the stage of 1. The BKZ algorithm can use the lattice enumeration for solving SVP in the projected lattice blocks (however, some other functions, such as sieve algorithm, can be used too) [34]. The first vector of a BKZ_β-reduced basis B bounded by $\|b_1\| \leq (\frac{\beta}{\pi e})^{\frac{n-1}{\beta-1}} \lambda_1(\mathcal{L}(B))$.

Note: Chen's thesis [39] introduces the following approximation for the Root-Hermite factor of basis after running the BKZ algorithm [22]:

$$\text{RHF}(\mathcal{L}(B)) = \left(\frac{\beta}{2\pi e}(\pi\beta)^{1/\beta}\right)^{\frac{1}{2(\beta-1)}} \tag{15}$$

2.4 Enumeration and Pruning Techniques

In this section, the pruning techniques and their corresponding concepts needed in this paper are discussed briefly.

Note: Since lattice blocks are assumed to be used in the BKZ algorithms, the notation $\mathcal{L}(b_j, b_{j+1}, \dots, b_k)$ refers to the projected form of $\pi_j(b_j, b_{j+1}, \dots, b_k)$, as a lattice block from the index j to k, whose vectors are projected on the vectors of $(b_1, b_2, \dots, b_{j-1})$.

Full-enumeration. For initial enumeration radius R, the tree of full-enumeration enumerates all lattice points in n-dimensional ball of radius R.

Full-enumeration Cost. For initial enumeration radius R, the number of total nodes at the level l of the full enumeration tree can be computed as follows [2]:

$$N \approx \sum_{l=1}^{k-j+1=d} H_l, \tag{16}$$

$$\textit{where } H_l = \frac{1}{2}\frac{V_l(R)}{\prod_{i=d-l+1}^{d} \|b_i^*\|} = \frac{1}{2}\frac{R^l V_l(1)}{\prod_{i=d-l+1}^{d} \|b_i^*\|} \tag{17}$$

The notation of H_l represents the Gaussian Heuristic prediction of the number of nodes at the level l (see [2, 34]). The cost of the full-enumeration is given by $T_{node} \times N$, where T_{node} is the average time for processing one node in the enumeration tree, and N is the number of total nodes in this tree [34].

The technique of "GNR-pruning" ("Sound-pruning"), which is introduced by Gama, Nguyen and Regev, uses the main concepts of "Cylinder-intersection" and "Bounding function". Our test results in this paper focus on BKZ-reduction with GNR-pruning.

Cylinder-intersection. The l-dimensional cylinder-intersection with radius $(R_1, \dots, R_l)$ is defined as follows [34]:

$$\mathcal{C}_{R_1,\dots,R_l} = \{(x_1, \dots, x_l) \in \mathbb{R}^l, \forall\ 1 \leq i \leq l, \sum_{t=1}^{i} x_t^2 \leq R_i^2\} \tag{18}$$

Bounding function. The vector of $\mathcal{R} = [\mathcal{R}_1, \mathcal{R}_2, \dots, \mathcal{R}_\beta]$ where $0 \leq \mathcal{R}_1 \leq \mathcal{R}_2 \leq \cdots \leq \mathcal{R}_\beta = 1$, when multiplied by initial radius R, defines a bounded cylinder-intersections with radius $(R_1, \dots, R_l) = (R \times \mathcal{R}_1, \dots, R \times \mathcal{R}_l)$ for $1 \leq l \leq \beta$, and consequently can be used to prune the enumeration tree [34].

GNR-pruning (Sound pruning). For a lattice block of

$B_{[j,k]} = (b_j, b_{j+1}, \ldots, b_k)$ and the coefficient vector $x \in \mathbb{Z}^\beta$, GNR-pruning replaces the inequalities of $\|\pi_{k+1-i}(x \cdot B_{[j,k]})\| \leq R$ for $1 \leq i \leq k-j+1$ as a bounded ball (in full-enumeration) by $\|\pi_{k+1-i}(x \cdot B_{[j,k]})\| \leq \mathcal{R}_i \times R$, where $0 \leq \mathcal{R}_1 \leq \cdots \leq \mathcal{R}_{k-j+1} = 1$ as a cylinder-intersection [2].

The pseudo-code of the GNR pruned enumeration is shown in Appendix B from [34].

Solution Vector. For a lattice block of $B_{[j,k]} = (b_j, b_{j+1}, \ldots, b_k)$ and the coefficient vector $x \in \mathbb{Z}^\beta$, the projected vector of $\pi_j(v) = \pi_j(x \cdot B_{[j,k]})$ where satisfies all of the conditions of $\|\pi_{k+1-i}(x \cdot B_{[j,k]})\| \leq \mathcal{R}_i \times R$ for $1 \leq i \leq k-j+1$, is a final solution vector;

Note: In this paper, $\pi_j(v)$ is shown by the notation of v for simplicity.

One of the main specifications of a bounding function is the success probability which can be defined as follows [34].

The Success Probability of bounding function: For any lattice block of $b_j, b_{j+1}, \ldots, b_k$, initial enumeration radius R and bounding function $\mathcal{R}$, if there is just one lattice vector v in n-dimensional ball with the radius of R (i.e., $\|v\| \leq R$), the probability of finding solution vector v after GNR pruning by bounding function $\mathcal{R}$ in enumeration tree is defined as the success probability of $\mathcal{R}$, which is shown by $\mathrm{p_{succ}}(\mathcal{R})$.

For analysis of the success probability of the GNR bounding function, Gama *et al.* use the following heuristics [34].

Heuristic 2. *The distribution of the coordinates of the target vector v, when written in the normalized Gram-Schmidt basis $(b_1^*/\|b_1^*\|, \ldots, b_n^*/\|b_n^*\|)$ of the input basis, look like those of a uniformly distributed vector of norm $\|v\|$.*

Heuristic 3. *The distribution of the normalized Gram-Schmidt orthogonalization $(b_1^*/\|b_1^*\|, \ldots, b_n^*/\|b_n^*\|)$ of a random reduced basis $(b_1, \ldots, b_n)$ looks like that of a uniformly distributed orthogonal matrix.*

Note: Random reduced basis refers to a basis that is randomized at first, then would be reduced by a reduction algorithm like BKZ.

The success of experiments in paper [2] provides some partial evidences verifying Heuristic 2 and Heuristic 3, however, more experiments are needed to validate these heuristics.

The coefficient vector of $z = (z_1, z_2, \ldots, z_{k-j+1=d})$ in Heuristic 2 which corresponds with the target lattice vector of v, can be formulated as follows (note that, $b_i^*/\|b_i^*\|$ refers to the i-th vector of the orthonormal basis of $b_1^*\|b_1^*\|, \ldots, b_n^*/\|b_n^*\|$) [34]:

$$v = [z_1, \ldots, z_d].\begin{bmatrix} b_k^*/\|b_k^*\| \\ \vdots \\ b_j^*/\|b_j^*\| \end{bmatrix} = [v_1, \ldots, v_m] \quad (19)$$

Note: The concepts of GSO coefficient matrix of μ and GSO coefficient as $\mu_{i,j}$ (which are defined in Section 2.1) are different from the concept of coefficient vectors of z, u, w and y in this section.

The entries of the coefficient vector z are reversed, in the way that z_i corresponds to $b_{k-i+1}^*/\|b_{k-i+1}^*\|$. Also this is clear that $\|z\| = \|v\|$ [34]. Also, the vector of $u = (u_1, u_2, \ldots, u_{k-j+1=d}) = (z_1/R, z_2/R, \ldots, z_d/R)$ is defined to be uniformly distributed in the d-dimensional ball of the radius 1 (by the notation of $u \sim Ball_d$). By using these formulations, the success probability of a GNR bounding function $\mathcal{R}$ can be formally defined as follows [34]:

$$\mathrm{p_{succ}}(\mathcal{R}) = \mathrm{Pr}_{u \sim \mathrm{Ball}_d}\left(\forall i \in [1,d], \sum_{l=1}^{i} u_l^2 \leq \frac{R_i^2}{R_d^2}\right) = \mathrm{Pr}_{u \sim \mathrm{Ball}_d}\left(\forall i \in [1,d], \sum_{l=1}^{i} u_l^2 \leq \mathcal{R}_i^2\right) \quad (20)$$

Note: Since the size of last blocks of BKZ (at each round) becomes less than initial block size of β, so the variable size of $d = k-j+1$ is used in this paper to emphasize this fact.

Note: The dimension of vectors of b_i^* and v is m in this paper (m differs from the rank of lattice block size of d).

Note: The solution vector v from enumeration function over lattice block $\mathcal{L}_{[j,k]}$ is a GSO projected vector which is orthogonal over the previous basis vectors in $\mathcal{L}_{[1,j-1]}$, and remind that the notation of $\mathcal{L}_{[1,d]}$ here represents $\mathcal{L}_{[j,k]}$. Also for vector of X, the projection notation of $\pi_1(X)$ represents $\pi_j(X)$.

The solution vector v can be written by the coefficient vector $w = (z_d/\|b_1^*\|, \ldots, z_2/\|b_{d-1}^*\|, z_1/\|b_d^*\|)$ on the GSO block of $[b_1^*, \ldots, b_d^*]$ as follows [31]:

$$v = [w_1, \ldots, w_d].\begin{bmatrix} b_1^* \\ \vdots \\ b_d^* \end{bmatrix} = [v_1, \ldots, v_m] \quad (21)$$

After inserting the solution vector v at the first of the lattice block $\mathcal{L}_{[1,d]}$ which results in the block of $(v, b_1^*, \ldots, b_d^*)$ with $d+1$ vectors, one of the vectors from the GSO block $(v, b_1^*, \ldots, b_d^*)$ should be eliminated after updating GSO norms of these $d+1$ vectors. The lattice enumeration uses the integer coefficients y_i for enumerating over the projected lattice block $\mathcal{L}_{[1,d]}$, therefore the coefficients w_i in the vector of w depend on integer entries in the vector of $y = [y_1, y_2, \ldots, y_d]$, as follows [32]:

$$v = y \cdot \begin{pmatrix} \pi_1(b_1) \\ \vdots \\ \pi_1(b_d) \end{pmatrix} = y \cdot \begin{pmatrix} b_1^* \\ \vdots \\ b_d^* + \sum_{i=1}^{d-1} \mu_{d,i} b_i^* \end{pmatrix} \quad (22)$$

$$\|v\|^2 = \underbrace{\left(y_1 + \sum_{i=2}^{d} y_i \mu_{i,1}\right)^2}_{z_d^2} \|b_1^*\|^2 + \cdots + \underbrace{\left(y_g + \sum_{i=g+1}^{d} y_i \mu_{i,g}\right)^2}_{z_{d-g+1}^2} \|b_g^*\|^2 + \cdots + \underbrace{y_d^2 \|b_d^*\|^2}_{z_1^2} \Rightarrow (23)$$

$$w = \left[\underbrace{y_1 + \sum_{i=2}^{d} y_i \mu_{i,1}}_{w_1}, \ldots, \underbrace{y_g + \sum_{i=g+1}^{d} y_i \mu_{i,g}}_{w_g}, \ldots, \underbrace{y_d}_{w_d}\right] \quad (24)$$

Following main theorem is introduced in [31].

Theorem 1. *The projected vector $b_g^* \in \{b_1^*, \ldots, b_d^*\}$ which is eliminated after inserting the enumeration solution v, has the GSO norm of $\|b_g^*\| \leq \|v\|$, and the coefficient w_g is always the last non-zero coefficient in vector of w in lattice block of $\mathcal{L}_{[1,d]}$, as follows:*

$$w_g = y_g = 1 \quad (25)$$

Based on Equation (24), a zero coefficient of y_i does not always result in $w_i = 0$, except for indices after the last non-zero coefficient of y_g [31]. The entries of coefficient vectors of w and z can be sampled for full-enumeration as follows [31]:

$$w_i \leftarrow (-1)^{\lfloor \text{rand}_{[0\ldots 2)} \rfloor} \sqrt{\frac{\omega_i(\|v\|^2 - \|b_g^*\|^2)}{\|b_i^*\|^2 \sum_{t=1}^{g-1} \omega_t}}, \quad (26)$$
$$\text{where } \omega_i \leftarrow \text{Gamma}(1/2, 2)$$

$$z_{d-i+1} \leftarrow (-1)^{\lfloor \text{rand}_{[0\ldots 2)} \rfloor} \sqrt{\frac{\omega_i(\|v\|^2 - \|b_g^*\|^2)}{\sum_{t=1}^{g-1} \omega_t}}, \quad (27)$$
$$\text{where } \omega_i \leftarrow \text{Gamma}(1/2, 2)$$

Also, we introduce some approximate techniques for sampling the coefficient vectors of z, u, w, and y in [31] for GNR-enumeration with any success probability of bounding functions, moreover our revised and better techniques for sampling these coefficient vectors can be studied in [32]).

2.5 Solution Norm of GNR Enumeration

Chen and Nguyen performed some experiments showing that for the sufficiently large block size β, the norm of the best solution vector of enumeration can be approximated by $\text{GH}(\mathcal{L}_{[j,k]})$ (see Figure 4 in [2]). Also, Chen and Nguyen perform some experiments to compare the final norm of enumeration solution to $\text{GH}(\mathcal{L}_{[j,k]})$, depending on the starting index j of a local block $\mathcal{L}_{[j,k]}$ for one round of BKZ. At result, for the starting index of j, the final norm (of enumeration solution) is shown significantly lower than $\text{GH}(\mathcal{L}_{[j,k]})$. This behavior of solution norm in running of BKZ is named "head concavity phenomenon" in BKZ, and is discussed in [5]. However, for the last indices (tail of GSO norms), the GSO norms are significantly larger. Corresponding with the name of "head concavity", this last phenomena is named in this paper as "tail convexity". Finally, in the middle of lattice basis, which include the most of the lattice blocks in basis, each enumeration solution is expected to be bounded as follows [2]:

$$0.95 \times \text{GH}(\mathcal{L}_{[j,k]}) \leq \|v\| \leq 1.05 \times \text{GH}(\mathcal{L}_{[j,k]}) \quad (28)$$

To the best of our knowledge, this test in [2] is performed with some block sizes of $\beta \leq 70$. Also, this third behaviour of BKZ, is named as "random manner of middle lattice blocks" in this paper. This experiment may correspond with that behaviour of GSA which is not satisfied precisely in the first and last indexes [40]. The probability distribution of the best solution norm in a lattice basis/block is introduced in Chen's thesis [39] as the following theorem [5].

Theorem 2. *For random lattice $\mathcal{L}_1$ with rank n and unit volume, the distribution of $V_n(1) \times \lambda_1(\mathcal{L}_1)^n$ converges to the distribution of Expo$(1/2)$ for $n \to \infty$.*

Theorem 2 says that the norm of the shortest vector of lattice block of $\mathcal{L}_\beta$ (as the solution norm of full-enumeration over the block of $\mathcal{L}_\beta$) can be sampled as follows [5]:

$$\lambda_1(\mathcal{L}_\beta) = \text{X}^{1/n} \text{GH}(\mathcal{L}_\beta), \text{ where } \text{X} \leftarrow \text{Expo}(\tfrac{1}{2}) \quad (29)$$

Note: Theorem 2 and formula Equation (29), which are used for configuring Algorithm 2 in our test of Section 4, only would be considered for full-enumeration (i.e., the GNR-enumeration function including a bounding function with success probability 1), not for any pruned enumeration.

Note: Our proposed technique in Lemma 3 from [31] defines a general method for sampling the solution norm of GNR-enumeration with any success probability of bounding functions. At the same time, Theorem 2 is entirely consistent with this technique (Lemma 3 from [31]).

2.6 Enumeration Radius

The enumeration radius R in paper [2] is defined as follows by some partial modification [29]:

$$\|R\| = \begin{cases} \min(\sqrt{\Upsilon}\text{GH}(\mathcal{L}_{[j,k]}), \|b_j^*\|) & \text{if } k - j + 1 \geq 30 \\ \|b_j^*\| & \text{otherwise} \end{cases} \quad (30)$$

where $\sqrt{\Upsilon}$ is the initial radius parameter. For block size of $k-j>30$, the parameter of r_{FAC} is defined as follows [29]:

$$r_{\text{FAC}} = \frac{\min\left(\sqrt{\Upsilon}\text{GH}(\mathcal{L}_{[j,k]}), \|b_j^*\|\right)}{\text{GH}(\mathcal{L}_{[j,k]})} \tag{31}$$

The main problem in choosing the enumeration radius is to find the smallest radius which is not smaller than the shortest vector in the input lattice block [29]. For this end, Chen and Nguyen claim that the radius parameter of Υ in practice can be selected as $\sqrt{\Upsilon} = \sqrt{1.1} \approx 1.05$ (see Figure 3 in [2]). By using Theorem 2, the optimal enumeration radius can be defined from the concept of full-enumeration success probability, which is defined in [29]. A full-enumeration with initial radius R intrinsically prunes enumeration by using enumeration radius (i.e., using an enumeration radius is concretely a type of pruning) [29]. As mentioned, former estimation of enumeration radius in Equation (28) uses experimental tests to estimate an accurate bound for $\frac{\lambda_1(\mathcal{L}_\beta)}{\text{GH}(\mathcal{L}_\beta)}$ in average-case (see Figure 3 in [2]), to be used for selecting r_{FAC}. Theorem 3 which is introduced by us in [29], introduces an exact definition of this bound.

Theorem 3. *For given number X of random lattice blocks, the tight bound of $\frac{\lambda_1(\mathcal{L}_\beta)}{GH(\mathcal{L}_\beta)}$, as the effective bound of r_{FAC}, is estimated in average-case, as follows:*

$$\sqrt[\beta]{-2\ln\left(1-\text{p}_{\text{MIN}}\right)} \leq r_{\text{FAC}} = \frac{\lambda_1(\mathcal{L}_\beta)}{\text{GH}(\mathcal{L}_\beta)} \leq \sqrt[\beta]{-2\ln\left(1-\text{p}_{\text{OPT}}\right)},$$
$$\textit{where } \text{p}_{\text{MIN}} = 1/\text{X} \textit{ and } \text{p}_{\text{OPT}} = 1-\varepsilon. \tag{32}$$

One of the main parameters introduced based on Theorem 3 is "Minimum hopeful radius parameter of $r_{\text{FAC}_{\text{min}}}$" which is defined as follows [29].

Minimum hopeful radius parameter ($r_{FAC_{min}}$). For given number X of random lattice blocks, the minimum radius parameter leads to success probability of $\text{p}_{\text{MIN}} = \text{p}_{\text{succ}}(1^\beta, R, \mathcal{L}_\beta) = 1/X$ for full-enumeration over these X blocks where $R = r_{\text{FAC}_{\text{min}}} \times \text{GH}(\mathcal{L}_\beta)$ (i.e., only one of the full-enumerations over these X blocks probably returns the best solution).

Remark 4. The random manner of lattice blocks $\mathcal{L}_{[j,k]}$ in the BKZ algorithm would be observed only for Hdown $\leq j \leq$ Tup where "Hdown" represents the maximum index in head concavity and "Tup" represents the minimum index in tail convexity, so for each round of BKZ algorithm (or BKZ simulation), the number X of random lattice blocks can be assumed as $X = \text{Tup} - \text{Hdown} + 1$ [29].

3 Our Contributions

This paper claims that former BKZ Simulations are not necessarily accurate enough for exact lattice security analysis, so for the first time, this study introduces two provable tools of "Emulation of updating GSO norms/coefficients" and "Emulation of LLL function" to be used in designing an accurate BKZ Simulation. This section proves that for a typical SVP solver "Z" (e.g., GNR-enumeration, Sieving, discrete pruning), if there is a simulation of "Z_emulate" which provably emulates the behavior of practical running of "Z", then Our BKZ Simulation (Algorithm 1) by using "emulate_SVPSolver"="Z_emulate" can provably emulate the Experimental Running of BKZ algorithm (Algorithm 6) using SVP solver "Z". Using the phrase of "emulation" instead of "simulation" for functions of updating GSO norms/coefficients and LLL function, refers to this fact that, "Emulation of updating GSO norms/coefficients" and "Emulation of LLL function" exactly represent the operations of "Updating GSO norms/coefficients" and "LLL function" in the BKZ algorithm. Definition 1 declares our formal definition of "emulate" (and "emulation") in this paper.

Note: A function in mathematics with domain "A" and codomain "B" is a binary relation of "$B = \text{F}(A)$", while for every $(a_1, b_1) \in \text{F}$ and $(a_2, b_2) \in \text{F}$, if $a_1 = a_2$ then $b_1 = b_2$.

Definition 1. For typical functions of "AlgorithmT" and "Emulate_AlgorithmT" in mathematics, if "Y = Emulate_AlgorithmT(X)" emulates "y = AlgorithmT(x)", then there are two functions of "f" and "g" with conditions of "$X = f(x)$" and "$Y = g(y)$".

Based on Definition 1 in the scope of this research, x can be assumed as input basis B to BKZ, and y can be assumed as output basis B' reduced by BKZ. In other hand, in the scope of this research, X can be assumed as GSO norms $\ell_i = \log\left(\|b_i^*\|\right)$ together with GSO Coefficient Matrix μ of input basis B to BKZ, while Y can be assumed as GSO norms $\ell'_i = \log\left(\|b'^*_i\|\right)$ together with GSO Coefficient Matrix μ' of output basis B' reduced by BKZ.

Note: The general concept of "simulation" is not the same exact as the concept of "emulation" in Definition 1.

Section 3.1 introduces our contribution of "Emulation of updating GSO norms/coefficients" (and corresponding discussions). Section 3.2 introduces our contribution of "Emulation of LLL function" (and corresponding discussions). Finally, our emulation of the BKZ algorithm ("Our Accurate BKZ Simulation") is proposed in Section 3.3.

3.1 Emulation of Updating GSO Norms/Coefficients

After inserting the solution vector v at the first of lattice block $\mathcal{L}_{[1,d]}$, the LLL function and the process

of updating GSO should be simulated. To the best of our knowledge, the other BKZ simulations try to leave the process of updating GSO in BKZ simulation by non-exact approximated approaches. For example, the Chen-Nguyen BKZ simulation uses an error-prone strategy, also as shown in this section, the simulations in paper [3, 5] nearly tries to hold only GSA assumption (see Lemma 3). However, why is it essential to design an exact emulation of LLL and updating GSO? Also, how does it work? For the first question, the motivation for designing this component is to make better the cost estimation in BKZ reduction. To have a better outlook on this motivation, see Remark 5.

Remark 5. Suppose in the first round of the BKZ algorithm, at first the full-LLL algorithm is applied (see line 1 in Algorithm 5 from Appendix A.2) in order to satisfaction of Lovasz-criteria and size condition. Next, when the first lattice block of the BKZ algorithm begins to be enumerated, all proposed simulations nearly can estimate the cost in the same manner which results in nearly similar cost value. For the next lattice blocks of $\mathcal{L}_{[2,\beta+1]}$ to $\mathcal{L}_{[n-\beta+1,n]}$, after each enumeration success, the Experimental Running of BKZ algorithm performs the partial-LLL (see Algorithm 4 from Appendix A.1), and then updates the GSO norms. Assume that the block size is sufficiently big, therefore, the small changes in the quality of the blocks may lead to non-negligible changes in estimating the cost of enumeration. Also, this paper emphasizes that the exact measurements of quality of current block after an enumeration success cannot be estimated by using the assumption of GSA in simulation of updating GSO norms.

For second question, by adding the solution vector v at the first of lattice block $\mathcal{L}_{[1,d]}$, the new GSO norm for each vector b_i^* would be computed in Lemma 1 as follows.

Lemma 1. *For an input lattice block $\mathcal{L}_{[1,d]}$, after inserting the enumeration solution v at the first of this block, the GSO norms of this block can be updated by relation Equation (33):*

$$\begin{aligned}\|b^*_{i\,new}\| &= \|b^*_{i-1}\| \times \sqrt{1 - \frac{w_{i-1}^2 \|b^*_{i-1}\|^2}{\|\pi_{i-1}(v)\|^2}} \\ &= \|b^*_{i-1}\| \times \sqrt{1 - \frac{z_{d-i+2}^2}{\|\pi_{i-1}(v)\|^2}}, \qquad (33)\end{aligned}$$

where $2 \le i \le d+1$ *and* $\|b^*_{1\,new}\| = \|v\|$

See proof in Appendix B.1;

In relation Equation (33), the notation of $\|b^*_{i\,new}\|$ represents the GSO norm of vector i in newly updated lattice block, while by using formula Equation (33), the value of $\|b^*_{g+1\,new}\|$ becomes 0, since this vector is a dependent vector to the previous vectors of $b^*_{1\,new}, \ldots, b^*_{g\,new}$. In addition to GSO norms, the GSO coefficient matrix μ is among the main parameters for emulation of LLL, so this matrix should be updated too. For an input basis, the initial GSO coefficient matrix μ can be computed simply by applying the "compute_GSO" function or using some version of the LLL function which internally computes the matrix of μ. Besides Lemma 1, the emulation of updating GSO to update the coefficient matrix of μ, is formulated in Lemma 2 (as mentioned, in this paper, the notation v is referred to $\pi_j(v)$).

Lemma 2. *For an input lattice block $\mathcal{L}_{[j,k]}$, after inserting the enumeration solution vector v at the first of the block, the process of updating the GSO coefficient matrix μ into coefficient matrix μ'' can be defined as follows:*

1: $\mu''_{\ell,l} = \mu_{\ell,l},\ for\ 1 \le l < \ell < j$

2: $\mu''_{\ell,l} = \sum_{i=1}^{g} y_i \mu_{j+i-1,l},\ for\ 1 \le l < j = \ell$

3: $\mu''_{\ell,l} = \mu_{\ell-1,l},\ for\ 1 \le l < j < \ell \le j+g-1$

4: $\mu''_{\ell,l} = \mu_{\ell,l},\ for\ 1 \le l < j\ and\ j+g \le \ell$

5: $\mu''_{\ell,l} = \frac{\sum_{t=j}^{\ell-1} \mu_{\ell-1,t} w_{t-j+1} \|b_t^*\|^2}{\|\pi_j(v)\|^2},\ for\ j = l < \ell \le j+g-1$

6: $\mu''_{\ell,l} = \frac{\sum_{t=j}^{j+g-1} \mu_{\ell,t} w_{t-j+1} \|b_t^*\|^2}{\|\pi_j(v)\|^2},\ for\ j = l < j+g \le \ell$

7: $\mu''_{\ell,l} = \mu_{\ell-1,l-1} - \frac{w_{l-j} \times \left(\sum_{t=l}^{\ell-1} \mu_{\ell-1,t} w_{t-j+1} \|b_t^*\|^2\right)}{\|\pi_l(v)\|^2},$
$for\ j < l < \ell \le j+g-1$

8: $\mu''_{\ell,l} = \mu_{\ell,l-1} - \frac{w_{l-j} \times \left(\sum_{t=l}^{j+g-1} \mu_{\ell,t} w_{t-j+1} \|b_t^*\|^2\right)}{\|\pi_l(v)\|^2},$
$for\ j < l < j+g \le \ell$

9: $\mu''_{\ell,l} = \mu_{\ell,l},\ for\ j+g \le l < \ell$

10: $\mu''_{\ell,l} = 1,\ for\ l = \ell$

11: $\mu''_{\ell,l} = 0, for \ell < l$

See proof in Appendix B.2; Also Algorithm 7 in Appendix A.3 shows the pseudo-code of our method for emulation of updating GSO in our BKZ simulation.

As mentioned, the simulation of updating GSO in paper [3] follows the assumption of GSA. Also, by the study of the simulation of updating GSO in Algorithm 4 from paper [5] deeply, this would be found that, although the determinant of current lattice blocks is preserved after updating GSO in simulation algorithm of paper [5], it does not work strictly under Heuristic 2 and Heuristic 3 (see Lemma 3). This paper claims that the simulation in paper [5] nearly tries to hold only the GSA assumption to simulate the update-GSO and the partial-LLL. Although this method in [5] may simulate the process of update-GSO plus partial-LLL approximately, we feel that this is not exact enough for accurate estimation of cost of BKZ

for sufficiently big block sizes.

Lemma 3. *The simulation of updating GSO in Shi Bai* et al.*'s BKZ simulation [5] after the full-enumeration over the lattice block $\mathcal{L}_{[1,d]}$, which operates as follows, is not held under Heuristic 2 and Heuristic 3.*

$$\|b^{*'}_{1\,new}\| \leftarrow \|v\| \tag{34}$$

$$\|b^{*'}_{2\,new}\| \leftarrow \|b^*_1\|\sqrt{1-1/d} \tag{35}$$

$$\|b^{*'}_{i\,new}\| \leftarrow \|b^*_i\| \times \sqrt[\beta-2]{\frac{\|b^*_2\|}{\|v\|\sqrt{1-1/d}}}, \quad for\ 3 \leq i \leq d \tag{36}$$

See proof in Appendix B.3;

Chen and Nguyen believe that since the LLL-reduction in their BKZ simulation [2] after a successful enumeration usually changes the values of the subsequent GSO norms of $\|b^*_i\|$, no comparison needs and simulation can replace the first GSO-norm in all subsequent local blocks of $\mathcal{L}_{[j,k]}$ with value of $\text{GH}(\mathcal{L}_{[j,k]})$. Note that if this strategy of Chen-Nguyen's simulation is ignored (by eliminating the lines 6, 11, 14, and 17 in Algorithm 4 from paper [2]), the GSO output of the simulation is not be modified, unless this simulation faces with the blocks of $\mathcal{L}_{[j,k]}$ which have the condition of $\|b^*_j\| > \text{GH}(\mathcal{L}_{[j,k]})$. As mentioned, many lattice blocks in the middle of a basis have random manner, so by using the proposed bound of Equation (32) in Theorem 3, the first GSO-norm of these middle blocks should be more than $r_{\text{FAC}_{\min}}$. Surprisingly, by our simulation results in Section 4.3, the condition of $\|b^*_j\| < r_{\text{FAC}_{\min}} \times \text{GH}(\mathcal{L}_{[j,k]})$ can be observed many times in the rounds of Chen-Nguyen's simulation. This phenomena is named as "GSO violation error" in this paper. This error returns to the non-suitable updating of GSO norms after each enumeration.

The process of updating GSO norms after each successful enumeration is defined to refresh/re-compute the GSO norms and GSO coefficients for all basis vectors, while this updating (re-computing GSO norms/coefficients) is not observed in the Chen-Nguyen's BKZ simulation. The Chen-Nguyen's simulation by setting $\|b^*_j\|$ to $\text{GH}(\mathcal{L}_{[j,k]})$ in the conditions of $\|b^*_j\| < \text{GH}(\mathcal{L}_{[j,k]})$ tries to solve this problem (GSO violation error), but this is clear that this strategy is wrong, since this strategy is applied on the partially corrupted projected lattice blocks. Since Chen-Nguyen's BKZ simulator must compute the GSO norms sequentially, these accumulative errors make the output of BKZ simulation more far from true expected output, as the rounds of BKZ simulation are increased. Note that our simulation never show any GSO violation errors in the test results (see Section 4.4), because the use of the emulation of LLL and updating GSO, together with precisely sampling of solution norm and coefficient vectors of w and z.

3.2 Emulation of LLL function

Lemma 4 makes better sense of how LLL function affects the GSO norms of an input basis.

Lemma 4. *For GSO basis $[b^*_1, \ldots, b^*_n]$, the function of $LLL_{\delta \approx 1}$ is equivalent to the fully-finished function of BKZ_2 with full-enumeration and, after satisfaction of Lovasz condition over $[b^*_i, b^*_{i+1}]$, only exchanges the locations of b_i and b_{i+1}, and modifies their GSO norms into Equation (37), while other GSO norms would not be modified:*

$$[\|\pi_i(b_{i+1})\|, \|\pi_{i+1}(b_i)\|] = \left[\sqrt{\|b^*_{i+1}\|^2 + \mu^2_{i+1,i}\|b^*_i\|^2}, \frac{\|b^*_i\| \times \|b^*_{i+1}\|}{\sqrt{\|b^*_{i+1}\|^2 + \mu^2_{i+1,i}\|b^*_i\|^2}}\right] \tag{37}$$

See proof in Appendix B.4; By using Lemma 4, following corollaries can be introduced.

Corollary 1. The quality of $LLL_{\delta \leq 1}$-reduced GSO basis $b^*_1, \ldots, b^*_n$ is similar or worse than the quality of full-finished BKZ_2, so that against BKZ with a block size of $\beta > 2$, the enumeration success (in BKZ) can generate a new vector v by linear combination of projected lattice block vectors, the function of LLL just sort the vectors in a two dimensional lattice block under Lovasz-condition.

Corollary 2. A full-finished BKZ_β with full enumeration is at least LLL_δ-reduction over basis $b_1, b_2, \ldots, b_n$, since a BKZ_β with full enumeration would be at least a full-finished BKZ_2 with full enumeration, therefore the BKZ simulations with full-enumeration and high rounds doesn't need to LLL_δ-reduction.

Corollary 3. The function of LLL_δ-reduction over basis $b_1, b_2, \ldots, b_n$ dose not necessarily sort the original Euclidean norm of vectors, but just tries to sort each two projected vectors of $\pi_j(b_j)$ and $\pi_j(b_{j+1})$.

Corresponding to Corollary 3, this fact can be extended for block sizes $\beta > 2$ by Lemma 5 as follows:

Lemma 5. *For lattice block $\mathcal{L}_{[j,k]}$, a GNR enumeration with bounding function $\mathcal{R}$ and enumeration radius $R = \|b^*_j\|$ which returns the solution vector v with norm of $\|v\| \leq \min(\|\pi_j(b_j)\|, \ldots, \|\pi_j(b_k)\|)$, can be defined as follows:*

$$\mathcal{R} = \left\{\mathcal{R} \middle| \frac{\|b^*_{k-i+1}\|}{\|b^*_j\|} \leq \mathcal{R}_i,\ for\ \mathcal{L}_{[j,k]}\ and\ j \leq i \leq k\right\} \tag{38}$$

Proof. By using the condition of $\frac{\|b^*_{k-i+1}\|}{\|b^*_j\|} \leq \mathcal{R}_i$ with enumeration radius $\|b^*_j\|$, it is possible to enumerate all coefficient vectors z with form of

$z = \{z_{0\leq i<g} = 0 | g \in [1,\dots,d]\}$ and consequently the solution vector at worst-case can be considered as $\min(\|\pi_j(b_j)\|,\dots,\|\pi_j(b_k)\|)$, therefore in general case, the solution vector v returned by GNR-enumeration in Lemma 5 has the norm of $\|v\| \leq \min(\|\pi_j(b_j)\|,\dots,\|\pi_j(b_k)\|)$. □

For just getting the best norm of basis (solving SVP in the lattice basis), after each round of BKZ_β, the shortest vectors of basis (based on Euclidean norm) are gathered smoothly down to the first indices of basis vectors, while we believe that the Lovasz condition nearly tries to accelerates this gathering negligibly by its two-dimensional projected sorting after each enumeration success, which is noted in Corollary 3. Also the best root-Hermite factor which is returned by LLL is not at all comparable with BKZ_β with high block sizes. Besides, the emulation of LLL makes our BKZ simulator more complicated and consequently heavier in runtime. By using these facts, why does it essential to use LLL method in a BKZ simulation? There are two main reasons for using the emulation of LLL after each enumeration success in design of BKZ simulation, as follows:

- *Problem 1*: Although just by use of updating GSO (and ignoring LLL function), the total correctness of BKZ simulation is not violated, but it should be noted that, the structure of formula Equation (33) drives the shape of next block to a worse quality, so the enumeration cost over the next block increases in an unpleasant way.
- *Problem 2*: Also just by using the model of updating GSO which is introduced in Section 3.1 (and ignoring LLL function), an overflow error in calculations over real numbers can be observed (especially in the case of using float point precision). This error is tensed for bigger block sizes. Note that, such errors can be observed in experimental running of original BKZ algorithm too. To find that how this error can be occurred, note to following scenario: BKZ simulation is performed over a lattice basis with dimension $n = 2000$ with block size $\beta = 200$; At first, full-$LLL_{\delta\approx 0.99}$ reduces the input basis, then the quality of basis reaches to $q \approx 1.045$, and accordingly, there is a ratio of $\|b_j^*\|/\|b_{j+\beta-1}^*\| \approx 2^{12.6}$ between the first vector and end vector of the block of $\mathcal{L}_{[j,k]}$; For $\|b_j^*\| \approx 2^{20}$, by using Equation (6), Equation (11) and Equation (12), Gaussian heuristic can be computed as $\text{GH}(\mathcal{L}_{[j,k]}) \approx 2^{13.7}$; Assume that, the returned solution v has a GSO norm which is close to the Gaussian heuristic (i.e., $\|v\| \approx 2^{13.7}$); By using Corollary 1 in [31], assume $g \approx \text{CUT} \approx d$, therefore this is expected that $\|b_{g-1}^*\| \approx 2^{7.43}$ and $\|b_g^*\| \approx 2^{7.4}$; Now for updating GSO, assume that the coefficient vector w is sampled by using formula Equation (26); By using Corollary 2 in [31], the expected value for w_{g-1}^2 and w_g^2 are approximated as $\text{E}\left[w_{g-1}^2\right] \approx 2^{4.9}$ and $\text{E}\left[w_g^2\right] \approx 1$; Accordingly, by using Equation (33), GSO norm of $\|b_{g_{new}}^*\|$ is approximated in average-case by $\|b_{g_{new}}^*\| \approx \frac{3}{100}\|b_{g-1}^*\| \approx 2^{2.37}$. In addition to the anomaly for GSO norm of $\|b_{g_{new}}^*\|$, this happening (anomaly) more or less would be done for other GSO vectors of $b_{i\,new}^*$ which are nearer to $b_{g_{new}}^*$ in the lattice block too. The GSO norm of $\|b_{g-1}^*\|$ would be lessen by a factor of $\sqrt{1-\frac{w_{g-1}^2\|b_{g-1}^*\|^2}{\|\pi_{g-1}(v)\|^2}}$ and located in index g in the updated block, so that for next enumeration it is considered as the index of $g-1$, and consequently this anomaly repeats with more decrease for next norm of $\|b_{g_{new}}^*\|$. Finally this leads to a condition for the basis which contains some GSO norms being extremely smaller than the real number's precision in the software/hardware platform for calculations.

By emulating Lovasz condition in BKZ simulation, these two problems will be solved. For this end, in any violation of Lovasz criterion, the relation Equation (37) should be applied in the way implemented in Algorithm 4 from Appendix A.1 (partial version of the LLL algorithm). After each satisfaction of Lovasz condition in LLL reduction, the GSO coefficient matrix μ can be updated by Lemma 6, as follows.

Lemma 6. *For GSO basis $b_1^*,\dots,b_n^*$, after satisfaction of Lovasz condition over $[b_i^*, b_{i+1}^*]$ in LLL_δ reduction with parameter of $\delta \approx 1$, the process of updating the GSO coefficient matrix μ can be done as follows:*

$$\mu''_{i+1,i} = \frac{\mu_{i+1,i}\|b_i^*\|^2}{\|b_{i+1}^*\|^2+\mu_{i+1,i}^2\|b_i^*\|^2} \tag{39}$$

$$\mu''_{l,i} = \frac{\mu_{l,i+1}\|b_{i+1}^*\|^2+\mu_{i+1,i}\mu_{l,i}\|b_i^*\|^2}{\|b_{i+1}^*\|^2+\mu_{i+1,i}^2\|b_i^*\|^2},\ i+2 \leq l \leq n \tag{40}$$

$$\mu''_{l,i+1} = \mu_{l,i} - \mu_{l,i+1} \times \mu_{i+1,i},\ i+2 \leq l \leq n \tag{41}$$

$\mu_{l,i+1} \leftarrow \mu''_{l,i+1},\ i+2 \leq l \leq n$

$\mu_{l,i} \leftarrow \mu''_{l,i},\ i+2 \leq l \leq n$

For $1 \leq t \leq i-1$: swap_entry$(\mu_{i,t}, \mu_{i+1,t})$
/ $\mu_{i,i}$ and $\mu_{i+1,i+1}$ remain 1 * /*

$\mu_{i+1,i} \leftarrow \mu''_{i+1,i}$

See proof in Appendix B.5.

Note: The loop for relation Equation (40) and Equation (41) should be applied for up to end of basis vectors (i.e., end index of n).

Note: Our observations show that Lemma 6 is fully consistent with LLL function (which is implemented

by NTL).

Remark 6. For an input GSO basis $b_1^*, \ldots, b_n^*$ in our emulated LLL, before checking the Lovasz condition over the GSO vectors of b_i^* and b_{i+1}^*, the emulation of size reduction process for vector b_{i+1} should be applied on matrix μ, similar to the original loop in partial/full-LLL (Algorithm 3 and Algorithm 4 in Appendix A.1).

Note: The decreasing counter in the outer loop of Size-Reduction step of partial/full-LLL (Algorithm 3 and Algorithm 4 in Appendix A.1) is essential for correctness of size reduction process.

Also the pseudo-code of our emulation of partial-LLL is introduced in Algorithm 8 from Appendix A.3 (which represents full-LLL and partial-LLL).

3.3 The Emulation of BKZ Algorithm

As discussed in Section 3.1 and Section 3.2, our contributions in this paper are two provable tools of "Emulation of updating GSO norms/coefficients" (see Lemma 1 and Lemma 2) and "Emulation of LLL function" (see Lemma 4 and Lemma 6) as two main parts in designing an emulation for BKZ. Algorithm 1 shows the pseudo-code of our emulation of BKZ algorithm (our accurate BKZ Simulation).

Algorithm 1 Our emulation of BKZ algorithm (Our Accurate BKZ Simulation)

Input: GSO norms $\ell_i = \log(\|b_i^*\|)$ of basis B for $i = 1, \ldots, n$; $2 \leq \beta \leq n$, GSO Coef Matrix μ, $1/4 \leq \delta < 1$, SVPSolver's input_parameters of $Param$, number of rounds $\text{ROUND}_{\text{NUM}}$.
Output: GSO norms $\ell_{[1,\ldots,n]}$ and GSO matrix μ for simulated output basis.

1: emulate_LLL($\ell_{[1,\ldots,n]}, \mu, \delta$);
2: **for** ($l = 1, 2, \ldots, \text{ROUND}_{\text{NUM}}$) **do**
3: **for** ($j = 1, 2, \ldots, n-2$) **do**
4: $k = \min(j + \beta - 1, n)$;
5: $h = \min(k + 1, n)$;
6: $[\ell_{new}, y, cost] \leftarrow$ emulate_SVPSolver($\ell_{[j\ldots k]}, Param$);
7: **if** $\ell_{new} < \ell_j$ **then** //equivalent to condition: $if(y \neq (1, 0, \ldots, 0))$
8: $\ell'_{[1,\ldots,n]} \leftarrow [\ell_1, \ldots, \ell_{j-1}, \ell_{new}, \ell_j, \ldots, \ell_n]$;
$\ell_{[1,\ldots,n]} \leftarrow$ emulate_UpdateGSO($\ell'_{[1,\ldots,n]}, y, \mu, j, k$);
emulate_LLL($\ell_1, \ldots, \ell_h, \mu, \delta$) at stage j;
9: **else**
10: emulate_LLL($\ell_1, \ldots, \ell_h, \mu, \delta$) at stage $h - 1$;
11: **end if**
12: **end for**
13: **end for**

By using following theorem, this paper proves that for a typical SVP solver "Z" (e.g., GNR-enumeration, Sieving, discrete pruning, etc.), if there is a simulation of "Z_emulate" which provably emulates the behaviour of practical running of "Z", then Algorithm 1 by using "emulate_SVPSolver"="Z_emulate" can provably emulates the Experimental Running of BKZ algorithm (Algorithm 6) using SVP solver "Z".

Theorem 4. *If the function of "emulate_SVPSolver" in Algorithm 1 emulates the behaviour of corresponding function of "SVPSolver" in Algorithm 6 (under Definition 1), then Algorithm 1 emulates Algorithm 6.*

Proof. Based on Definition 1 in the scope of this research, x can be assumed as input basis B to BKZ, and y can be assumed as output basis B' reduced by BKZ. In other hand, in the scope of this research, X can be assumed as the collection of GSO norms $\ell_i = \log(\|b_i^*\|)$ together with GSO Coefficient Matrix μ from input basis B to BKZ, while Y can be assumed as the collection of GSO norms $\ell'_i = \log(\|b'^*_i\|)$ together with GSO Coefficient Matrix μ' from output basis B' reduced by BKZ.

Lemma 1 and Lemma 2 prove that the function of "emulate_UpdateGSO" emulates the task of updating GSO norms/coefficients, which is embedded in LLL function at lines of 1,7,8,9,10,11 from Algorithm 6.

Lemma 4 and Lemma 6 prove that "emulate_LLL" emulates LLL function at lines of 1,7,8,9,10,11 from Algorithm 6.

By assumption of that "emulate_SVPSolver" in Algorithm 1 emulates the behaviour of "SVPSolver" in Algorithm 6, all lines of $1, 2, \ldots, 13$ from Algorithm 1 are completely as the emulation of corresponding lines of $1, 2, \ldots, 13$ in Algorithm 6. □

For practical use, our accurate BKZ Simulation in Algorithm 1 needs to an exact emulation of SVP-Solver which should be memoryless and fast enough. There are two classes of SVP-Solvers[4]: Exponential-space SVP-Solvers (like sieve algorithms) and Polynomial-space SVP-Solvers (like enumeration algorithms). Most implementations of BKZ (such as [2, 3, 12, 13, 41]) use enumeration with cylinder pruning [34]. If the approximation factor (in approximate-SVP) would be relaxed, pruned enumeration can heuristically achieve bigger exponential speed-ups than sieving SVP-Solver's one [4].

Our former contributions and achievements in [29–32] can introduce a claimant simulation of GNR-enumeration. However our works of [30–32] may not be a true emulation of GNR-enumeration, because of using some heuristics in their design and analysis, such as Heuristic 1 (Gaussian Heuristic), Heuristic 2, Heuristic 3 and Geometric Series Assumption (GSA). Therefore, this paper don't claim that our proposed simulation of GNR-enumeration in Algorithm 2 would be an emulation of GNR-enumeration, while it is just a claimant simulation of GNR-enumeration. Accordingly, if Algorithm 2 would be called in line 6 from

Algorithm 1, then Algorithm 1 only works as an accurate simulation of BKZ (not as an emulation of BKZ).

Algorithm 2 Simulate_GNREnumeration

Input: GSO norms of lattice block $\ell_{[j\ldots k]} = [\log(\|b_j^*\|), \log(\|b_{j+1}^*\|), \ldots, \log(\|b_k^*\|)]$ from input basis B, radius param $\sqrt{\Upsilon}$, $\mathrm{p_{SUCCT}}$ as success probability of bounding function $\mathcal{R}$.

Output: return $[\ell_{new}, y, cost]$.

1: GH = compute_GaussianHeuristic($\ell_{[j\ldots k]}$);
2: r_{FAC} = compute_radFac($\ell_{[j\ldots k]}, \sqrt{\Upsilon}$);
3: $R = r_{\mathrm{FAC}} \times \mathrm{GH}$;//compute enumeration radius
4: loop{//begin loop
5: $\mathcal{R} \leftarrow$ generate_BoundingFunction($\mathrm{p_{SUCCT}}, \ell_{[j\ldots k]}$);
6: $\mathrm{p_{SUCCT}}'$ = estimate_Psucc($\mathcal{R}, \ell_{[j\ldots k]}$);
7: }until($\mathrm{p_{SUCCT}}' \approx \mathrm{p_{SUCCT}}$);//end loop
8: $\ell_{new} \leftarrow$ Sample_Norm_of_EnumSolution($\ell_{[j\ldots k]}, r_{\mathrm{FAC}}$);
9: $y \leftarrow$ Sample_of_y($\ell_{[j\ldots k]}, \mathcal{R}, R, \ell_{new}$);
10: $cost$ = estimate_cost($\ell_{[j\ldots k]}, \mathcal{R}, R, \mathrm{GH}$);

Algorithm 2 includes some main components which are studied in the author's former works, as follow:

- The function of "compute_radFac" returns the radius factor of r_{FAC} based on lattice block and radius parameter of $\sqrt{\Upsilon}$ (see our discussions on enumeration radius factor in Section 3.1 from [29]).
- The function of "generate_BoundingFunction" generates the bounding function based on specified success probability and current lattice block (see our discussions on generation of optimal bounding function in [30]).
- The function of "estimate_Psucc" estimates the success probability of bonding function $\mathcal{R}$ with current configuration of GNR-enumeration over lattice block of $\ell_{[j\ldots k]}$ (see our discussions on estimation of success probability in [29]).
- A sample of norm of GNR-enumeration solution vector is generated by the function of "Sample_Norm_of_EnumSolution" (see our discussions on sampling the norm of GNR-enumeration solution vector in [31]).
- The function of "Sample_of_y" generates a sample of coefficient vector y for GNR-enumeration solution (see our discussions on sampling the coefficient vectors of z, u, w, y for GNR-enumeration solution in our work of [31] and our revised techniques from [32]).
- The function of "estimate_cost" estimates the cost of GNR-enumeration function with current configuration (see our discussions on estimation of enumeration cost in [29]).

4 Our Test Results

This section shows the value of our contributions by using sufficient simulation/experimental test results. All the implementations and simulations are compiled with MSVC x64 bit C++. Also our tests use the following hardware platform: ASUS motherboard series Z97-K, Intel® Core™ $i7-4790K$ processor with the base frequency of 4 GHz, 16 GB RAM. Also, the running times (in seconds) in Section 4.3 are assumed to be provided for each single real-core.

This section uses 20 random lattice bases in the sense of Goldstein and Mayer (SVP lattice challenges) [42, 43] with dimension of $n = 100$, with seeds of $1, 2, 3, \ldots, 20$ (include "`vpchallengedim100eed1`", ..., "`vpchallengedim100eed20`"). The average value of Gaussian parameters of these 20 random lattice bases is $\mathrm{GH}(\mathcal{L}) \approx 2541.25$ (see formula Equation (6)); This paper dose not apply any more randomization technique on these 20 random lattice bases (such as randomizing technique introduced by Martin R Albrecht in [44]), however all these 20 random lattice bases, before our tests being applied on them, are reduced by LLL function of "`long LLL(ZZ& det, mat_ZZ& B, long a=99, long b=100, long verbose = 0)`" in NTL library [41], and consequently, the average value of root-Hermite factor of these 20 LLL-reduced (random) lattice bases is $\mathrm{RHF} \approx 1.020239$.

Our test results in this paper use four main instances of BKZ as follows: "*Experimental running of Original BKZ algorithm*" (Algorithm 5 from Appendix A.2), "*Our BKZ simulation*" (Algorithm 1), "*Chen-Nguyen's BKZ simulation*" (see Algorithm 2 from [2]) and "*BKZ Simulation by Shi Bai* et al." (see Algorithm 3 from [5]); All these four instances of BKZ are supposed to be run with full-enumeration (nearly as exact-SVP Solver of BKZ with success probability $\mathrm{p_{succ}} = 1$) over 20 random lattice bases. The block size in running of BKZ (either in original BKZ algorithm or BKZ simulations) in this test is set to $\beta = 50$; The LLL algorithm in all the tests of this paper (either in original BKZ algorithm or BKZ simulations) uses the parameter of $\delta = 0.99$.

Remark 7. This paper uses function of "`long BKZ_FP(mat_ZZ& BB, mat_ZZ* UU, double delta, long beta, long prune, LLLCheckFct check)`" with parameter of "`prune=0`" from NTL library [41] as the implementation of "Original BKZ algorithm" (Algorithm 5 from Appendix A.2); In fact, this paper dose not guarantees that function of "`BKZ_FP`" implements Algorithm 5 from Appendix A.2 exactly. The function of "`BKZ_FP`" may include some software speedups, memory saving techniques, float point errors in running, etc., which cause that, this function ("`BKZ_FP`") dose not implement exactly the procedure of "Original BKZ algorithm" (Algorithm 5 from Appendix A.2), while "Our BKZ simulation" (Algorithm 1) is nearly designed to simulate "Original BKZ algorithm" (Algorithm 5 from Appendix

A.2). Accordingly, our test results in this section may cannot show the true closeness of "Our BKZ simulation" to "Original BKZ algorithm", just if the function of "`BKZ_FP`" dose not implement exactly (and truly) "Original BKZ algorithm" (Algorithm 5 from Appendix A.2).

Note: In this paper, the phrase of "Original BKZ algorithm" or "Experimental running of Original BKZ algorithm" refers to the same BKZ instance (i.e., the practical implementation of Algorithm 5 from Appendix A.2).

Our BKZ Simulation (Algorithm 1) in the tests of this section uses the sampling method of solution norm by formula Equation (29) from Theorem 2 (which is fully compatible with our revised technique of sampling solution norm in Lemma 3 from [31]). Moreover, our BKZ simulation (Algorithm 1) in the tests of this section uses the idea of sampling method of coefficient vectors of z, u, w, y in Lemma 10 from [31], which is defined as follows (see our complementary discussions on sampling coefficient vectors in [32]):

$$w_l^2 = \begin{cases} \frac{\|v\|^2 - \|b_g^*\|^2}{\|b_l^*\|^2 \times (d-1)} & for\ 1 \leq l \leq g-1 \\ 1 & for\ l = g \\ 0 & for\ g < l \leq d \end{cases} \quad (42)$$

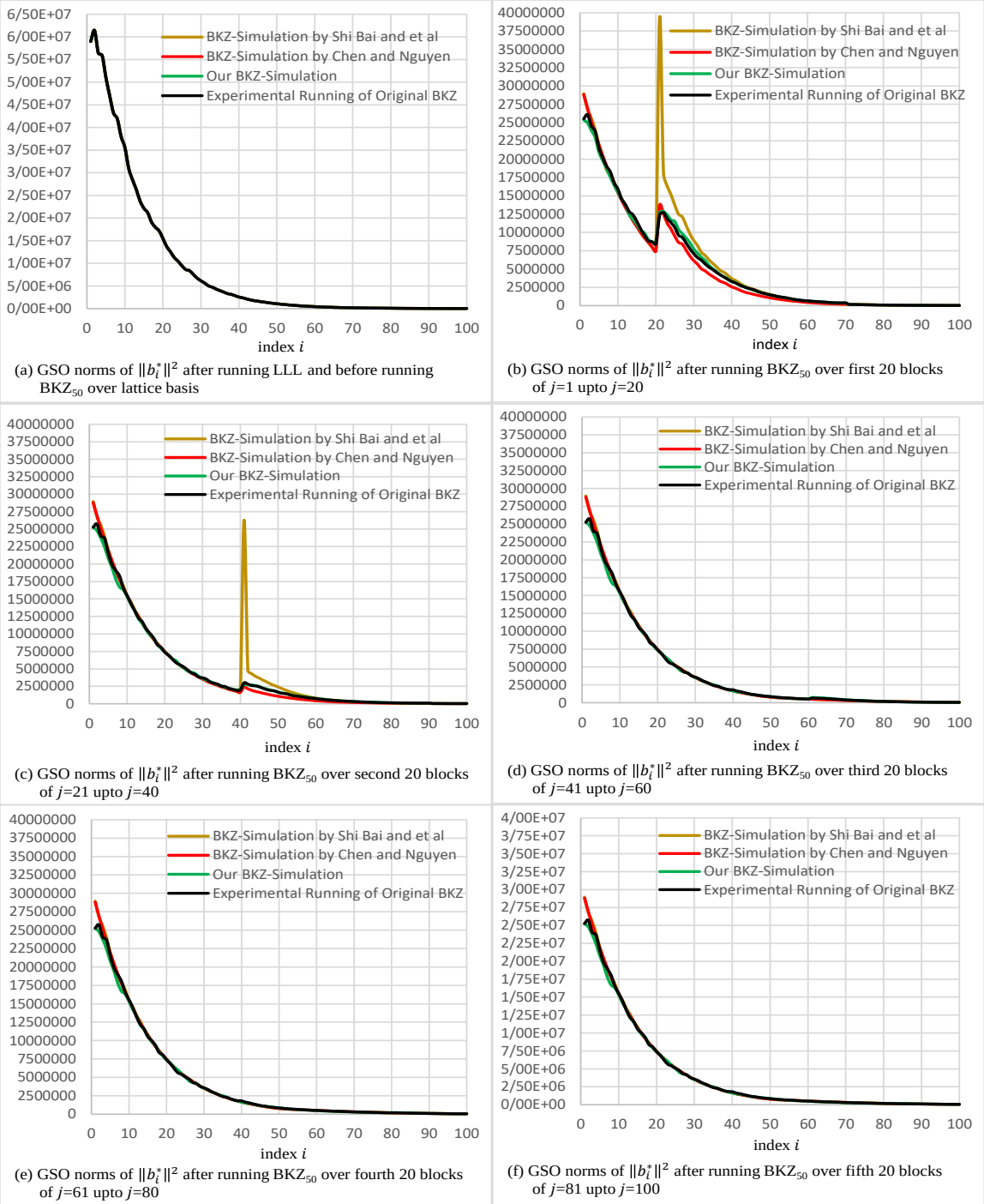
4.1 GSO-Norms of $\|b_i^*\|^2$ in Running of BKZ-Algorithm/BKZ-Simulations

In this section, GSO-norms of $\|b_i^*\|^2$ in experimental running of original BKZ algorithm would be compared with corresponding norms of $\|b_i^*\|^2$ in running of Chen-Nguyen's BKZ simulation [2], BKZ Simulation by Shi Bai *et al.* [5] and our BKZ simulation (Algorithm 1); Figure 1 shows the average value of GSO norms of $\|b_i^*\|^2$ after applying "Round 1" of BKZ_{50} (in "Experimental Running of Original BKZ algorithm", "Chen-Nguyen's BKZ simulation", "BKZ Simulation by Shi Bai *et al.*" and "Our BKZ simulation") over 20 random lattice bases at each step of BKZ (which includes enumerations over 20 subsequent lattice blocks at each subfigures). Our strategy in this test is to compare the closeness of GSO norms of $\|b_i^*\|^2$ in "Experimental Running of Original BKZ algorithm" to the norms of $\|b_i^*\|^2$ in "Our BKZ Simulation", against the closeness of the GSO norms $\|b_i^*\|^2$ in "Experimental Running of Original BKZ algorithm" to the GSO norms of $\|b_i^*\|^2$ in "Chen-Nguyen's BKZ simulation" and "BKZ Simulation by Shi Bai *et al.*".

Note: Each point in every subfigures of Figure 1 show GSO norm $\|b_i^*\|^2$ in projected GSO lattice basis of $[b_1^*, \ldots, b_i^*, \ldots, b_n^*]$.

As shown in Figure 1-a, the GSO norms of $\|b_i^*\|^2$ after running LLL and before running BKZ_{50} over each 20 random lattice basis for all four instances of BKZ in this test (i.e., "Original BKZ algorithm", "Chen-Nguyen's BKZ simulation", "BKZ Simulation by Shi Bai *et al.*" and "Our BKZ simulation") are exactly similar. Then in Figure 1-b, "Original BKZ algorithm", "Chen-Nguyen's BKZ simulation", "BKZ Simulation by Shi Bai *et al.*" and "Our BKZ simulation" would be run over only first 20 lattice blocks of $\mathcal{L}_{[1,50]}$, $\mathcal{L}_{[2,51]}$, ..., $\mathcal{L}_{[20,69]}$. As shown in Figure 1-b, the GSO norms of $\|b_i^*\|^2$ by original BKZ algorithm are more close to the norms of $\|b_i^*\|^2$ by our BKZ simulation than to the GSO norms of $\|b_i^*\|^2$ by "Chen-Nguyen's BKZ simulation" or "BKZ Simulation by Shi Bai *et al.*". Also as shown in Figure 1-c up to Figure 1-f (in running of these four instances of BKZ algorithm/simulation over next packs of 20 subsequent lattice blocks), again the GSO norms of $\|b_i^*\|^2$ by original BKZ algorithm are more close to the norms of $\|b_i^*\|^2$ by our BKZ simulation than to the GSO norms of $\|b_i^*\|^2$ by "Chen-Nguyen's BKZ simulation" or "BKZ Simulation by Shi Bai *et al.*". Note that, since "Chen-Nguyen's BKZ simulation" and "BKZ Simulation by Shi Bai *et al.*" use "the average $\log(\|b_i^*\|)$ of an HKZ-reduced random unit-volume 45-dimensional lattice" as the output GSO norms of last 45 vectors of basis after one round of running BKZ simulation, therefore the shape of GSO norms of $\|b_i^*\|^2$ after running Chen-Nguyen's BKZ simulation for third 20 blocks (in Figure 1-d) and fourth 20 blocks (in Figure 1-e) are assumed to be equal to the final shape of GSO norms of $\|b_i^*\|^2$ after completing "Round 1" of Chen-Nguyen's BKZ simulation and BKZ Simulation by Shi Bai *et al.* which is shown in Figure 1-f; Altogether, in all steps of Figure 1, the shape of the GSO norms of $\|b_i^*\|^2$ of "Original BKZ algorithm" are more close to the GSO norms of $\|b_i^*\|^2$ of "Our BKZ simulation" than to "Chen-Nguyen's BKZ simulation" and "BKZ Simulation by Shi Bai *et al.*".

Moreover, Figure 2 shows the average value of GSO norms of $\|b_i^*\|^2$ after applying "Rounds $1, 2, 3, \ldots, 12$" of BKZ_{50} (in "Experimental Running of Original BKZ algorithm", "Chen-Nguyen's BKZ simulation", "BKZ Simulation by Shi Bai *et al.*" and "Our BKZ simulation") over 20 random lattice bases; Figure 2 shows that the average value of GSO norms of $\|b_i^*\|^2$ after applying "Rounds $1, 2, 3, \ldots, 12$" of "Our BKZ simulation" is bit less than "Experimental Running of Original BKZ algorithm", however this phenomenon may be caused by our discussions in Remark 7 (or not).



(a) GSO norms of $\|b_i^*\|^2$ after running LLL and before running BKZ_{50} over lattice basis

(b) GSO norms of $\|b_i^*\|^2$ after running BKZ_{50} over first 20 blocks of j=1 upto j=20

(c) GSO norms of $\|b_i^*\|^2$ after running BKZ_{50} over second 20 blocks of j=21 upto j=40

(d) GSO norms of $\|b_i^*\|^2$ after running BKZ_{50} over third 20 blocks of j=41 upto j=60

(e) GSO norms of $\|b_i^*\|^2$ after running BKZ_{50} over fourth 20 blocks of j=61 upto j=80

(f) GSO norms of $\|b_i^*\|^2$ after running BKZ_{50} over fifth 20 blocks of j=81 upto j=100

Figure 1. Average values of GSO norms of $\|b_i^*\|^2$ after running BKZ_{50} (in "Experimental Running of Original BKZ Algorithm", "Our BKZ Simulation", "Chen-Nguyen's BKZ Simulation" and "BKZ Simulation by Shi Bai *et al.*") over 20 random lattice bases at each step of "Round 1" of BKZ (including enumerations over 20 subsequent lattice blocks at each subfigures)

4.2 Root-Hermite Factor of Basis After Run of BKZ-Algorithm/BKZ-Simulations

In this section, the Root-Hermite factor (RHF) of basis after running of Original BKZ algorithm would be compared with the Root-Hermite factor of basis after running of Chen-Nguyen's BKZ simulation [2], BKZ simulation by Shi Bai *et al.* [5] and our BKZ simulation (Algorithm 1); Figure 3 shows the average values of Root-Hermite factor of basis after applying "Rounds $1, 2, 3, \ldots, 12$" of BKZ_{50} instances (include "Original BKZ algorithm", "Our BKZ simulation", "Chen-Nguyen's BKZ simulation" and "BKZ simulation by Shi Bai *et al.*") over 20 random lattice bases.

Similar to our strategy in Section 4.1, in Figure 3,

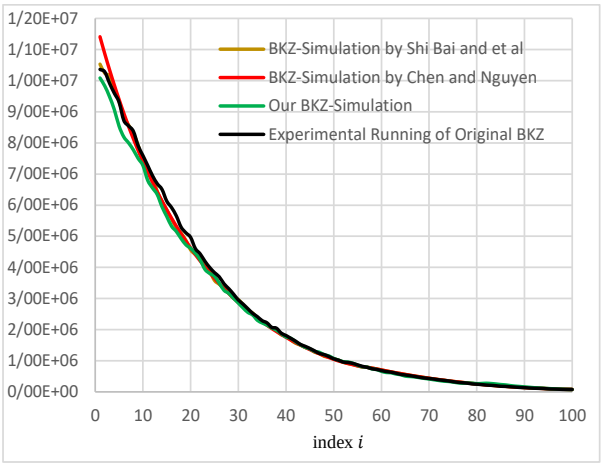


Figure 2. Average values of GSO norms of $\|b_i^*\|^2$ after applying "Rounds $1, 2, 3, \ldots, 12$" of BKZ_{50} (in "Original BKZ Algorithm", "Chen-Nguyen's BKZ Simulation", "BKZ Simulation by Shi Bai *et al.*" and "Our BKZ Simulation") over 20 random lattice bases

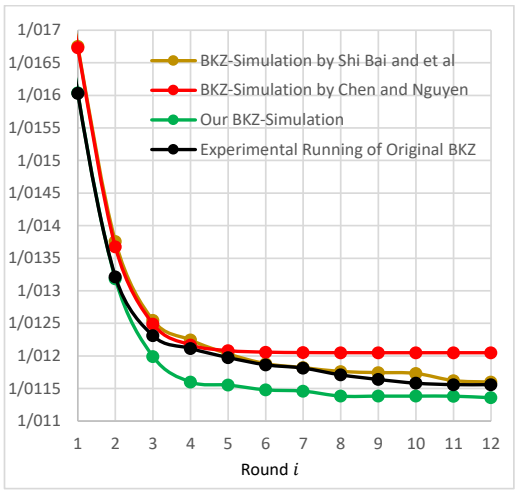


Figure 3. Average values of Root-Hermite Factor (RHF) of basis after applying "Rounds $1, 2, 3, \ldots, 12$" of four mentioned BKZ_{50} instances (with block size $\beta = 50$) over 20 random lattice bases

we compare the closeness of the root-Hermite factors of basis after applying different rounds of "Experimental Running of Original BKZ algorithm" to the corresponding root-Hermite factors in "Our BKZ Simulation", against the closeness of the root-Hermite factors of basis after applying different rounds of "Experimental Running of Original BKZ algorithm" to the corresponding root-Hermite factors in "BKZ simulation by Shi Bai *et al.*" and "Chen-Nguyen's BKZ simulation"; As shown in Figure 3, the root-Hermite factors of basis after applying some rounds of "Experimental Running of Original BKZ algorithm" are more closed to the corresponding root-Hermite factors in case of "Our BKZ Simulation", and for some other rounds are more closed to the corresponding root-Hermite factors in case of "BKZ simulation by Shi Bai *et al.*". However this phenomenon may be caused by our discussions in Remark 7 (or not).

Moreover, Table 1 compares the average values of Root-Hermite factor (RHF) of basis after applying "Round 1" and also after applying "Rounds $1, 2, \ldots, 12$" of BKZ_{50} instances over 20 random lattice bases; Table 1 includes Chen's approximation [39] for root-Hermite factor of basis after running the BKZ algorithm (see relation Equation (15)).

Note: Since to the best of our knowledge, Chen's approximation is introduced for the basis being reduced by original BKZ algorithm with full enumeration, lattice block size of β and infinite number of rounds, so it is compared just with Root-Hermite factors of the bases being reduced by last round of BKZ instances in this test (i.e., after "Round 12" of BKZ).

As shown in Table 1, the average root-Hermite factor of basis after applying "Round 1" of "Experimental Running of Original BKZ algorithm" are more close to the corresponding root-Hermite factor in case of "Our BKZ simulation"; In other side, the root-Hermite factors of basis after applying "Rounds $1, 2, \ldots, 12$" of "Experimental Running of Original BKZ algorithm" are a bit more close to the corresponding root-Hermite factors in the case of "BKZ simulation by Shi Bai *et al.*". However this phenomenon (after "Rounds $1, 2, \ldots, 12$") may be caused by our discussions in Remark 7 (or not).

4.3 Cost Results in Running of BKZ-Algorithm/BKZ-Simulations

In this section, the cost results of "Experimental Running of Original BKZ algorithm" would be compared with the cost of applying Chen-Nguyen's BKZ simulation [2], BKZ simulation by Shi Bai *et al.* [5], Our BKZ simulation (Algorithm 1) and some cost model of BKZ. This subsection uses the cost model of $CM1$ as follows.

CM1: The cost of enumeration function on lattice block of $\mathcal{L}_{[1\ldots\beta]}$ can be estimated as follows [20] (which are fitting the same data from [39]):

$$\mathrm{CM1}(\beta) = 2^{0.187\beta\log_2(\beta) - 1.019\beta + 16.1} \tag{43}$$

The total cost of BKZ by cost model of CM1 can be computed simply as follows ("$\mathrm{ROUND}_{\mathrm{NUM}}$" is the number of rounds of BKZ):

$$\begin{aligned} &Cost_{BKZ}(n, \beta, \mathrm{ROUND}_{\mathrm{NUM}}) = \\ &(n - \beta + 1) \times \mathrm{CM1}(\beta) \times \mathrm{ROUND}_{\mathrm{NUM}} \end{aligned} \tag{44}$$

Note: The phrase of "Enumeration Cost" in this paper is referred to the number of enumeration nodes which are enumerated at each subsequent lattice enumerations of BKZ.

Table 1. A comparison between the average values of Root-Hermite Factor (RHF) of basis after applying "Round 1" and also "Rounds $1,\ldots,12$" of "Original BKZ Algorithm", "Our BKZ Simulation", "Chen-Nguyen's BKZ Simulation" and "BKZ Simulation by Shi Bai & *et al.*" (and also Chen's Approximation [39]) over 20 random lattice bases with dimension $n = 100$ and block size $\beta = 50$

	Experimental Running of Original BKZ algorithm	Our BKZ simulation	Chen-Nguyen's BKZ simulation	BKZ simulation by Shi Bai *et al.* [5]	Approximation in Chen's thesis; see relation (15)
Root-Hermite Factor after Round 1	1.01603	1.01604	1.01673	1.01675	-
Root-Hermite Factor after Rounds 1,...,12	1.01156	1.01136	1.01205	1.01160	1.01206

Note: The phrase of "Cost of BKZ" in this paper is referred to the sum of the total number of enumeration nodes which are enumerated at all subsequent lattice enumerations of BKZ.

Note: The phrase of "RunTime of BKZ" in this paper is referred to sum of the total running time (in seconds) of all subsequent lattice enumerations in BKZ.

Figure 4 shows the $\log_2$ of average values of enumeration cost (nodes) in running "Round 1" of BKZ_{50} instances (include "Experimental Running of Original BKZ algorithm", "Our BKZ simulation", "BKZ simulation by Shi Bai *et al.*", "Chen-Nguyen's BKZ simulation" and "Cost Model of CM1") over each lattice block of $\mathcal{L}_{[1,50]}$, $\mathcal{L}_{[2,51]}$, ..., $\mathcal{L}_{[99,100]}$ in 20 random lattice bases. Each point at (lattice block) index j in Figure 4 shows the $\log_2$ of total enumeration nodes in full-enumerations over each lattice block of $\mathcal{L}_{[j,\min(j+50-1,100)]}$ (i.e., $\mathcal{L}_{[1,50]}$, $\mathcal{L}_{[2,51]}$, ..., $\mathcal{L}_{[99,100]}$) which is estimated by using the formula of Equation (16).

Note: However we discuss a revised technique for estimation of GNR-enumeration cost in [29], this paper dose not need to that technique for our tests in this section; In fact, since our tests in this section use only full-enumerations over lattice blocks in "Original BKZ algorithm", "Our BKZ simulation", "Chen-Nguyen's BKZ simulation" and "BKZ simulation by Shi Bai *et al.*", the estimation of total nodes of these full-enumerations can be estimated just by using the simple formula of Equation (16).

Similar to our strategy in Section 4.1 and Section 4.2, Figure 4 compares the closeness of enumeration costs in "Experimental Running of Original BKZ algorithm" to "Our BKZ Simulation" against closeness of enumeration costs in "Experimental Running of Original BKZ algorithm" to "BKZ simulation by Shi Bai *et al.*", "Chen-Nguyen's BKZ simulation" and "Cost Model of CM1". As shown in Figure 4, this is obvious that the enumeration costs in "Experimental Running of Original BKZ algorithm" are more closed to the enumeration costs in "Our BKZ Simulation" than three other instances of BKZ.

Figure 5 shows the $\log_2$ of average total cost of running each "Rounda $1,2,3,\ldots,12$" of BKZ_{50} instances (include "Experimental Running of Original BKZ algorithm", "Our BKZ simulation", "Chen-Nguyen's BKZ simulation" and "BKZ simulation by Shi Bai *et al.*") and "Cost Model of CM1" over 20 random lattice bases.

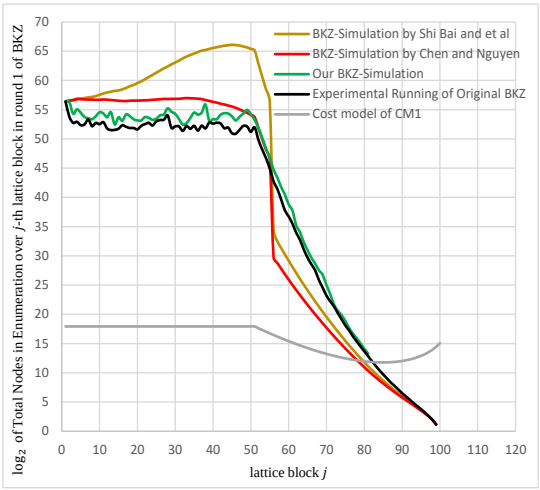


Figure 4. $\log_2$ of average value of enumeration cost in running "Round 1" of different BKZ_{50} instances (with block size $\beta = 50$) over each lattice block of $\mathcal{L}_{[j,\min(j+50-1,100)]}$ (i.e., $\mathcal{L}_{[1,50]}$, $\mathcal{L}_{[2,51]}$, ..., $\mathcal{L}_{[99,100]}$) in 20 random lattice bases

As shown in Figure 5, this is obvious that the average total cost of "Experimental Running of Original BKZ algorithm" at each "Rounds $1,2,3,\ldots,12$" is more close to the average total cost of "Our BKZ simulation", than to the average total cost of "Chen-Nguyen's BKZ simulation", "BKZ simulation by Shi Bai *et al.*" and "Cost Model of CM1".

Also, Table 2 shows the average total cost (nodes) of "Round 1" and "Rounds $1,2,\ldots,12$" of BKZ instances (include "Experimental Running of Original BKZ algorithm", "Our BKZ Simulation", "Chen-Nguyen's BKZ simulation" and "BKZ simulation by Shi Bai *et al.*") and "Cost Model of CM1" over 20 random lattice bases.

As shown in Table 2, this is obvious that the average total cost of "Experimental Running of Original BKZ algorithm" at "Round 1" and also whole "Rounds of $1,2,3,\ldots,12$" is more close to the average total cost of "Our BKZ simulation", than to the average

Table 2. The average total cost of "Round 1" and "Rounds $1, 2, \ldots, 12$" of four mentioned BKZ instances and "Cost Model of CM1" over 20 random lattice bases

	Experimental Running of Original BKZ algorithm	Our BKZ simulation	Chen-Nguyen's BKZ simulation	BKZ simulation by Shi Bai *et al.* [5]	Total cost of BKZ by cost models of CM1
Total Cost of Round 1 of BKZ	$\approx 2^{58.5}$	$\approx 2^{59.9}$	$\approx 2^{62.2}$	$\approx 2^{70}$	$\approx 2^{23.7}$
Total Cost of Rounds 1,2,...,12 of BKZ	$\approx 2^{59.3}$	$\approx 2^{61.9}$	$\approx 2^{62.2}$	$\approx 2^{70}$	$\approx 2^{27.3}$

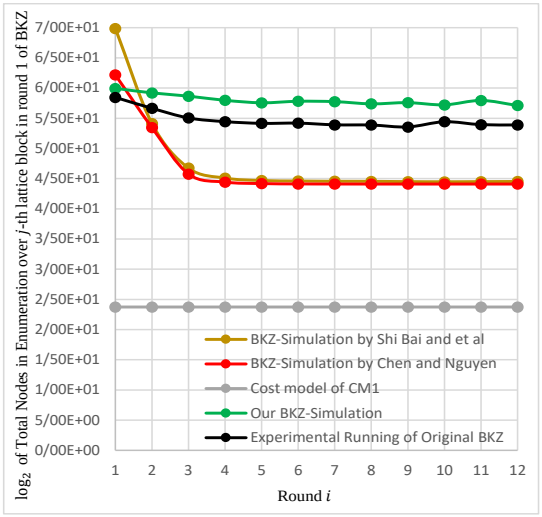


Figure 5. The $\log_2$ of average total cost of "Experimental Running of Original BKZ Algorithm", "Our BKZ Simulation", "Chen-Nguyen's BKZ Simulation", "BKZ Simulation by Shi Bai *et al.*" and "Cost Model of CM1" at each "Rounds $1, 2, 3, \ldots, 12$" of BKZ over 20 random lattice bases

total cost of "Chen-Nguyen's BKZ simulation", "BKZ simulation by Shi Bai *et al.*" and "Cost Model of CM1".

Note: As shown accurately in Figure 5, the cost of "Chen-Nguyen's BKZ simulation" cannot be supposed similar to the cost of "Our BKZ simulation", therefore the cost result of "Chen-Nguyen's BKZ simulation" in Table 2 cannot represent the similarity to the cost result of "Our BKZ simulation".

Moreover, Table 3 shows the average RunTime of "Round 1" and "Rounds $1, 2, \ldots, 12$" of BKZ instances (include "Experimental Running of Original BKZ algorithm", "Our BKZ Simulation", "Chen-Nguyen's BKZ simulation" and "BKZ simulation by Shi Bai *et al.*") and "Cost Model of CM1" over 20 random lattice bases. The total runtime of applying "Experimental Running of Original BKZ algorithm" over each random lattice basis is recorded by using the function of "`double GetTime()`" over each enumeration at function of "`BKZ_FP`" in NTL library [41] on our Hardware platform (which is specified at the first of this section). Also, for computing the average runtime of enumerations in the BKZ Simulations and the Cost Model of CM1, this paper uses massive tests to approximate the number of nodes enumerated in lattice enumerations of "Original BKZ algorithm" in "one second" over 12240 lattice blocks with $\beta = 50$, and over 11520 lattice blocks with $\beta = 2, 3, 4, \ldots, 49$; Finally the results of this test shows that the average number of enumeration nodes which can be enumerated in lattice enumeration function (of BKZ) is almost $N_{avg} \approx 2^{44.3}$ nodes per second (number of nodes enumerated at one second) on our Hardware platform in this paper. By using N_{avg}, the average total cost (i.e., sum of total enumeration nodes) for "Round 1" and "Rounds $1, 2, \ldots, 12$" of "Our BKZ Simulation", "Chen-Nguyen's BKZ simulation", "BKZ simulation by Shi Bai *et al.*" and "Cost Model of CM1" over 20 random lattice bases in Table 2 can be simply change into the average total RunTime of them in Table 3. For better sense, the format of RunTime in Table 3 is set to "Years/Days/Hours/Minutes/Seconds/MicroSeconds".

These surprising gaps between the predictions of RunTime are not negligible in Lattice-based security analysis. Although the best RunTime predictions is belong to "Our BKZ Simulation", the gap between the RunTime of "Our BKZ Simulation" and "Experimental Running of Original BKZ algorithm" is not small yet (however this gap may be caused partially by our discussions in Remark 7 or not).

Note: The approximation of $N_{avg} \approx 2^{44.3}$ (the average number of enumeration nodes which can be enumerated in lattice enumeration function on our Hardware/Software platform in this paper) seems to be non-accurate and should be re-estimated with better tests in further studies.

Note: As mentioned before, based on our results in Figure 5, the cost of "Chen-Nguyen's BKZ simulation" cannot be supposed to be similar to the cost of "Our BKZ simulation", therefore the RunTime of "Chen-Nguyen's BKZ simulation" in Table 3 cannot represent the similarity to the RunTime of "Our BKZ simulation".

4.4 Test Results for GSO Violation Errors

Chen-Nguyen's BKZ simulation uses $\text{GH}(\mathcal{L}_{[j,k]})$ for expecting the GSO norm of full-enumeration solution vector, but wrong strategy of this simulation to update GSO norms causes many GSO violation er-

Table 3. The average RunTime of "Round 1" and "Rounds $1, 2, \ldots, 12$" of some BKZ instances and Cost Model of "CM1" over 20 random lattice bases

	Experimental Running of Original BKZ algorithm	Our BKZ-Simulation	Chen-Nguyen's BKZ simulation	BKZ simulation by Shi Bai *et al.* [5]	Total cost of BKZ by cost models of CM1
RunTime of Round 1 of BKZ	-	-	-	1 Year/	-
	-	-	2 Days/	108 Days/	-
	1 Hours/	11 Hours/	6 Hours/	13 Hours/	-
	35 Minutes/	38 Minutes/	28 Minutes/	41 Minutes/	-
	53 Seconds/	44 Seconds/	32 Seconds/	31 Seconds/	-
	-	-	-	-	0.5 MicroSeconds/
RunTime of Rounds 1,2,...,12 of BKZ	-	-	-	1 Year/	-
	-	1 Days/	2 Days/	108 Days/	-
	7 Hours/	20 Hours/	6 Hours/	13 Hours/	-
	29 Minutes/	13 Minutes/	36 Minutes/	54 Minutes/	-
	15 Seconds/	43 Seconds/	31 Seconds/	15 Seconds/	-
	-	-	-	-	6 MicroSeconds/

rors which is defined as observing the condition of "$\|b_j^*\| < r_{\text{FAC}_{\min}} \times \text{GH}(\mathcal{L}_{[j,k]})$" for each lattice block of $\mathcal{L}_{[j,k]}$. These errors would be eliminated automatically in Our BKZ simulation (Algorithm 1) by using proposed emulation of LLL and updating-GSO (discussed in Section 3), together with accurate sampling of solution norm and coefficient vectors of w and z (see [31, 32]).

In fact, the concepts of "head concavity" (see [5]), "tail convexity", and "random manner of middle lattice blocks" are depend to each other closely; Shi Bai *et al.*, introduce sufficient tests in paper [5] which verify the head concavity in their BKZ simulation and reject this concavity in Chen-Nguyen's simulation. Since verifying the head concavity and tail convexity needs to some tests on "Experimental Running of Original BKZ algorithm" (specially with high success probability of bounding functions for enumeration), therefore the running time for these tests limits us to use only moderate size of lattice blocks (e.g., to the best of our knowledge, paper [5] just uses block sizes of $\beta \leq 60$). In other side, since our approach just focuses on "random manner of middle blocks" by counting the GSO violations errors, this is possible to perform all these tests just by simulation for high block sizes (e.g., $\beta > 250$).

Table 4 shows the average rate of GSO violations in "Our BKZ simulation" (Algorithm 1) and "Chen-Nguyen's BKZ simulation". For this test, the number of 11 random lattices in the sense of Goldstein and Mayer [42, 43] with rank of $n = 240$ are used. Also for running simulation tests, the moderate block size of $\beta_0 = 60$ is used. For the test block size of $\beta_0 = 60$, the index of Tup (see Remark 4) for these random lattice bases is 180, also the index of Hdown (see Remark 4) is set to 50 (based on the experiments in [5], together with some safety border considered in this paper). The parameter of p_{MIN} is defined as $\text{p}_{\text{MIN}} = \frac{1}{\text{Tup}-\text{Hdown}+1} \approx 0.00769$ in this test (see Theorem 3 in Section 2.6).

5 Conclusions and Future Works

The BKZ algorithm has a main role in security analysis of lattice-based cryptography, therefore the total cost and quality of output basis by this algorithm should be determined exactly to be used in bit-security estimation and parameter selection of lattice cryptographic primitives. However the exact behavior of BKZ algorithm with small block sizes can be studied by Experimental Running of Original BKZ algorithm, this behavior for higher block sizes (e.g., $\beta \geq 100$) should be simulated (instead of Experimental Running of Original BKZ algorithm) to be time-tolerable. There are some main studies in this scope which introduce different simulations of BKZ, such as: BKZ simulation by Chen and Nguyen [2], BKZ simulation by Aono *et al.* [3] and BKZ simulation by Shi Bai *et al.* [5].

Albrecht *et al.* by using LWE-estimator from [20] estimate the cost of primal and dual lattice attacks against LWE-based schemes and primal attacks against NTRU-based schemes using some proposed cost models for BKZ. The cost model is a way to show the total nodes of processing in SVP-Solvers (e.g., GNR-enumeration, sieving algorithm, discrete pruning, etc.); Definition of an exact formula for the cost model of BKZ algorithm in achieving a certain value of root-Hermite factor, simplifies the security analysis of lattice based cryptographic primitives. Unfortunately, this paper claims that these cost models are not necessarily exact enough for lattice security analysis, since as shown in paper [22], some typical cryptographic scheme "A" under one cost model may be considered harder (to be broken) than another typical cryptographic scheme "B", while scheme "A" under another cost model may appear weaker (to be broken) than scheme "B". For example, the cost of dual attack against "`Titanium.PKE-2048-1.41-1198081`" (with claimed bit-security of 256) by using enumeration with four different cost models in Table 10 from [22] is estimated as 2^{595}, 2^{652}, 2^{1096} and 2^{1474}. Accordingly,

Table 4. Average rate of GSO violations in Chen-Nguyen's BKZ simulation and our BKZ simulation over 11 random lattice bases

Round#	GSO Errors in Chen-Nguyen Sim.	Rate of GSO Errors in Chen-Nguyen Sim.	GSO Errors in Our Sim. (Algorithm 1)	Rate of GSO Errors in Our Sim. (Algorithm 1)
Round 1	20.64	15.8%	0	0%
Round 2	38.64	29.5%	0	0%
Round 3	60.27	46.0%	0	0%
Round 4	71.45	54.5%	0	0%
Round 5	76.18	58.2%	0	0%
Round 6	78	59.5%	0	0%
Round 7	79	60.3%	0	0%
Round 8	79	60.3%	0	0%
Round 9	79	60.3%	0	0%
Round 10	79	60.3%	0	0%
Round 11	79	60.3%	0	0%
Round 12	79	60.3%	0	0%
Round 13	79	60.3%	0	0%
Round 14	79	60.3%	0	0%
Round 15	79	60.3%	0	0%
Round 16	79	60.3%	0	0%
Round 17	79	60.3%	0	0%

how security analyser can use such these bit-security estimations to compare the efficiency and security of different lattice cryptographic primitives; Also how the cryptographic designers can choose some efficient/secure parameter sets for their cryptographic primitives by using such these non-exact bit-security estimations. To solve this problem of non-exact bit-security estimations in current LWE/NTRU-based schemes against primal and dual lattice attacks, this paper design an "Accurate BKZ Simulation" which is expected to make the estimations of total cost of lattice attacks more precise, for reaching to a specified value of root-Hermite factor (or a specified degree of basis quality measure).

Our contributions in this paper include two provable tools of "Emulation of updating GSO norms/coefficients" (see Lemma 1 and Lemma 2) and "Emulation of LLL function" (see Lemma 4 and Lemma 6) as two main parts in designing an accurate BKZ simulation. In fact, this paper proves that for a typical SVP solver "Z" (e.g., GNR-enumeration, Sieving, discrete pruning, etc.), if there is a simulation of "Z_emulate" which provably emulates the behaviour of practical running of "Z", then Our BKZ Simulation (Algorithm 1) by using "emulate_SVPSolver"="Z_emulate" can provably emulates the Experimental Running of BKZ algorithm (Algorithm 6) using SVP solver "Z". By using our former contributions and achievements in [29–32] which try to simulate the behaviour of GNR-enumeration (as a SVP-Solver of BKZ) with better accuracy, this paper assembles our two contributions (in this paper), together with our former contributions in [29–32] (as our simulation of GNR-enumeration), and introduces a claimant accurate BKZ simulation than before. More precisely, our accurate BKZ simulation can lead to more exact bit-security estimations (and more efficient/secure parameter set) for lattice-based schemes (e.g., [23–28, 33]) against any lattice attacks which use lattice reductions. However there are more applications which can be counted for using an accurate BKZ simulation. The use of our two contributions of "Emulation of updating GSO norms/coefficients" and "Emulation of LLL function" in designing our accurate BKZ simulation, introduce following privileges over former proposed BKZ simulations [2, 3, 5]:

- Against former BKZ simulations which only get the "GSO norms" of basis vectors as input/output parameters, our BKZ simulation uses "GSO norms" together with "GSO coefficients" as input/output, and this makes the opportunity of taking more information about the the output basis of BKZ simulation, such as approximation of Euclidean norms of output basis vectors by using Equation (9), which may be used in some lattice attacks in further studies.
- Chen-Nguyen's BKZ simulation [2] uses an error-prone strategy for updating GSO norms/coefficients, which is noted in [5] too, but this problem is solved in our BKZ simulation.
- Also paper [3] does not explicitly introduce a process of updating GSO for its BKZ simulation under Heuristic 2 and Heuristic 3 from [34], rather it just analyses its simulation under assumption of GSA, while (to the best of our knowledge) the assumption of GSA is known as a non-exact approximate prediction only for middle lattice blocks, but this weakness is solved in our BKZ simulation.
- Moreover in Lemma 3, this is proved that the process of updating GSO in [5] is not hold under Heuristic 2 and Heuristic 3 from [34] too, even this paper claims that paper [5] nearly just tries to hold the assumption of GSA to simulate updating GSO norms/coefficients, but this weakness is solved in our BKZ simulation.
- By using just "Emulation of updating GSO norms/coefficients" and ignoring "Emulation

of LLL function" after each enumeration over current lattice block, the correctness of BKZ simulation is not violated, however this paper shows that the shape of next block is changed into a worse quality, so the enumeration cost over the next block increases in an unpleasant way (to the best of our knowledge, the emulation of LLL algorithm is not at all considered in former corresponding studies [2, 3, 5]). This problem is solved in our BKZ simulation.
- Another main reason for using "Emulation of LLL function" after each enumeration success in design of our BKZ simulation is that an overflow error in calculations over real numbers can be observed (especially in the case of using float point precision), while this error is tensed for bigger block sizes (to the best of our knowledge, the emulation of LLL algorithm is not at all considered in papers of [2, 3, 5]). This problem is solved in our BKZ simulation.
- Moreover, Our BKZ Simulation is fully compatible with our former contributions which simulate the behavior of GNR-enumeration (as a SVP-Solver of BKZ) with better accuracy (include: "Revised Estimations for Cost and Success Probability of GNR-Enumeration" [29], "Optimal bounding function for GNR-enumeration" [30], "Better Sampling Method of Enumeration Solution for BKZ Simulation" [31] and "Revised Method for Sampling Coefficient Vector of GNR-enumeration Solution" [32]). However, Our BKZ Simulation can use any other SVP-Solver of BKZ (instead of GNR-Enumeration) too.

Our test results in this paper verify the value of our contributions in achieving more accuracy of BKZ simulation (and consequently in better accuracy of lattice bit-security estimations). Briefly, our test results include following cases:

- Our test results show that, altogether the shape of GSO norms of $\|b_i^*\|^2$ in "Experimental Running of Original BKZ algorithm" is more similar (and close) to the shape of $\|b_i^*\|^2$ in "Our BKZ Simulation", than to the GSO norms of $\|b_i^*\|^2$ in "Chen-Nguyen's BKZ simulation" and "BKZ simulation by Shi Bai *et al.*".
- Also, our test results show that, altogether the root-Hermite factor of basis after Experimental Running of Original BKZ algorithm is nearly close to the root-Hermite factor of basis after applying "Our BKZ Simulation".
- Also our test results show that, altogether the total cost (and running time) in "Experimental Running of Original BKZ algorithm" is nearly more similar (and close) to the total cost (and running time) in "Our BKZ Simulation", than to the total cost (and running time) in "Chen-Nguyen's BKZ simulation", "BKZ Simulation by Shi Bai *et al.*" and some other BKZ models in our test.
- Moreover, Chen-Nguyen's BKZ simulation uses $\text{GH}(\mathcal{L}_{[j,k]})$ for expecting the GSO norms of full-enumeration's solution vectors, but wrong strategy of updating GSO norms/coefficients in Chen-Nguyen's BKZ simulation leads to many GSO violation errors for lattice blocks, while our final test results (in this paper) verify that whole these errors would be eliminated automatically in our BKZ simulation.

However our test results use our accurate BKZ simulation (Algorithm 1) with use of Algorithm 2 (our simulation of GNR-enumeration) called in line 6 from Algorithm 1 over lattice blocks, this BKZ simulation can be modified for using other SVP-solver's simulations (or emulations). This is clear that, only the skeleton of BKZ algorithm together with our contributions in this paper (i.e., "Emulation of updating GSO norms/coefficients" and "Emulation of LLL function") would not be modified for using other SVP solver's simulations. The simulations of different SVP solvers can be used in our BKZ simulation, such as:

- Lattice enumeration by discrete pruning technique (see [45] and some simulation variant in [46]),
- Sieving algorithm for solving SVP (such as by variants in [47, 48]),
- The novel idea for enumeration by integrating sparse orthogonalized integer representations for shortest vectors [49],
- Evolutionary searches for solving SVP (see [50, 51]).

Some main further studies, which can use our contributions of this paper, are counted as follows. Assembling and configuration of a full-parameterized/flexible BKZ simulation to simulate every form of the BKZ algorithm, such as: the original Schnorr-Euchner's BKZ algorithm [52] (see Algorithm 5 from Appendix A.2), progressive-BKZ with increasing block size (see [2, 3, 36, 53]), progressive-BKZ with increasing success probability (see [36, 53]), pressed-BKZ algorithm [5], BKZ 2.0 algorithm [2], revised-version of BKZ 2.0 algorithm [54], Pump and jump-BKZ (pnj-BKZ) algorithm [15], Improved Progressive Pnj-BKZ (ProPnjBKZ) algorithm [17], Parallelized BKZ variants or BKZ with parallelized SVP-Solvers (especially in total cost estimation of BKZ) [55] and even our proposed software technique to speed up the BKZ algorithm [56]. Also, our accurate BKZ simulation can be used in revising the

estimations of total cost of applying lattice attacks for reaching to a specified value of root-Hermite factor or a specified basis quality measure, and consequently it can lead to more exact bit-security estimations and more efficient/secure parameter selections for current LWE/NTRU-based schemes (e.g., [23–28]) against primal and dual lattice attacks. Moreover, our BKZ simulation can be used as a smaller part of other lattice attacks, such as simulating the shape of GSO norms of $\|b_i^*\|^2$, root-Hermite factor and estimated total nodes (cost) in running of the BKZ algorithm in the attack of "hybrid lattice-reduction and meet-in-the-middle attack against NTRU" [57].

Acknowledgment

The authors would like to thank the anonymous reviewers for their much valuable, constructive and insightful comments which helped us to improve the presentation of this work significantly, also thank the editorial board and editorial staff of the ISeCure journal for their generous help in publishing this study.

References

[1] Anyu Wang, Dianyan Xiao, and Yang Yu. Lattice-based cryptosystems in standardisation processes: A survey. *IET Information Security*, 17(2):227–243, 2023.

[2] Yuanmi Chen and Phong Q Nguyen. Bkz 2.0: Better lattice security estimates. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 1–20. Springer, 2011.

[3] Yoshinori Aono, Yuntao Wang, Takuya Hayashi, and Tsuyoshi Takagi. Improved progressive bkz algorithms and their precise cost estimation by sharp simulator. In *Advances in Cryptology–EUROCRYPT 2016: 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part I 35*, pages 789–819. Springer, 2016.

[4] Jianwei Li and Phong Q Nguyen. A complete analysis of the bkz lattice reduction algorithm. *Cryptology ePrint Archive*, 2020.

[5] Shi Bai, Damien Stehlé, and Weiqiang Wen. Measuring, simulating and exploiting the head concavity phenomenon in bkz. In *Advances in Cryptology–ASIACRYPT 2018: 24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, December 2–6, 2018, Proceedings, Part I 24*, pages 369–404. Springer, 2018.

[6] Tanja Lange Daniel J. Bernstein, Chitchanok Chuengsatiansup and Christine van Vredendaal. "ntru prime". Technical report, National Institute of Standards and Technology, 2017. URL https://csrc.nist.gov/Projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-1-submissions. Submission at Round 1.

[7] Jeff Hoffstein, Jill Pipher, John M Schanck, Joseph H Silverman, William Whyte, and Zhenfei Zhang. Choosing parameters for ntruencrypt. In *Cryptographers' Track at the RSA Conference*, pages 3–18. Springer, 2017.

[8] Eamonn W Postlethwaite and Fernando Virdia. On the success probability of solving unique svp via bkz. In *IACR International Conference on Public-Key Cryptography*, pages 68–98. Springer, 2021.

[9] Yuntao Wang, Yoshinori Aono, and Tsuyoshi Takagi. Hardness evaluation for search lwe problem using progressive bkz simulator. *IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences*, 101(12):2162–2170, 2018.

[10] Joop van de Pol and Nigel P Smart. Estimating key sizes for high dimensional lattice-based systems. In *IMA International Conference on Cryptography and Coding*, pages 290–303. Springer, 2013.

[11] Dana Dachman-Soled, Léo Ducas, Huijing Gong, and Mélissa Rossi. Lwe with side information: attacks and concrete security estimation. In *Annual International Cryptology Conference*, pages 329–358. Springer, 2020.

[12] Martin R Albrecht, Shi Bai, Pierre-Alain Fouque, Paul Kirchner, Damien Stehlé, and Weiqiang Wen. Faster enumeration-based lattice reduction: root hermite factor time. In *Annual International Cryptology Conference*, pages 186–212. Springer, 2020.

[13] Martin R Albrecht, Shi Bai, Jianwei Li, and Joe Rowell. Lattice reduction with approximate enumeration oracles: practical algorithms and concrete performance. In *Annual International Cryptology Conference*, pages 732–759. Springer, 2021.

[14] Qian Guo and Thomas Johansson. Faster dual lattice attacks for solving lwe with applications to crystals. In *Advances in Cryptology–ASIACRYPT 2021: 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6–10, 2021, Proceedings, Part IV 27*, pages 33–62. Springer, 2021.

[15] Leizhang Wang, Wenwen Xia, Geng Wang, Baocang Wang, and Dawu Gu. Improved pump and jump bkz by sharp simulator. *Cryptology ePrint*

Archive, 2022.

[16] Leizhang Wang, Yuntao Wang, and Baocang Wang. A trade-off svp-solving strategy based on a sharper pnj-bkz simulator. In *Proceedings of the 2023 ACM Asia Conference on Computer and Communications Security*, pages 664–677, 2023.

[17] Wenwen Xia, Leizhang Wang, Dawu Gu, Baocang Wang, *et al.* Improved progressive bkz with lattice sieving and a two-step mode for solving usvp. *Cryptology ePrint Archive*, 2022.

[18] Ziyu Zhao and Jintai Ding. Several improvements on bkz algorithm. *Cryptology ePrint Archive*, 2022.

[19] Zishen Zhao and Guangwu Xu. On the measurement and simulation of the bkz behavior for q-ary lattices. In *International Conference on Information Security and Cryptology*, pages 463–482. Springer, 2022.

[20] Martin R Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of learning with errors. *Journal of Mathematical Cryptology*, 9 (3):169–203, 2015.

[21] M. R. Albrecht and *et al.* Estimate all the {LWE, NTRU} schemes! online version. URL https://estimate-all-the-lwe-ntru-schemes.github.io/docs/. Accessed: 2024-8-12.

[22] 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! In *Security and Cryptography for Networks: 11th International Conference, SCN 2018, Amalfi, Italy, September 5–7, 2018, Proceedings 11*, pages 351–367. Springer, 2018.

[23] Erdem Alkim, Paulo SLM Barreto, Nina Bindel, Juliane Krämer, Patrick Longa, and Jefferson E Ricardini. The lattice-based digital signature scheme qtesla. In *International Conference on Applied Cryptography and Network Security*, pages 441–460. Springer, 2020.

[24] Javad Sharafi and Hassan Daghigh. A ring-lwe-based digital signature inspired by lindner–peikert scheme. *Journal of Mathematical Cryptology*, 16(1):205–214, 2022.

[25] D.J. Bernstein and *et al.* "ntru prime". Technical report, National Institute of Standards and Technology, 2020. URL https://csrc.nist.gov/Projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions. Submission at Round 3.

[26] Peter Schwabe, Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Gregor Seiler, and Damien Stehlé. "crystals-kyber". Technical report, National Institute of Standards and Technology, 2020. URL https://csrc.nist.gov/Projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions. Submission at Round 3.

[27] Chitchanok Chuengsatiansup, Thomas Prest, Damien Stehlé, Alexandre Wallet, and Keita Xagawa. Modfalcon: Compact signatures based on module-ntru lattices. In *Proceedings of the 15th ACM Asia Conference on Computer and Communications Security*, pages 853–866, 2020.

[28] Lyubashevsky and *et al.* "d.s.s.: Crystals-dilithium". Technical report, National Institute of Standards and Technology, 2020. URL https://csrc.nist.gov/Projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions. Submission at Round 3.

[29] AR Payandeh and GR Moghissi. Revised estimations for cost and success probability of gnr-enumeration. *Journal of Electrical and Computer Engineering Innovations (JECEI)*, 11(2): 459–480, 2023.

[30] Gholam Reza Moghissi and Ali Payandeh. Optimal bounding function for gnr-enumeration. *International Journal of Mathematical Sciences and Computing (IJMSC)*, 8(1):1–17, 2022.

[31] Gholam Reza Moghissi and Ali Payandeh. Better sampling method of enumeration solution for bkz-simulation. *ISeCure*, 13(2), 2021.

[32] Gholam Reza Moghissi and Ali Payandeh. Revised method for sampling coefficient vector of gnr-enumeration solution. *Int. J. Math. Sci. Comput.(IJMSC)*, 8(3):1–20, 2022.

[33] Reza Ebrahimi Atani, Shahabaddin Ebrahimi Atani, and Amir Hassani Karbasi. Eeh: Aggh-like public key cryptosystem over the eisenstein integers using polynomial representations. *ISeCure*, 7(2), 2015.

[34] Nicolas Gama, Phong Q Nguyen, and Oded Regev. Lattice enumeration using extreme pruning. In *Advances in Cryptology–EUROCRYPT 2010: 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, May 30–June 3, 2010. Proceedings 29*, pages 257–278. Springer, 2010.

[35] Daniele Micciancio and Oded Regev. Lattice-based cryptography. In *Post-quantum cryptography*, pages 147–191. Springer, 2009.

[36] Gholam Reza Moghissi and Ali Payandeh. Design of optimal progressive bkz with increasing success-probabilities and increasing block-sizes. *Journal of Computing and Security*, 9(2):65–93,

2022.

[37] Claus Peter Schnorr. Lattice reduction by random sampling and birthday methods. In *STACS 2003: 20th Annual Symposium on Theoretical Aspects of Computer Science Berlin, Germany, February 27–March 1, 2003 Proceedings 20*, pages 145–156. Springer, 2003.

[38] Arjen K Lenstra, Hendrik Willem Lenstra, and László Lovász. Factoring polynomials with rational coefficients. 1982.

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

[40] Johannes Buchmann and Christoph Ludwig. Practical lattice basis sampling reduction. In *International Algorithmic Number Theory Symposium*, pages 222–237. Springer, 2006.

[41] V. Shoup, "NTL: a library for doing number theory". Online. Available at: http://www.shoup.net/ntl/. Accessed: 2024-8-12.

[42] SVP Challenge. Online. Available at: https://www.latticechallenge.org/svp-challenge/index.php. Accessed: 2024-8-12.

[43] Daniel Goldstein and Andrew Mayer. On the equidistribution of hecke points. 2003.

[44] GitHub hosting service, "fplll library project". Online. Available at: https://github.com/fplll/fplll. Accessed: 2024-8-12.

[45] Yoshinori Aono and Phong Q Nguyen. Random sampling revisited: lattice enumeration with discrete pruning. In *Advances in Cryptology–EUROCRYPT 2017: 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30–May 4, 2017, Proceedings, Part II 36*, pages 65–102. Springer, 2017.

[46] Luan Luan, Chunxiang Gu, Yonghui Zheng, and Yanan Shi. Lattice enumeration with discrete pruning: Improvements, cost estimation and optimal parameters. *Mathematics*, 11(3):766, 2023.

[47] Léo Ducas. Shortest vector from lattice sieving: a few dimensions for free. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 125–145. Springer, 2018.

[48] Anja Becker, Léo Ducas, Nicolas Gama, and Thijs Laarhoven. New directions in nearest neighbor searching with applications to lattice sieving. In *Proceedings of the twenty-seventh annual ACM-SIAM symposium on Discrete algorithms*, pages 10–24. SIAM, 2016.

[49] Zhongxiang Zheng, Xiaoyun Wang, Guangwu Xu, and Yang Yu. Orthogonalized lattice enumeration for solving svp. *Science China Information Sciences*, 61:1–15, 2018.

[50] Gholam Reza Moghissi and Ali Payandeh. A parallel evolutionary search for shortest vector problem. *International Journal of Information Technology and Computer Science*, 2019.

[51] Luan Luan, Chunxiang Gu, and Yonghui Zheng. A genetic algorithm with restart strategy for solving approximate shortest vector problem. In *2020 12th International Conference on Advanced Computational Intelligence (ICACI)*, pages 243–250. IEEE, 2020.

[52] Joop van de Pol. Lattice-based cryptography. *Eindhoven University of Technology, Department of Mathematics and Computer Science*, 2011.

[53] Gholam Reza Moghissi and Ali Payandeh. Using progressive success probabilities for sound-pruned enumerations in bkz algorithm. *International Journal of Computer Network and Information Security*, 11(9):10, 2018.

[54] Gholam Reza Moghissi and Ali Payandeh. Rejecting claimed speedup of $2^{\beta/2}$ in extreme pruning and revising bkz 2.0 for better speedup. *Journal of Computing and Security*, 8(1):65–91, 2021.

[55] Nariaki TATEIWA. *Development and Numerical Experiments of Massively Parallel Framework and Software for Shortest Vector Problem*. PhD thesis, Graduate School of Mathematics, KYUSHU UNIVERSITY. January 17, 2022.

[56] Gholam Reza Moghissi and Ali Payandeh. A software technique to speed up bkz implementations. In *3rd International Conference on Electrical Engineering*, 2018. URL https://civilica.com/doc/831793.

[57] Nick Howgrave-Graham. A hybrid lattice-reduction and meet-in-the-middle attack against ntru. In *Advances in Cryptology-CRYPTO 2007: 27th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2007. Proceedings 27*, pages 150–169. Springer, 2007.

Gholam Reza Moghissi received the M.Sc. degree in Department of ICT at Malek-e-Ashtar University of Technology, Tehran, Iran, in 2016. His researches focus on Information Security.

Ali Payandeh received the M.Sc. degree in Electrical Engineering from Tarbiat Modares University in 1994, and the Ph.D. degree in Electrical Engineering from K.N. Toosi University of Technology (Tehran, Iran) in 2006. He is now an assistant professor in the Department of Information and Communications Technology at Malek-e-Ashtar University of

Technology, Iran. His research interests include Information Theory, Coding Theory, Cryptography, Security Protocols, Secure Communications, and Satellite Communications.

Appendices

A. Pseudo-Codes of Algorithms in This Paper

In this appendix, the pseudo-codes of essential algorithms in this paper would be introduced.

A.1 The LLL Algorithm

Algorithm 3 shows the pseudo-code of full-LLL function [52].

Algorithm 3 LLL algorithm

Input: input basis $B = (b_1, \ldots, b_n) \in \mathbb{Z}^{n \times m}$, GSO Coef Matrix μ, $\frac{1}{4} \leq \delta < 1$.
Output: output basis B, GSO Coef Matrix μ.

1: **for** $(i = 1$ *to* $n - 1)$ **do**
  //Reduction Step (Size-reduction):
2: **for** $(l = i$ *downto* $1)$ **do**
3: $b_{i+1} \leftarrow b_{i+1} - \lfloor \mu_{i+1,l} \rceil b_l$;
4: **for** $(t = 1$ *to* $l)$ **do**
5: $\mu_{i+1,t} \leftarrow \mu_{i+1,t} - \lfloor \mu_{i+1,l} \rceil \mu_{l,t}$;
6: **end for**
7: **end for**
  //Swap Step (Lovasz criterion):
8: **if** $(\delta \times \|b_i^*\|^2) > \|\mu_{i+1,i} \times b_i^* + b_{i+1}^*\|^2$ **then**
9: swap(b_i, b_{i+1}); $i - -$;
10: **else**
11: $i + +$;
12: **end if**
13: **end for**

Note that the decreasing order of index from $l = i$ *downto* 1, in line 2 from Algorithm 3 is necessary for correctness of this algorithm. Also, the pseudo-code of partial-LLL (a modified version of full-LLL) as a main part of original version of BKZ can be studied in Algorithm 4.

A.2 Schnorr-Euchner's BKZ Algorithm

Algorithm 5 shows the pseudo-code of original version of BKZ algorithm [52].

Algorithm 6 shows the pseudo-code of a generalized version of original BKZ algorithm which can be aborted after rounds roundnum, also it can use any form of SVP-Solvers (e.g., GNR-enumeration, Sieving, discrete pruning);

A.3 Emulation of "Updating GSO Norms/Coefficients" and "LLL Function"

Algorithm 7 shows the pseudo-code of our method for emulation of updating GSO in our BKZ simulation

Algorithm 4 Partial version of LLL algorithm

Input: input basis $B' = (b_1, \ldots, b_{end}) \in \mathbb{Z}^{end \times m}$, GSO Coef Matrix μ, $\frac{1}{4} \leq \delta < 1$, stage of *start*.
Output: output basis B', GSO Coef Matrix μ.

1: **for** $(i = \max(start - 1, 1); i \leq end - 1;)$ **do**
  //Reduction Step:
2: **for** $(l = i$ *downto* $1)$ **do**
3: $b_{i+1} \leftarrow b_{i+1} - \lfloor \mu_{i+1,l} \rceil b_l$;
4: **for** $(t = 1$ *to* $l)$ **do**
5: $\mu_{i+1,t} \leftarrow \mu_{i+1,t} - \lfloor \mu_{i+1,l} \rceil \mu_{l,t}$;
6: **end for**
7: **end for**
  //Swap Step:
8: **if** $((\delta \times \|b_i^*\|^2) > \|\mu_{i+1,i} \times b_i^* + b_{i+1}^*\|^2)$ **then**
9: swap(b_i, b_{i+1}); $i - -$;
10: **else**
11: $i + +$;
12: **end if**
13: **end for**

Algorithm 5 Block Korkin-Zolotarev (BKZ)

Input: input basis $B = (b_1, \ldots, b_n) \in \mathbb{Z}^{n \times m}$, GSO Coef Matrix μ, $2 \leq \beta \leq n$, $1/4 \leq \delta < 1$.
Output: BKZ$_\beta$ reduced basis B.

1: LLL(B, μ, δ); //LLL reduce the basis and update μ
2: **for** $(z = 0, j = 0; z < n - 1;)$ **do**
3: $j = (j \mod (n - 1)) + 1$;
4: $k = \min(j + \beta - 1, n)$;
5: $h = \min(k + 1, n)$;
6: $y \leftarrow \text{Enum}(\|b_j^*\|^2, \|b_{j+1}^*\|^2, \ldots, \|b_k^*\|^2, \mu_{[j,k]})$;
7: **if** $(y \neq (1, 0, \ldots, 0))$ **then**
8: LLL$([b_1, \ldots, b_{j-1}, \sum_{l=j}^{k} y_l b_l, b_j, \ldots, b_h], \mu, \delta)$ at stage j;
9: $z = 0$;
10: **else**
11: LLL $([b_1, \ldots, b_h], \mu, \delta)$ at stage $h - 1$;
12: $z + +$;
13: **end if**
14: **end for**

Algorithm 6 Aborted version of BKZ algorithm with any SVP-Solver

Input: input basis $B = (b_1, \ldots, b_n) \in \mathbb{Z}^{n \times m}$, GSO Coef Matrix μ, $2 \leq \beta \leq n$, $1/4 \leq \delta < 1$, SVPSolver's input_parameters of *param*, number of rounds ROUND$_{\text{NUM}}$.
Output: BKZ$_\beta$ reduced basis B after number of rounds of ROUND$_{\text{NUM}}$.

1: LLL(B, μ, δ); //LLL reduce the basis and update μ
2: **for** $(l = 1, 2, \ldots,$ ROUND$_{\text{NUM}})$ **do**
3: **for** $(j = 1, 2, \ldots, n - 2)$ **do**
4: $k = \min(j + \beta - 1, n)$;
5: $h = \min(k + 1, n)$;
6: $y \leftarrow$ SVPSolver$(\|b_j^*\|^2, \|b_{j+1}^*\|^2, \ldots, \|b_k^*\|^2, \mu_{[j,k]}, param)$;
7: **if** $(y \neq (1, 0, \ldots, 0))$ **then**
8: LLL$([b_1, \ldots, b_{j-1}, \sum_{l=j}^{k} y_l b_l, b_j, \ldots, b_h], \mu, \delta)$ at stage j;
9: **else**
10: LLL $([b_1, \ldots, b_h], \mu, \delta)$ at stage $h - 1$;
11: **end if**
12: **end for**
13: **end for**

Algorithm 7 Emulation of updating GSO (emulate_UpdateGSO)

Input: GSO norms for basis $B^*_{[1,n]} = [\|b_1^*\|, \ldots, \|b_n^*\|]$, start index j, end index f, GSO Coef Matrix μ, index g, sampled norm of solution as ℓ_{new} /*$\ell_{new} = \|\pi_j(v)\| = \|v\|$*/, coefficient vector w.

Output: GSO norms of basis $B^*_{[1,n]} = [\|b_1^*\|, \ldots, \|b_n^*\|]$ and updated coefficient matrix μ.

1: **for**$(1 \le l < \ell < j)$ $\{\mu''_{\ell,l} = \mu_{\ell,l};\}$
2: **for**$(1 \le l < j = \ell)$ $\{\mu''_{\ell,l} = \sum_{i=1}^{g} y_i \mu_{j+i-1,l};\}$
3: **for**$(1 \le l < j < \ell \le j+g-1)$ $\{\mu''_{\ell,l} = \mu_{\ell-1,l};\}$
4: **for**$(1 \le l < j$ *and* $j+g \le \ell)$ $\{\mu''_{\ell,l} = \mu_{\ell,l};\}$
5: **for**$(j = l < \ell \le j+g-1)$ $\mu''_{\ell,l} = \frac{\sum_{t=j}^{\ell-1} \mu_{\ell-1,t} w_{t-j+1} \|b_t^*\|^2}{\|\pi_j(v)\|^2}$;
6: **for**$(j = l < j+g \le \ell)$ $\{\mu''_{\ell,l} = \frac{\sum_{t=j}^{j+g-1} \mu_{\ell,t} w_{t-j+1} \|b_t^*\|^2}{\|\pi_j(v)\|^2}$;$\}$
7: **for**$(j < l < \ell \le j+g-1)\{$
$\mu''_{\ell,l} = \mu_{\ell-1,l-1} - \frac{w_{l-j} \times \left(\sum_{t=l}^{\ell-1} \mu_{\ell-1,t} w_{t-j+1} \|b_t^*\|^2\right)}{\|\pi_l(v)\|^2}$;$\}$
8: **for**$(j < l < j+g \le \ell)\{$
$\mu''_{\ell,l} = \mu_{\ell,l-1} - \frac{w_{l-j} \times \left(\sum_{t=l}^{j+g-1} \mu_{\ell,t} w_{t-j+1} \|b_t^*\|^2\right)}{\|\pi_l(v)\|^2}$;$\}$
9: **for**$(j+g \le l < \ell)$ $\{\mu''_{\ell,l} = \mu_{\ell,l};\}$
10: **for**$(l = \ell)$ $\{\mu''_{\ell,l} = 1;\}$
11: **for**$(\ell < l)$ $\{\mu''_{\ell,l} = 0;\}$
12: $B^{*'}_{[1,d]} \leftarrow [\|b_j^*\|, \ldots, \|b_f^*\|]$;
13: $B^{*'}_{[1,d]}$.delete(g);
14: $B^{*'}_{[1,d]}$.insert$(1, \ell_{new})$;
//The result of lines 12, 13, 14:
//$B^{*'}_{[1,d]} \leftarrow [\|v\|, \|b_1^{*'}\|, \ldots, \|b_{g-1}^{*'}\|, \|b_{g+1}^{*'}\|, \ldots, \|b_d^{*'}\|]$
15: $w \leftarrow [0, w_1, w_2, \ldots, w_{g-1}, 0, 0, 0]$;
//The index of w are shifted to right:
//$w_1, \ldots, w_{g-1}$ are shifted into $w_2, \ldots, w_g$
16: **for**$(i = 1, \ldots, g)$ { /*see Equation (33)*/
$\|b_{i\,new}^{*'}\| \leftarrow \|b_i^{*'}\| \times \sqrt{1 - \frac{w_i^2 \|b_i^{*'}\|^2}{\ell_{new}^2 - \sum_{l=1}^{i-1} w_l^2 \|b_l^{*'}\|^2}}$;$\}$
/*Note: Only index of $i = 1 \ldots g$ will be modified and the remains are kept unchanged*/
17: **for**$(i = 1, \ldots, g)$ $\{\|b^*_{j+i-1}\| \leftarrow \|b_{i\,new}^{*'}\|;\}$
//$\|b^*_{j+g}\|$ to $\|b^*_{j+d-1}\|$ was not modified in $B^*_{[1,n]}$.
18: $\mu \leftarrow \mu''$; //update coefficient matrix μ

as follows.

Also we introduce the pseudo-code of our emulation of partial-LLL in Algorithm 8 (which represents full-LLL and partial-LLL).

B. Proof of Lemmas and Theorems

In this appendix, our proposed lemmas and theorems which are introduced in this paper would be proved (however some small proofs would be included in the main text of paper).

B.1 Proof of Lemma 1

Since the solution vector v is generated (found) by linear combination of vectors $b_1, \ldots, b_g$, so by using relation Equation (25):

Algorithm 8 Emulation of Partial-LLL (emulate_LLL)

Input: GSO norms of $[\|b_1^*\|, \ldots, \|b_{end}^*\|]$, GSO Coef Matrix μ, LLL parameter of $\delta \approx 1$, stage of "START".

Output: Partial GSO norms $\|b_1^*\|, \ldots, \|b_{end}^*\|$ and matrix μ.

1: $i = \max(\text{START} - 1, 1)$;
2: **while**$(i \le end - 1)$
//Reduction Step:
3: **for**$(l = i$ *downto* $1)$
4: **for** $(t = 1$ *to* $l)$
5: $\mu_{i+1,t} \leftarrow \mu_{i+1,t} - \lfloor \mu_{i+1,l} \rceil \mu_{l,t}$;
6: **end for**
7: **end for**
//Swap Step:
8: **if**$((\delta \times \|b_i^*\|^2) > \mu_{i+1,i}^2 \|b_i^*\|^2 + \|b_{i+1}^*\|^2)$
//start of Lovasz if
9: $\mu''_{i+1,i} = \frac{\mu_{i+1,i} \|b_i^*\|^2}{\|b_{i+1}^*\|^2 + \mu_{i+1,i}^2 \|b_i^*\|^2}$; //see Equation (39)
10: **for**$(l = i + 2$ *to* $n)$
11: $\mu''_{l,i} = \frac{\mu_{l,i+1} \|b_{i+1}^*\|^2 + \mu_{i+1,i} \mu_{l,i} \|b_i^*\|^2}{\|b_{i+1}^*\|^2 + \mu_{i+1,i}^2 \|b_i^*\|^2}$; //see Equation (40)
12: $\mu''_{l,i+1} = \mu_{l,i} - \mu_{l,i+1} \times \mu_{i+1,i}$;//see Equation (41)
13: $\mu_{l,i+1} \leftarrow \mu''_{l,i+1}$; $\mu_{l,i} \leftarrow \mu''_{l,i}$;
14: **end for**
15: **for**$(t = 1$ *to* $i - 1)$ //$\mu_{i,i}$ and $\mu_{i+1,i+1}$ remain 1
swap_entry$(\mu_{i,t}, \mu_{i+1,t})$;
16: **end for**
17: $\|b_i^*\|'' = \sqrt{\|b_{i+1}^*\|^2 + \mu_{i+1,i}^2 \|b_i^*\|^2}$; //see Equation (37)
18: $\|b_{i+1}^*\|'' = \frac{\|b_i^*\| \times \|b_{i+1}^*\|}{\sqrt{\|b_{i+1}^*\|^2 + \mu_{i+1,i}^2 \|b_i^*\|^2}}$; //see Equation (37)
19: $\mu_{i+1,i} \leftarrow \mu''_{i+1,i}$.
20: $\|b_i^*\| \leftarrow \|b_i^*\|''$; $\|b_{i+1}^*\| \leftarrow \|b_{i+1}^*\|''$; $i - -$;
//end of Lovasz if
21: **else**
22: $i + +$;
23: **end if**
24: **end while**

$$\text{Span}(v, b_1, \ldots, b_g)^\perp = \text{Span}(b_1, \ldots, b_g, v)^\perp = \text{Span}(b_1^*, \ldots, b_{g-1}^*, \pi_g(v) = b_g^*, b_g^*)^\perp = \text{Span}(b_1^*, \ldots, b_{g-1}^*, b_g^*)^\perp \quad (45)$$

The solution vector v which is inserted just before the place of vector b_i in the lattice block, makes the GSO projected form of vector b_i (and next vectors of that block) as the same as the GSO projected form of vector b_i after inserting vector v at the beginning of lattice block, as follows (by using Equation (45)):

$$\mathcal{L}_{[1,d+1]_{new}} = (v, b_{1\,new}^*, b_{2\,new}^*, \ldots, b_{i\,new}^*, \ldots, b_{g-1\,new}^*, b_{g+1}^*, \ldots, b_d^*) = (b_1^*, b_2^*, \ldots, \pi_i(v), b_{i\,new}^*, \ldots, b_{g-1\,new}^*, b_{g+1}^*, \ldots, b_d^*) \quad (46)$$

By using Equation (46), here are two ways to prove the relation Equation (33) in Lemma 1, as follows.

Proof 1: As shown in Figure 6, the GSO norm $\|b_i^*\|$

would be updated after inserting the solution vector v just before the place of vector b_i in the lattice block (and other vectors of the block can be updated in the same way):

$$\|b^*_{i\,new}\| = \|b^*_i\|\sin\theta = \|b^*_i\|\sqrt{1-\cos\theta^2} \Rightarrow$$
$$\|b^*_{i\,new}\| = \|b^*_i\|\sqrt{1-\frac{w_i^2\|b^*_i\|^2}{\|\pi_i(v)\|^2}} = \|b^*_i\|\sqrt{1-\frac{z^2_{d-i+1}}{\|\pi_i(v)\|^2}}$$

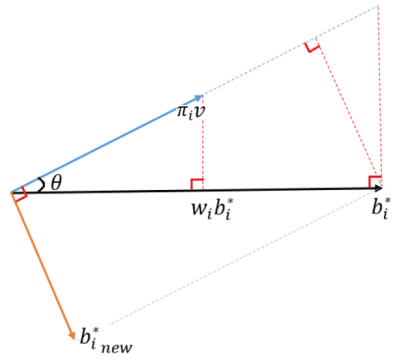


Figure 6. Updating each GSO vector b^*_i in lattice blocks after inserting enumeration solution v at the first of block

Proof 2: By using Equation (46) and Equation (8), the other way to prove Equation (33) can be drawn as follows:

$$b^*_{i\,new} = b_i - \sum_{t=1}^{i-1}\mu_{i,t}b^*_t - \frac{b_i\cdot\pi_i(v)}{\|\pi_i(v)\|^2}\pi_i(v) =$$
$$b^*_i - \frac{b_i\cdot\pi_i(v)}{\|\pi_i(v)\|^2}\pi_i(v) = b^*_i - \frac{b_i\cdot(w_ib^*_i+\cdots+w_gb^*_g)}{\|\pi_i(v)\|^2}\pi_i(v) =$$
$$b^*_i - \frac{(b^*_i+\sum_{t=1}^{i-1}\mu_{i,t}b^*_t)\cdot(w_ib^*_i+\cdots+w_gb^*_g)}{\|\pi_i(v)\|^2}\pi_i(v) =$$
$$b^*_i - \frac{w_ib^*_i\cdot b^*_i}{\|\pi_i(v)\|^2}\pi_i(v) =$$
$$b^*_i - \frac{w_ib^*_i\cdot b^*_i}{\|\pi_i(v)\|^2}(w_ib^*_i+\cdots+w_gb^*_g) \Rightarrow$$

$$b^*_{i\,new} = (1-\frac{w_i^2\|b^*_i\|^2}{\|\pi_i(v)\|^2})b^*_i - \frac{w_i\|b^*_i\|^2}{\|\pi_i(v)\|^2}(w_{i+1}b^*_{i+1}+\cdots+w_gb^*_g), \quad (47)$$
$$where\ 1\le i\le g-1$$

The relation Equation (47) at indices of $j,\ldots,j+g-1$ would be changed into relation Equation (48) by setting $l=i+1$:

$$b^*_{l\,new} = (1-\frac{w_{l-j}^2\|b^*_{l-1}\|^2}{\|\pi_{l-1}(v)\|^2})b^*_{l-1} - \frac{w_{l-j}\|b^*_{l-1}\|^2}{\|\pi_{l-1}(v)\|^2}(w_{l-j+1}b^*_l+\cdots+w_gb^*_{j+g-1}),\ where\ 2\le l\le g \quad (48)$$

By using Equation (47):

$$\|b^*_{i\,new}\|^2 =$$
$$(1-\frac{w_i^2\|b^*_i\|^2}{\|\pi_i(v)\|^2})^2\|b^*_i\|^2+\frac{w_i^2\|b^*_i\|^4}{\|\pi_i(v)\|^4}(w_{i+1}^2\|b^*_{i+1}\|^2+\cdots+w_g^2\|b^*_g\|^2) =$$
$$(1-\frac{w_i^2\|b^*_i\|^2}{\|\pi_i(v)\|^2})^2\|b^*_i\|^2+\frac{w_i^2\|b^*_i\|^4(\|\pi_i(v)\|^2-w_i^2\|b^*_i\|^2)}{\|\pi_i(v)\|^4} =$$
$$\|b^*_i\|^2-\frac{2w_i^2\|b^*_i\|^4}{\|\pi_i(v)\|^2}+\frac{w_i^4\|b^*_i\|^6}{\|\pi_i(v)\|^4}+\frac{w_i^2\|b^*_i\|^4}{\|\pi_i(v)\|^2}-\frac{w_i^4\|b^*_i\|^6}{\|\pi_i(v)\|^4} =$$
$$\|b^*_i\|^2-\frac{w_i^2\|b^*_i\|^4}{\|\pi_i(v)\|^2} \Rightarrow$$
$$\|b^*_{i\,new}\| = \|b^*_i\|\sqrt{1-\frac{w_i^2\|b^*_i\|^2}{\|\pi_i(v)\|^2}}$$

Besides these two ways to prove the relation Equation (33) in Lemma 1, this can be shown that the determinant of GSO block $(b^*_1,b^*_2,\ldots,b^*_g,\ldots,b^*_d)$ would be preserved by using Equation (46), as follows.

State 1: The determinant of block is preserved when solution vector v would be inserted after the place of vector b_{g-1}:

$$\det(\mathcal{L}_{[1,d]})^2 = \|b^*_1\|^2\|b^*_2\|^2\cdots\|b^*_g\|^2\cdots\|b^*_d\|^2 =$$
$$\|b^*_1\|^2\|b^*_2\|^2\cdots\|b^*_{g-1}\|^2\times\underbrace{\pi_g(v)^2}_{=\|b^*_g\|^2}\times\|b^*_{g+1}\|^2\cdots\|b^*_d\|^2$$

State t: The determinant of block is preserved when solution vector v would be inserted after the place of vector b_{g-t} for $2<t<g$ as follows:

$$\det(\mathcal{L}_{[1,d]})^2 = \|b^*_1\|^2\|b^*_2\|^2\cdots\|b^*_g\|^2\cdots\|b^*_d\|^2 =$$
$$\|b^*_1\|^2\cdots\pi_{g-t+1}(v)^2\|b^*_{g-t+1_{new}}\|^2\cdots\|b^*_{g-1_{new}}\|^2\cdots\|b^*_d\|^2,$$
$$where\ \pi_{g-t+1}(v)^2 =$$
$$w_{g-t+1}^2\|b^*_{g-t+1}\|^2+\cdots+w_{g-1}^2\|b^*_{g-1}\|^2+\|b^*_g\|^2 \Rightarrow$$
$$\|b^*_{g-t+1}\|^2\cdots\|b^*_{g-1}\|^2\|b^*_g\|^2 =$$
$$\pi_{g-t+1}(v)^2\|b^*_{g-t+1_{new}}\|^2\cdots\|b^*_{g-1_{new}}\|^2$$

By using relation Equation (27) and relation Equation (33):

$$\|b^*_{g-t+1}\|^2\cdots\|b^*_{g-1}\|^2\|b^*_g\|^2 = \pi_{g-t+1}(v)^2\|b^*_{g-t+1}\|^2\cdot$$
$$(1-\frac{w_{g-t+1}^2\|b^*_{g-t+1}\|^2}{\pi_{g-t+1}(v)^2})\cdots\|b^*_{g-1}\|^2(1-\frac{w_{g-1}^2\|b^*_{g-1}\|^2}{\pi_{g-1}(v)^2}) \Rightarrow$$
$$\|b^*_g\|^2 = \pi_{g-t+1}(v)^2\times\left(1-\frac{w_{g-t+1}^2\|b^*_{g-t+1}\|^2}{\pi_{g-t+1}(v)^2}\right)\times$$
$$\left(1-\frac{w_{g-t+2}^2\|b^*_{g-t+2}\|^2}{\pi_{g-t+2}(v)^2}\right)\cdots\left(1-\frac{w_{g-1}^2\|b^*_{g-1}\|^2}{\pi_{g-1}(v)^2}\right) \Rightarrow$$
$$\frac{\|b^*_g\|^2}{\pi_{g-t+1}(v)^2} = (\frac{\pi_{g-t+2}(v)^2}{\pi_{g-t+1}(v)^2})(\frac{\pi_{g-t+3}(v)^2}{\pi_{g-t+2}(v)^2})\cdots(\frac{\|b^*_g\|^2}{\pi_{g-1}(v)^2}) = \frac{\|b^*_g\|^2}{\pi_{g-t+1}(v)^2}$$ □

B.2 Proof of Lemma 2

After inserting the solution vector v at the first of corresponding lattice block, the Euclidean norm of basis vectors don't changed. Just the GSO vectors of $b^*_j,\ldots,b^*_g$ in the corresponding block, and some GSO coefficients in the matrix μ should be modified. By using relation Equation (8), the projected vector $\pi_1(v)$ can be defined as follows.

Note: The notation $\mathcal{L}_{[1,d]}$ is needed to be changed into $\mathcal{L}_{[j,k]}$ in this proof.

Note: The notation v in this paper always refers to $\pi_j(v)$, not essentially $\pi_1(v)$, also the notation of $\mu_{v,i}$ refers to a temporary entry in GSO coefficient matrix μ between solution vector v and basis vector b_i.

$$\pi_j(v) = \pi_1(v) - \sum_{l=1}^{j-1}\mu_{v,l}b^*_l \Rightarrow$$
$$\pi_1(v) = \pi_j(v) + \sum_{l=1}^{j-1}\mu_{v,l}b^*_l$$

By using relation Equation (22):

$$\pi_j(v) = y_1\pi_j(b_j)+y_2\pi_j(b_{j+1})+\cdots+y_g\pi_j(b_{j+g-1}) \Rightarrow$$

$$\pi_1(v) = y_1 b_j + y_2 b_{j+1} + \cdots + y_g b_{j+g-1} = y_1(\pi_j(b_j) + \sum_{l=1}^{j-1}\mu_{j,l}b_l^*)+\cdots+y_g(\pi_j(b_{j+g-1})+\sum_{l=1}^{j-1}\mu_{j+g-1,l}b_l^*) = [y_1\pi_j(b_j)+\cdots+y_g\pi_j(b_{j+g-1})] + [y_1(\sum_{l=1}^{j-1}\mu_{j,l}b_l^*) + \cdots + y_g(\sum_{l=1}^{j-1}\mu_{j+g-1,l}b_l^*)] \Rightarrow$$

$$\pi_1(v) = [y_1\pi_j(b_j)+\cdots+y_g\pi_j(b_{j+g-1})] + [\sum_{l=1}^{j-1}(\sum_{i=1}^{g} y_i\mu_{j+i-1,l}) \times b_l^*] \quad (49)$$

This proof would be introduced in following 10 steps.

Step 1: Since the vectors of b_ℓ and b_l^* is not changed for $1 \le l < \ell < j$, so $\mu''_{\ell,l} = \mu_{\ell,l}$.

Step 2: For $1 \le l < j = \ell$, by using Equation (49):

$$\mu''_{v,l} = \frac{\pi_1(v)\cdot b_l^*}{\|b_l^*\|^2} = \frac{(\sum_{i=1}^{g} y_i\mu_{j+i-1,l})b_l^*\cdot b_l^*}{\|b_l^*\|^2} = \frac{(\sum_{i=1}^{g} y_i\mu_{j+i-1,l})\|b_l^*\|^2}{\|b_l^*\|^2} \Rightarrow$$

$$\mu''_{v,l} = \sum_{i=1}^{g} y_i\mu_{j+i-1,l}, \ \ where\ 1 \le l < j = \ell$$

Step 3: The vector of b_ℓ and b_l^* is just shifted for $1 \le l < j < \ell \le j+g-1$, as $\mu''_{\ell,l} = \mu_{\ell-1,l}$, since the index of $\ell = j+g-1$ is eliminated index and the vectors of j to $j+g-2$ would be shifted to $j+1$ to $j+g-1$.

Step 4: Since b_ℓ and b_l^* is not changed for $1 \le l < j < \ell$, so it is set to $\mu''_{\ell,l} = \mu_{\ell,l}$.

Step 5: For $j = l < \ell \le j+g-1$, $\mu''_{\ell,v} = \frac{b_\ell\cdot\pi_j(v)}{\|\pi_j(v)\|^2}$. By using Equation (8) and Equation (24), $\mu''_{\ell,v} = \frac{b_\ell\cdot(w_1b_j^*+\cdots+w_gb_{j+g-1}^*)}{\|\pi_j(v)\|^2} = \frac{(b_\ell^*+\sum_{t=1}^{\ell-1}\mu_{\ell,t}b_t^*)\cdot(w_1b_j^*+\cdots+w_gb_{j+g-1}^*)}{\|\pi_j(v)\|^2}$. The expression of $\sum_{t=1}^{j-1}\mu_{\ell,t}b_t^*$ is not used in the inner product at the numerator of the division:

$$\mu''_{\ell,v} = \frac{(\sum_{t=j}^{\ell}\mu_{\ell,t}b_t^*)\cdot(w_1b_j^*+\cdots+w_{\ell-j+1}b_\ell^*)}{\|\pi_j(v)\|^2} = \frac{\sum_{t=j}^{\ell}\mu_{\ell,t}w_{t-j+1}b_t^*\cdot b_t^*}{\|\pi_j(v)\|^2} \Rightarrow$$

$$\mu''_{\ell,v} = \frac{\sum_{t=j}^{\ell}\mu_{\ell,t}w_{t-j+1}\|b_t^*\|^2}{\|\pi_j(v)\|^2}, \ \ j = l < \ell \le j+g-1$$

Since $\ell = j+g-1$ is the eliminated index, and vectors with indices of j up to $j+g-2$ are shifted to indices of $j+1$ up to $j+g-1$, previous relation would be changed into:

$$\mu''_{\ell,l} = \frac{\sum_{t=j}^{\ell-1}\mu_{\ell-1,t}w_{t-j+1}\|b_t^*\|^2}{\|\pi_j(v)\|^2}, \ \ j = l < \ell \le j+g-1$$

Step 6: By using our reasoning in previous step, for $j = l < j+g \le \ell$, $\mu''_{\ell,v} = \frac{b_\ell.\pi_j(v)}{\|\pi_j(v)\|^2}$. Therefore:

$$\mu''_{\ell,v} = \frac{(b_\ell^*+\sum_{t=1}^{\ell-1}\mu_{\ell,t}b_t^*).(w_1b_j^*+\cdots+w_gb_{j+g-1}^*)}{\|\pi_j(v)\|^2}$$

The expression of $b_\ell^* + \sum_{t=1}^{j-1}\mu_{\ell,t}b_t^* + \sum_{t=j+g}^{\ell-1}\mu_{\ell,t}b_t^*$ would not be used in inner product at numerator of division:

$$\mu''_{\ell,v} = \frac{(\sum_{t=j}^{j+g-1}\mu_{\ell,t}b_t^*)\cdot(w_1b_j^*+\cdots+w_gb_{j+g-1}^*)}{\|\pi_j(v)\|^2} = \frac{\sum_{t=j}^{j+g-1}\mu_{\ell,t}w_{t-j+1}b_t^*\cdot b_t^*}{\|\pi_j(v)\|^2} \Rightarrow$$

$$\mu''_{\ell,l} = \frac{\sum_{t=j}^{j+g-1}\mu_{\ell,t}w_{t-j+1}\|b_t^*\|^2}{\|\pi_j(v)\|^2}, \ \ j = l < j+g \le \ell$$

Step 7: For $j < l < \ell \le j+g-1$:

$$\mu''_{\ell,l} = \frac{b_\ell\cdot b_{l\,new}^*}{\|b_{l\,new}^*\|^2} = \frac{(b_\ell^*+\sum_{t=1}^{\ell-1}\mu_{\ell,t}b_t^*).b_{l\,new}^*}{\|b_{l\,new}^*\|^2}$$

By using Equation (48) with new indices:

$$\mu''_{\ell,l} = \frac{\sum_{t=l-1}^{\ell}\mu_{\ell,t}b_t^*}{\|b_{l\,new}^*\|^2}.((1-\frac{w_{l-j}^2\|b_{l-1}^*\|^2}{\|\pi_{l-1}(v)\|^2})b_{l-1}^* - \frac{w_{l-j}\|b_{l-1}^*\|^2}{\|\pi_{l-1}(v)\|^2}(w_{l-j+1}b_l^*+\cdots+w_gb_{j+g-1}^*)) = \mu_{\ell,l-1} - \frac{w_{l-j}\times\sum_{t=l}^{\ell}\mu_{\ell,t}w_{t-j+1}\|b_t^*\|^2}{\|\pi_{l-1}(v)\|^2-w_{l-j}^2\|b_{l-1}^*\|^2} \Rightarrow$$

$$\mu''_{\ell,l} = \mu_{\ell,l-1} - \frac{w_{l-j}(\sum_{t=l}^{\ell}\mu_{\ell,t}w_{t-j+1}\|b_t^*\|^2)}{\|\pi_l(v)\|^2}$$

Since the index of $\ell = j+g-1$ is eliminated index, and the vectors with indices of j up to $j+g-2$ are shifted to indices of $j+1$ up to $j+g-1$, previous relation would be changed into:

$$\mu''_{\ell,l} = \mu_{\ell-1,l-1} - \frac{w_{l-j}(\sum_{t=l}^{\ell-1}\mu_{\ell-1,t}w_{t-j+1}\|b_t^*\|^2)}{\|\pi_l(v)\|^2},$$

$$where\ j < l < \ell \le j+g-1$$

Step 8: For $j < l < j+g \le \ell$:

$$\mu''_{\ell,l} = \frac{b_\ell\cdot b_{l\,new}^*}{\|b_{l\,new}^*\|^2} = \frac{(b_\ell^*+\sum_{t=1}^{\ell-1}\mu_{\ell,t}b_t^*).b_{l\,new}^*}{\|b_{l\,new}^*\|^2}$$

By using Equation (48) with new indices:

$$\mu''_{\ell,l} = \frac{\sum_{t=l-1}^{j+g-1}\mu_{\ell,t}b_t^*}{\|b_{l\,new}^*\|^2}\cdot((1-\frac{w_{l-j}^2\|b_{l-1}^*\|^2}{\|\pi_{l-1}(v)\|^2})b_{l-1}^* - \frac{w_{l-j}\|b_{l-1}^*\|^2}{\|\pi_{l-1}(v)\|^2}(w_{l-j+1}b_l^*+\cdots+w_gb_{j+g-1}^*)) \Rightarrow$$

$$\mu''_{\ell,l} = \mu_{\ell,l-1} - \frac{w_{l-j}(\sum_{t=l}^{j+g-1}\mu_{\ell,t}w_{t-j+1}\|b_t^*\|^2)}{\|\pi_l(v)\|^2}$$

Step 9: Since the vectors of b_ℓ and b_l^* are not changed for $j+g \le l < \ell$, so it is set to $\mu''_{\ell,l} = \mu_{\ell,l}$.

Step 10: This part is trivial.

Step 11: This part is trivial. □

B.3 Proof of Lemma 3

The proof of Lemma 3 in this appendix is designed in two steps as follows.

Step 1: In this step, it is shown that the update GSO in line 17 of Algorithm 4 from paper [5], for second vector in new lattice block, dose not works under Heuristic 2 and Heuristic 3. Update of second GSO norm in new lattice block under Heuristic 2 and Heuristic 3 in our paper is defined by Equation (33)

as follows:

$$\|b^*_{2new}\| = \|b^*_1\|\sqrt{1-\frac{w_1^2\|b^*_1\|^2}{\|\pi_1(v)\|^2}}$$

Update of second GSO norm of new block is defined in paper [5] as follows:

$$\|b^{*'}_{2new}\| = \|b^*_1\|\sqrt{1-1/d}$$

By using our emulation process of updating GSO, this is shown that updating GSO in [5] for second vector of new block, does not follow Heuristic 2 and Heuristic 3, as follows (for $\omega_t \in \text{Gamma}(\frac{1}{2}, 2)$):

$$X_1 = \sqrt{1-\frac{w_i^2\|b^*_i\|^2}{\|\pi_i(v)\|^2}} \overset{?}{\approx} \sqrt{1-1/d} \overset{i=1}{\Longrightarrow} \frac{\|v\|^2}{w_1^2\|b^*_1\|^2} \overset{?}{\approx} d \Rightarrow \frac{\|v\|^2(\sum_{t=1}^{g-1}\omega_t)}{\omega_1(\|v\|^2-\|b^*_g\|^2)} \overset{?}{\approx} d \Rightarrow \frac{\sum_{t=1}^{g-1}\omega_t}{\omega_1} \overset{?}{\approx} \frac{(\|v\|^2-\|b^*_g\|^2)d}{\|v\|^2} \Rightarrow 1+\frac{\sum_{t=2}^{g-1}\omega_t}{\omega_1} \overset{?}{\approx} \frac{(\|v\|^2-\|b^*_g\|^2)d}{\|v\|^2} \Rightarrow \sum_{t=2}^{g-1}\omega_t \overset{?}{\approx} \left(\frac{(\|v\|^2-\|b^*_g\|^2)d}{\|v\|^2}-1\right)\omega_1 \Rightarrow$$

$$X_2 = \sum_{t=2}^{g-1}\omega_t \text{ and } X_3 = \left(\frac{(\|v\|^2-\|b^*_g\|^2)d}{\|v\|^2}-1\right)\omega_1$$

The expected value and variance of X_2 can be determined as (by using Gamma distribution or CLT theorem) $\mathrm{E}[X_2] = g-2 \approx \text{CUT}-2$ and $\sigma^2[X_2] = 2(g-2) \approx 2\text{CUT}-4$.

Also for X_3, $\mathrm{E}[X_3] = \frac{(\|v\|^2-\|b^*_g\|^2)d}{\|v\|^2}-1$ and $\sigma^2[X_3] = \frac{2d(\|v\|^2-\|b^*_g\|^2)}{\|v\|^2}-2$.

Since the parameters of CUT and $\|b^*_g\|$ are closely depend to each other, therefore we cannot easily compare the expected norm and variance of X_2 and X_3. For example, for different bounding functions, especially with too small success probability over strong reduced lattice blocks, the value of CUT would be far from the value of d (i.e., $\text{CUT} \ll d$; see Figure 3 from [29]), however the value of $\|b^*_g\|$ would more close to the norm of $\|v\|^2$. In fact, to make better comparison in this case, some simulation/experimental tests can be useful.

Step 2: In this step, it is shown that the process of updating GSO in [5] for third vector up to the end vector in new lattice block doesn't emulate truly under Heuristic 2 and Heuristic 3. To violate Heuristic 2 and Heuristic 3, it needs to show that the random variables of $z^2_{d-g+1}, \ldots, z^2_{d-1}, z^2_d$ from coefficient vector z on the normalized orthogonal matrix $[b^*_d/\|b^*_d\|, \ldots, b^*_1/\|b^*_1\|]$ have not strictly decreasing values.

Updating GSO under Heuristic 2 and Heuristic 3 in this paper is defined as:

$$\|b^*_{i\,new}\| = \|b^*_{i-1}\|\sqrt{1-\frac{w_{i-1}^2\|b^*_{i-1}\|^2}{\|\pi_{i-1}(v)\|^2}}$$

Updating GSO in [5] is defined as:

$$\|b^{*'}_{i\,new}\| = \|b^*_i\|\sqrt[\beta-2]{\frac{\|b^*_1\|\|b^*_2\|}{\|v\|\|b^*_1\|\sqrt{1-1/d}}} = \|b^*_i\|\sqrt[\beta-2]{\frac{\|b^*_2\|}{\|v\|\sqrt{1-1/d}}}$$

By using Heuristic 2, Heuristic 3 and the assumption of equality for GSO norms after being update after these two mentioned process of updating GSO (i.e., $\|b^{*'}_{i\,new}\| = \|b^*_{i\,new}\|$), a contradiction would be observed as follows:

$$\|b^{*'}_{i\,new}\| = \|b^*_{i\,new}\| \Rightarrow \|b^*_i\|\sqrt[\beta-2]{\frac{\|b^*_2\|}{\|v\|\sqrt{1-1/d}}} = \|b^*_{i-1}\|\sqrt{1-\frac{w_{i-1}^2\|b^*_{i-1}\|^2}{\|\pi_{i-1}(v)\|^2}} \Rightarrow \frac{\|b^*_{i-1}\|^2-\|b^*_i\|^2\left(\frac{\|b^*_2\|}{\|v\|\sqrt{1-1/d}}\right)^{\frac{2}{\beta-2}}}{\|b^*_{i-1}\|^2} = \frac{w_{i-1}^2\|b^*_{i-1}\|^2}{\|\pi_{i-1}(v)\|^2} \Rightarrow$$

Under GSO assumption ($q = \|b^*_i\|/\|b^*_{i-1}\|$):

$$\frac{\|\pi_{i-1}(v)\|^2}{q^2}\left(q^2-\left(\frac{\|b^*_2\|}{\|v\|\sqrt{1-1/d}}\right)^{\frac{2}{\beta-2}}\right) = w_{i-1}^2\|b^*_{i-1}\|^2 = z^2_{d-i+2} \tag{50}$$

For a typical lattice block $\mathcal{L}_{[1,d]}$, the parameters of $\|b^*_2\|$, $\|v\|$ and q is invariant, so the following expression can be assumed invariant:

$$c_0 = \frac{q^2-(\frac{\|b^*_2\|}{\|v\|\sqrt{1-1/d}})^{\frac{2}{\beta-2}}}{q^2}$$

Lemma 7. *Under Heuristic 2 and Heuristic 3:* $\forall\ l_1, l_2$ *where* $1 \le l_1 < l_2 \le g$: $\|\pi_{l_1}(v)\| > \|\pi_{l_2}(v)\|$.

Proof. By using Equation (21), for $1 \le l_1 < l_2 \le g$:

$$\pi_{l_1}(v) = w_{l_1}b^*_{l_1} + w_{l_1+1}b^*_{l_1+1} + \cdots + w_g b^*_g$$
$$\pi_{l_2}(v) = w_{l_2}b^*_{l_2} + w_{l_2+1}b^*_{l_2+1} + \cdots + w_g b^*_g$$

Moreover, by using Remark 4 in [31]:

$$\text{Prob}(w_i = 0) \approx 0 \text{ for } 1 \le i \le g \Rightarrow$$
$$\text{Prob}(\|\pi_{l_1}(v)\| > \|\pi_{l_2}(v)\|) = 1 \qquad \square$$

In this appendix, by using Equation (50), the entries of z_{d-i+2} strictly has increasing values, since the values of $\|\pi_{i-1}(v)\|$ strictly has decreasing slope by Lemma 7. Therefore by using Lemma 9 in [31] which states that $\mathrm{E}[z^2_x] \approx \frac{\|v\|^2-\|b^*_g\|^2}{\text{CUT}}$, this is found that the coefficient vector z by Algorithm 4 from paper [5] is not expected to be uniformly distributed on the normalized orthogonal matrix $[b^*_d/\|b^*_d\|, \ldots, b^*_1/\|b^*_1\|]$ and consequently, Heuristic 2 and Heuristic 3 are violated.

Therefore, the assumed equality in Step 2 reaches to a contradiction, and the proof is completed. $\square$

B.4 Proof of Lemma 4

By using relation Equation (30), for each block $[b_j, b_{j+1}]$, the initial radius is $R = \|b^*_j\|$. By using

Remark 1, the GSO basis is automatically assumed to be size-reduced. After a full-finished running of BKZ_2 with full-enumeration, there is no 2-dimensional lattice block in this basis, in the way that full-enumeration succeeds over it. In other side, after running of $LLL_{\delta\approx 1}$, there is no two subsequent GSO vectors, in the way that Lovasz-condition over it would be violated. Therefore, if we show that 2-dimensional full-enumeration is equal to Lovasz-condition for a GSO basis, then the proof would be completed. In this proof, each two subsequent GSO vectors of GSO basis are represented by b_1^* and b_2^*.

(a) If full enumeration over $[b_1^*, b_2^*]$ dose not succeed (by using Equation (21) and Theorem 1):

$$v = [w_1 = 1, w_2 = 0] \cdot \begin{bmatrix} b_1^* \\ b_2^* \end{bmatrix} = b_1^*$$

The block after this full enumeration would be $[b_1^*, b_2^*]$ again.
By using Remark 2, since full-enumeration over this block dose not succeed, so the norm of $\|\pi_i(b_{i+1})\|$ isn't more or equal than $\|\pi_i(b_i)\|$, and consequently after Lovasz checking, the block remain in the form of $[b_1^*, b_2^*]$.

(b) If full-enumeration over $[b_1^*, b_2^*]$ succeeds (by using Equation (21) and Equation (24) and Theorem 1):

$$v = [w_1, w_2] \cdot \begin{bmatrix} b_1^* \\ b_2^* \end{bmatrix} = [y_1 + y_2\mu_{2,1}, y_2 = 1] \cdot \begin{bmatrix} b_1^* \\ b_2^* \end{bmatrix}$$

Since the norm of solution vector v from full-enumeration should be shortest norm (as the solution of exact-SVP) in the projected 2-dimensional lattice block, together with the fact of $\mu_{2,1} < \frac{1}{2}$, therefore y_1 should be zero and consequently vector v can be written as follows:

$$v = [w_1, w_2] \cdot \begin{bmatrix} b_1^* \\ b_2^* \end{bmatrix} = [\mu_{2,1}, y_2 = 1] \cdot \begin{bmatrix} b_1^* \\ b_2^* \end{bmatrix} = b_2^* + \mu_{2,1} b_1^* = \pi_1(b_2) \quad (51)$$

Therefore, the block after this full-enumeration would be $[\pi_1(b_2), \pi_2(b_1)]$ with GSO norms as follows (by using relation Equation (33)):

$$[\|\pi_1(b_2)\|, \|\pi_2(b_1)\|] = \left[\sqrt{\|b_2^*\|^2 + \mu_{2,1}^2\|b_1^*\|^2}, \frac{\|b_1^*\|\|b_2^*\|}{\sqrt{\|b_2^*\|^2 + \mu_{2,1}^2\|b_1^*\|^2}}\right]$$

By using Remark 2, since full-enumeration over this block succeeds, so by using Equation (51), the norm of $\|\pi_i(b_{i+1})\|$ would be more or equal to $\|\pi_i(b_i)\|$, and consequently after Lovasz checking, this block would be $[\pi_1(b_2), \pi_2(b_1)]$.

The proof is completed. □

B.5 Proof of Lemma 6

After swapping $b_i \leftrightarrow b_{i+1}$, the Euclidean norms of these vectors don't changed, while the GSO norms of b_i^*, b_{i+1}^* just be modified by formula Equation (37), also the GSO coefficients of $\mu_{i,1}$, $\mu_{i,2}$, …, $\mu_{i,i-1}$, $\mu_{i+1,1}$, $\mu_{i+1,2}$, …, $\mu_{i+1,i}$, and $\mu_{l,i}$, $\mu_{l,i+1}$ for $i+2 \leq l \leq n$ should be modified too. Note that, by using Equation (8), only if b_{i1} and b_{i2}^* are not modified then the corresponding GSO coefficient of $\mu_{i1,i2}$ would not be changed too, and therefore the remains of GSO coefficients in matrix μ would be unchanged. At first, the new value for coefficient of $\mu_{i+1,i}$ is restored as $\mu''_{i+1,i}$ which is substituted with $\mu_{i+1,i}$ at the end of the process. As mentioned, the sequence of operations is important in this lemma.

$$\pi_i(b_{i+1}) = b_{i+1}^* + \mu_{i+1,i} \times b_i^* \quad (52)$$

By using Equation (8) and Equation (52):

$$\mu''_{i+1,i} = \frac{b_i \cdot \pi_i(b_{i+1})}{\|\pi_i(b_{i+1})\|^2} = \frac{(b_i^* + \sum_{t=1}^{i-1} \mu_{i,t} b_t^*) \cdot (b_{i+1}^* + \mu_{i+1,i} b_i^*)}{\|\pi_i(b_{i+1})\|^2} = \frac{b_i^* \cdot (\mu_{i+1,i} b_i^*)}{\|\pi_i(b_{i+1})\|^2}$$

By using Equation (37):

$$\mu''_{i+1,i} = \frac{\mu_{i+1,i}\|b_i^*\|^2}{\|b_{i+1}^*\|^2 + \mu_{i+1,i}^2\|b_i^*\|^2}$$

For $l = i+2$ to n, new value for $\mu_{l,i}$ is updated as follows. By using Equation (8):

$$\mu''_{l,i} = \frac{b_l \cdot \pi_i(b_{i+1})}{\|\pi_i(b_{i+1})\|^2} = \frac{(b_l^* + \sum_{t=1}^{l-1} \mu_{l,t} b_t^*) \cdot (b_{i+1}^* + \mu_{i+1,i} b_i^*)}{\|\pi_i(b_{i+1})\|^2} = \frac{(\mu_{l,i} b_i^* + \mu_{l,i+1} b_{i+1}^*) \cdot (b_{i+1}^* + \mu_{i+1,i} b_i^*)}{\|\pi_i(b_{i+1})\|^2} = \frac{\mu_{l,i+1} b_{i+1}^* \cdot b_{i+1}^* + \mu_{i+1,i}\mu_{l,i} b_i^* \cdot b_i^*}{\|\pi_i(b_{i+1})\|^2}$$

By using Equation (37):

$$\mu''_{l,i} = \frac{\mu_{l,i+1}\|b_{i+1}^*\|^2 + \mu_{i+1,i}\mu_{l,i}\|b_i^*\|^2}{\|b_{i+1}^*\|^2 + \mu_{i+1,i}^2\|b_i^*\|^2}$$

Also for $l = i+2$ to n, by using Equation (52), new value for $\mu_{l,i+1}$ is updated as follows:

$$\pi_{i+1}(b_i) = b_i^* - \mu''_{i+1,i}\pi_i(b_{i+1}) \Rightarrow \pi_{i+1}(b_i) = b_i^* - \mu''_{i+1,i}(b_{i+1}^* + \mu_{i+1,i} b_i^*)$$

$$\pi_{i+1}(b_i) = (1 - \mu''_{i+1,i}\mu_{i+1,i}) \cdot b_i^* - \mu''_{i+1,i} \times b_{i+1}^* \quad (53)$$

By using Equation (8) and Equation (53):

$$\mu''_{l,i+1} = \frac{b_l \cdot \pi_{i+1}(b_i)}{\|\pi_{i+1}(b_i)\|^2} = \frac{(b_l^* + \sum_{t=1}^{l-1} \mu_{l,t} b_t^*) \cdot \left((1 - \mu''_{i+1,i}\mu_{i+1,i}) b_i^* - \mu''_{i+1,i} b_{i+1}^*\right)}{\|\pi_{i+1}(b_i)\|^2} \Rightarrow \mu''_{l,i+1} = \frac{\mu_{l,i}(1 - \mu''_{i+1,i}\mu_{i+1,i})\|b_i^*\|^2 - \mu_{l,i+1}\mu''_{i+1,i}\|b_{i+1}^*\|^2}{\|\pi_{i+1}(b_i)\|^2}$$

By using Equation (37):

$$\mu''_{l,i+1} = \mu_{l,i} - \mu_{l,i+1} \times \mu_{i+1,i}$$

Then following substitutions the matrix μ is applied:

(1) $\mu_{l,i+1} \leftarrow \mu''_{l,i+1}$, $i+2 \leq l \leq n$
(2) $\mu_{l,i} \leftarrow \mu''_{l,i}$, $i+2 \leq l \leq n$
(3) For $1 \leq t \leq i-1$: swap_entry$(\mu_{i,t}, \mu_{i+1,t})$ //$\mu_{i,i}$ and $\mu_{i+1,i+1}$ remain 1
(4) $\mu_{i+1,i} \leftarrow \mu''_{i+1,i}$ □