Improved Progressive BKZ with Lattice Sieving

Wenwen Xia[1,⋆], Leizhang Wang[1,⋆], Geng Wang[2], Dawu Gu[1,2], and Baocang Wang[1]

[1] Xidian University
{xiawenwen, lzwang_2, bcwang}@stu.xidian.edu.cn
[2] Shanghai Jiao Tong University
{wanggxx, dwgu}@sjtu.edu.cn

Abstract. BKZ is currently the most efficient algorithm in solving the *approximate shortest vector problem* (SVP_γ). One of the most important parameter choices in BKZ is the blocksize β, which greatly affects its efficiency and reduction effort. In 2016, Aono *et al.* presented *Improved Progressive BKZ* (pro-BKZ). By designing a blocksize strategy selection algorithm, their pro-BKZ runs faster than BKZ 2.0 which has a fixed blocksize. However, pro-BKZ only considers enumeration as its subroutine, without using the more efficient lattice sieving algorithm. Besides, their blocksize strategy selection is not optimal, so the strategy selection algorithm could further be improved.

In this paper, we present a new lattice solving algorithm called *Improved Progressive pnj-BKZ* (pro-pnj-BKZ) mainly based on an optimal blocksize and jump strategy selection algorithm for BKZ with sieving, which relies on accurate time cost models and simulating algorithms. We propose the following approaches:

- New simulator for sieving and BKZ with sieving. A simulator is used for simulating lattice reduction process without running the BKZ algorithm itself. We give a new simulator for BKZ, to simulate the cases where blocks in BKZ with sieve oracle jump by more than one dimension.
- New two-step mode for solving SVP_γ with BKZ and sieving. Other than a subroutine of BKZ, sieving can also be combined with BKZ to get a more efficient lattice solving algorithm, but the best way of combination is currently unknown. We show that to solve SVP_γ more efficiently, one should first repeatedly run BKZ to improve the quality of lattice basis and finally run progressive sieving once, since BKZ performs better in lattice basis reduction, while sieving performs better in finding short vectors. By our simulator, we can properly choose the suitable timing where the algorithm ends the BKZ routine and begins sieving.
- New blocksize and jump strategy selection algorithm for BKZ with sieving. Since the blocksize strategy selection algorithm in pro-BKZ is not optimal and doesn't consider the jump strategy, we design a new blocksize and jump strategy selection algorithm to give an optimal SVP_γ solving strategy. In particular, we improve the efficiency by 8.4∼10.5 times compared with the heuristic blocksize strategy in G6K. We also re-estimate the concrete hardness of the lattice-based NIST candidate schemes from the LWE primal attack and it decreases by 2.4∼10 bits.

- We test the efficiency of the pro-pnj-BKZ with the TU Darmstadt LWE challenge and break the LWE challenges with $(n, \alpha) \in \{(40, 0.035), (40, 0.040), (50, 0.025), (55, 0.020), (90, 0.005)\}$.

Keywords: cryptanalysis, lattice reduction, SVP_γ, progressive BKZ, optimal blocksize selection

1 Introduction

To date, many post-quantum cryptosystems are lattice-based, e.g. Dilithium [1], Kyber [2] which have been accepted as NIST standards. Lattice-based structures appear to be immune from both classical and quantum attacks. As a result, many lattice-based constructions are considered secure, assuming that certain well-studied computational lattice problems cannot be solved in polynomial time. A large fraction of lattice-based cryptographic mechanisms are built upon the LWE problem [3] and its variants [2–5]. One of the best-known cryptanalytic techniques against these problems is primal attack [6], which is widely used in cryptanalysis of lattice-based cryptosystems. The primal attack solves the LWE problem by reducing it to the unique Shortest Vector problem (uSVP_γ). It then calls an approximate Shortest Vector Problem (SVP_γ) solver to find the approximate shortest vector which is used to recover the solution of LWE.

SVP_γ is a basic lattice hard problem. In recent years, substantial improvements have been made in solving SVP_γ. In 1982, the first polynomial-time lattice reduction algorithm named the LLL [7] was proposed to solve SVP_γ with an exponential approximation factor γ. To solve the problem with a smaller approximation factor, Schnorr and Euchner [8] presented Block Korkin-Zolotarev (BKZ) reduction, which is considered as a combination of LLL algorithm and the enumeration algorithm to balance the algorithm's time consumption and the success probability using a parameter β called the blocksize. In the literature, many cryptanalysts improved the BKZ algorithm, e.g. the extreme pruning technique [9] to speed up enumeration, BKZ 2.0 [10] based on [9], approximate enumeration oracle [11] on speeding up enumeration, and parameters optimization in BKZ such as Improved Progressive BKZ (pro-BKZ) [12].

A BKZ simulator is used to predict the practical behavior of a BKZ algorithm (when $\beta \geq 45$), which is important in optimizing the parameter selection in BKZ. Based on the Gaussian heuristic, Chen and Nguyen refined the sandpile model from [13] and provided a BKZ simulator in BKZ 2.0 [10]. Using the properties that the last β vectors in BKZ-β reduction basis satisfy HKZ reduction and Gaussian heuristic, [12] proposed a simulator for predicting BKZ-β fully reduced basis. Since the BKZ 2.0 simulator could not accurately predict the head concavity phenomenon after multiple tours of BKZ-β, Bai *et al.* [14] considered the norm of the shortest vector as a random variable rather than a fixed value, and brought randomness into the BKZ 2.0 simulator. The new simulator [14] can effectively predict and explain the phenomenon of head concavity of lattice basis reduced by multiple tours of BKZ.

Based on the BKZ simulator, pro-BKZ [12] solves SVP_γ first by calling a series of BKZs with different blocksizes first to optimize the basis quality, and finally an SVP oracle to find the approximate shortest vector (we call it a two-step mode). One of the main contributions of their work is a blocksize strategy selection algorithm to generate the different blocksizes to be used in the BKZ reduction, which uses the shortest path algorithm to solve an optimized blocksize strategy by setting multiple different middle reduction qualities as the inner nodes. But in this paper, we shall show that their method is not supposed to generate an optimal blocksize strategy, and still has room for improvement.

Besides the development of BKZ reduction itself, some researchers attempted to replace the enumeration algorithm in BKZ with a sieving algorithm, as with the development of the memory manufacturing process, large-memory machines are available. A lattice sieving algorithm requires more memory but less time than enumeration.

In 2019, Albrecht *et al.* [15] designed the General Sieve Kernel (G6K), implemented a new version of BKZ named *pump-and-jump BKZ* (pnj-BKZ) and the progressive sieving algorithm named Pump can selectively call the Gauss sieve [16, 17], NV sieve [18], k-list sieve [19, 20] or BGJ1 sieve [21]. Pump is a generic design based on sieving algorithms using the progressive sieve introduced in [22] (Similar ideas are also independently proposed in [23]) with dimension-for-free technique [23], which makes the sieving process more efficient and allows a higher solving rate. Ducas *et al.* [24] improved the efficiency of G6K using GPU and implemented the fastest sieving algorithm BDGL16 [25] in both G6K and G6K-GPU-Tensor. Unlike classical BKZ using an enumeration algorithm as its SVP oracle, pnj-BKZ adopts Pump as its SVP oracle with a selective parameter *jump*. The *jump* value controls the jump stage of blocks in BKZ with a sieve oracle, which can jump by more than one dimension. Default mode in G6K using pnj-BKZ and Pump solves TU Darmstadt challenges 400 times faster than the previous records for comparable instances. However, the blocksize strategy used in the default parameter selection in G6K is heuristic. As shown in pro-BKZ [12], the default mode of G6K can be further improved by an optimized blocksize strategy selection algorithm. But to implement an optimized blocksize strategy selection algorithm adapting from pro-BKZ, simulating algorithms and cost models for both pnj-BKZ and Pump are necessary.

Contribution. In this work, we propose *Improved Progressive pnj-BKZ*(pro-pnj-BKZ) mainly based on an optimal blocksize and jump strategy selection algorithm for pnj-BKZ, which relies on accurate time cost models and simulating algorithms for pnj-BKZ and Pump. More specifically:

- We construct a new simulator to simulate the lattice reduction process for pnj-BKZ, especially in the case of $jump > 1$, and give a new Pump estimation algorithm for determining the dimension in Pump when solving SVP_γ and LWE problem. Furthermore, we design a new simulator to simulate the quality of lattice basis after Pump.

- We propose a new model for solving SVP_γ by the simulating algorithms. We modify the default mode in G6K to a two-step mode, which firstly calls a series

of pnj-BKZs following a blocksize selection strategy to reduce the basis and then uses a Pump algorithm to search the approximate shortest vector. Compared to the G6K's default strategy, we can efficiently save the time cost during the stage for improving the quality of lattice basis through the new blocksize and jump strategies. Besides, we avoid wasting time due to failed Pumps in G6K's default strategy by a high solving probability Pump which executes only once. Eventually, we solve SVP_γ more efficiently than G6K's default strategy through the improvement during these two stages.

- Based on the simulating algorithms and two-step mode, we give two new blocksize and jump strategy selection algorithms. We borrow the same shortest path algorithm as in pro-BKZ to design our first algorithm called *blocksize strategy selection algorithm based on pro-BKZ* (BSSA) by replacing the BKZ and enumeration algorithm with pnj-BKZ and Pump respectively. However, we find that the strategy generated by BSSA is not optimal. To obtain an optimal blocksize strategy, we design a new strategy selection algorithm named *blocksize strategy enumeration* (EnumBS). EnumBS can obtain an optimal blocksize strategy at the cost of a higher theoretical complexity than BSSA, but the time is still acceptable for low-dimensional lattices. Using the blocksize strategy chosen from EnumBS, the algorithm increases the efficiency at most 10.5 times in solving the TU Darmstadt LWE challenges compared with the default LWE solver in G6K.

- Based on these two blocksize strategy selection algorithms, we propose and have implemented the *Improved Progressive pnj-BKZ* (pro-pnj-BKZ) and have made public[1] and solved the TU Darmstadt LWE challenges[2] $(n, \alpha) \in \{(40,0.035), (90,0.005), (50,0.025), (55,0.020), (40,0.040)\}$.

Organization. The paper is organized as follows. Section 2 presents basic notations and preliminaries. Section 3 presents the sketch of pro-pnj-BKZ. Section 4 design a pnj-BKZ simulator and a Pump sieving dimension estimator for simulation. Section 5 proposes our two improved blocksize strategy selection algorithms in detail and compare them both in time cost and optimization effect. Section 6 presents the results for solving the TU Darmstadt LWE challenge by pro-pnj-BKZ, and compare it with other LWE solving algorithms. Section 7 gives a conclusion and the prospect for further study.

2 Preliminaries

2.1 Notations and Basic Definitions

We write a matrix $\mathbf{B}$ as $\mathbf{B} = (\mathbf{b}_0, \cdots, \mathbf{b}_{d-1})$ where $\mathbf{b}_i$ is the $(i+1)$-th column vector of $\mathbf{B}$. The Euclidean norm of a vector $\mathbf{v}$ is denoted by $\|\mathbf{v}\|$.

[1] https://github.com/Summwer/lwe-estimator-with-pnjbkz.git
[2] https://www.latticechallenge.org/lwe_challenge/challenge.php

If $\mathbf{B} \in \mathbb{R}^{d\times d}$ has full rank d, the lattice $\mathcal{L}$ generated by the basis $\mathbf{B}$ is denoted by $\mathcal{L}(\mathbf{B}) = \{\mathbf{Bx}|\mathbf{x} \in \mathbb{Z}^d\}$. We denote $\mathbf{B}^* = (\mathbf{b}_0^*, \cdots, \mathbf{b}_{d-1}^*)$ as the Gram-Schmidt orthogonalization of $\mathbf{B}$, in which $\mathbf{b}_i^* = \mathbf{b}_i - \sum_{j=0}^{i-1} \mu_{i,j}\mathbf{b}_j^*$, $\mu_{i,j} = \frac{\langle \mathbf{b}_i, \mathbf{b}_j^* \rangle}{\|\mathbf{b}_j^*\|^2}$.

For $i \in \{0, \cdots, d-1\}$, we denote the orthogonal projection to the span of $(\mathbf{b}_0, \cdots, \mathbf{b}_{i-1})$ by π_i, i.e. $\forall \mathbf{v}$, $\pi_i(\mathbf{v}) = \mathbf{v} - \sum_{j=0}^{i-1} \omega_j \mathbf{b}_j^*$, in which $\omega_j = \frac{\langle \mathbf{v}, \mathbf{b}_j^* \rangle}{\|\mathbf{b}_j^*\|^2}$.

For $i, j \in \mathbb{Z}_d$ and $0 \leq i < j \leq d-1$, given an arbitrary d-dimensional vector $\mathbf{v} = (v_0, \cdots, v_{d-1})$, define $\mathbf{v}_{[i:j]}$ as $(v_i, \cdots, v_{j-1})$ with a size $j - i$. For a lattice basis $\mathbf{B}$, let $\mathbf{B}_{[i:j]} \leftarrow (\mathbf{b}_i, \cdots, \mathbf{b}_{j-1})$. Moreover, we denote $\mathbf{B}_{\pi[i:j]}$ by the local projected block $(\pi_i(\mathbf{b}_i), \cdots, \pi_i(\mathbf{b}_{j-1}))$, and call $\mathcal{L}_{\pi[i:j]}$ the lattice generated by $\mathbf{B}_{\pi[i:j]}$. We use $\mathbf{B}_{\pi[i]}$ and $\mathcal{L}_{\pi[i]}$ as shorthands for $\mathbf{B}_{\pi[i:d]}$ and $\mathcal{L}_{\pi[i:d]}$.

The volume of a lattice $\mathcal{L}(\mathbf{B})$ is $\mathrm{Vol}(\mathcal{L}(\mathbf{B})) = \prod_{i=0}^{d-1} \|\mathbf{b}_i^*\|$, an invariant of the lattice. The first minimum of a lattice $\mathcal{L}(\mathbf{B})$ is the length of the shortest non-zero vector, denoted by $\lambda_1(\mathcal{L}(\mathbf{B}))$. We use the abbreviations $\mathrm{Vol}(\mathbf{B}) = \mathrm{Vol}(\mathcal{L}(\mathbf{B}))$ and $\lambda_1(\mathbf{B}) = \lambda_1(\mathcal{L}(\mathbf{B}))$.

Suppose the input basis is $\mathbf{B} = (\mathbf{b}_0, \cdots, \mathbf{b}_{d-1})$ and its corresponding Gram-Schmidt basis is $\mathbf{B}^* = (\mathbf{b}_0^*, \cdots, \mathbf{b}_{d-1}^*)$, the logarithms of the Gram-Schmidt norms $l_i = \ln(\|\mathbf{b}_i^*\|)$, $i \in \{0, \cdots, d-1\}$. Let $\mathsf{rr}(\mathbf{B}) = (l_0, \cdots, l_{d-1})$, abbreviate to rr, $\mathsf{rr}_{[i:j]} = (l_i, \cdots, l_{j-1})$.

Notations for algorithms description. Let BKZ-β/pnj-BKZ-(β, J) be an abbreviation of a one-tour BKZ/pnj-BKZ with blocksize β and jump value J. Assume the input basis is $\mathbf{B}$, and the basis $\mathbf{B}$ reaches a basis quality after calling sufficient tours of BKZ-β. To simplify the above step, we use β to imply the quality of a BKZ-β reduced basis. Let $\sharp$tours be the minimum tours for BKZ-β/pnj-BKZ-(β, J) to reach a BKZ-β/pnj-BKZ-(β, J) reduced basis. Denote t as the number of tours for implementing BKZ/pnj-BKZ with a fixed blocksize β.

Let $T_{\mathrm{BKZ}}(\beta)/T_{\mathrm{pnjBKZ}}(\beta, J)$ be time assumption of one-tour BKZ/pnj-BKZ with blocksize β and jump value J. Let $T_{\mathrm{BKZs}}(S)/T_{\mathrm{pnjBKZs}}(S)$ be total time cost for series of BKZ/pnj-BKZ with a specific blocksize strategy S (e.g. $S = [\beta_0, \cdots, \beta_{n-1}]$), abbreviate it as $T_{\mathrm{BKZs}}/T_{\mathrm{pnjBKZs}}$.

Denote $T_{\mathrm{pump}}(d_{\mathrm{svp}})$ as the time cost of Pump with sieve dimension equal to d_{svp}, abbreviate it as T_{pump}. Let PSC be the expected Pump cost to find the target vector, which will be explained in the section 4.2.

Definition 1. *(The Gaussian Distribution [26]) Let $\sigma \in \mathbb{R}$ be the standard deviation, $u \in \mathbb{R}$ be the mean value, a continuous Gaussian Distribution can be denoted as $N(u, \sigma^2)$. The probabilistic density function of $N(u, \sigma^2)$ is*

$$\rho_{N(u,\sigma^2)} = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-u)^2}{2\sigma^2}}.$$

Lemma 1. *(Chi-Squared Distribution [26]) Given n random variables $X_i \sim N(0, 1)$, the random variables $X_0^2 + \cdots + X_{m-1}^2$ follows a chi-squared distribution χ_m^2 over $\mathbb{R}^*$ of mean m and variance $2m$ with probabilistic density function*

$$\rho_{\chi_m^2}(x) = \frac{1}{2^{\frac{m}{2}}\Gamma(\frac{m}{2})} x^{\frac{m}{2}-1} e^{-\frac{x}{2}}.$$

Given m random variables $Y_i \sim N(0, \sigma^2)$, the random variables $Y_0^2 + \cdots + Y_{m-1}^2$ follows a scaled chi-squared distribution $\sigma^2 \cdot \chi_m^2$ over $\mathbb{R}^*$ of mean $m\sigma^2$ and variance $2m\sigma^2$.

Proposition 1. *(Gaussian Heuristic [23]) The expected first minimum of a lattice $\mathcal{L}$ (denoted as $\lambda_1(\mathcal{L}(\mathbf{B}))$) according to the Gaussian Heuristic denoted by $\mathrm{GH}(\mathcal{L})$ is given by*

$$\lambda_1(\mathcal{L}(\mathbf{B})) \approx \mathrm{GH}(\mathcal{L}) = \left(\frac{\mathrm{Vol}(\mathcal{L})}{\mathrm{V}_d(1)}\right)^{\frac{1}{d}} = \frac{\left(\Gamma(\frac{d}{2}+1)\cdot \mathrm{Vol}(\mathcal{L})\right)^{\frac{1}{d}}}{\sqrt{\pi}} \approx \sqrt{\frac{d}{2\pi e}} \cdot \mathrm{Vol}(\mathcal{L})^{\frac{1}{d}}.$$

Where $\mathrm{V}_d(1)$ is the volume of the d-dimensional unit sphere. We also write $\mathrm{GH}(\mathbf{B}) = \mathrm{GH}(\mathcal{L}(\mathbf{B}))$ and $\mathrm{GH}(rr_{[i:j]}) = \mathrm{GH}(\mathbf{B}_{\pi[i:j]})$.

Definition 2. *(HKZ reduction and BKZ reduction [23]) The basis $\mathbf{B}$ of a lattice $\mathcal{L}$ is said to be HKZ reduced if $\mathbf{b}_i^* = \lambda_1(\mathcal{L}(\mathbf{B}_{\pi[i:d]}))$, for all $i < d$. It is said BKZ reduced with block-size β (also called as BKZ-β reduced) if $\mathbf{b}_i^* = \lambda_1(\mathcal{L}(\mathbf{B}_{\pi[i:\min\{i+\beta,d\}]}))$, for all $i < d$.*

Definition 3. *(Root Hermite Factor [27]) For a basis $\mathbf{B}$ of d-dimensional lattice, the root Hermite factor is defined as*

$$\delta = \left(\|\mathbf{b}_0\| / \mathrm{Vol}(\mathbf{B})^{1/d}\right)^{1/d}, \tag{1}$$

for estimating the equality of the output vector of BKZ. For larger blocksize, it follows the asymptotic formula

$$\delta(\beta)^{2(\beta-1)} = \frac{\beta}{2\pi e}(\beta\pi)^{1/\beta}. \tag{2}$$

Definition 4. *(Geometric Series Assumption [15]) Let $\mathbf{B}$ be a lattice basis after lattice reduction, then Geometric Series Assumption states that $\|\mathbf{b}_i^*\| \approx \alpha \cdot \|\mathbf{b}_{i-1}^*\|$, $0 < \alpha < 1$.*

Combine the GSA with root-Hermite factor (Equation (1)) and $Vol(\mathcal{L}(\mathbf{B})) = \prod_{i=0}^{d-1} \|\mathbf{b}_i^*\|$, it infers that $\alpha = \delta^{-\frac{2d}{d-1}} \approx \delta^{-2}$.

2.2 Lattice Hard Problems

SVP Problem and its Variants

Definition 5. *(Shortest Vector Problem(SVP) [28]) Given an arbitrary basis $\mathbf{B}$ of some lattice $\mathcal{L} = \mathcal{L}(\mathbf{B})$, find a shortest non-zero vector, $\mathbf{v} \in \mathcal{L}$ for such that $\|\mathbf{v}\| = \lambda_1(\mathcal{L})$.*

Definition 6. *(Approximate Shortest Vector Problem(SVP_γ) [28]) Given an arbitrary basis $\mathbf{B}$ of some lattice $\mathcal{L} = \mathcal{L}(\mathbf{B})$, find a shortest non-zero vector, $\mathbf{v} \in \mathcal{L}$ such that $\|\mathbf{v}\| \leq \gamma(d) \cdot \lambda_1(\mathcal{L})$, where $\gamma(d) \geq 1$ is an approximation factor function of lattice dimension d.*

Definition 7. *(unique Shortest Vector Problem(uSVP$_\gamma$) [29]) Given an arbitrary basis* $\mathbf{B}$ *of some lattice* $\mathcal{L} = \mathcal{L}(\mathbf{B})$*,* $\mathcal{L}$ *satisfying the condition* $\gamma\lambda_1(\mathbf{B}) < \lambda_2(\mathbf{B})$ *($\lambda_2(\mathbf{B})$ is norm of the second shortest vector which is linearly independent to the shortest vector), find the shortest non-zero vector* $\mathbf{v}$ *such that* $\|\mathbf{v}\| = \lambda_1(\mathbf{B})$*.*

Definition 8. *(LWE$_{m,n,q,D_\sigma}$ Distribution [28, 30, 31]) Given a number of samples* $m \in \mathbb{Z}$*, a secret vector length* $n \in \mathbb{Z}$*, a modulo* $q \in \mathbb{Z}$ *, a probability distribution* D_σ*. Uniformly sample a matrix* $\mathbf{A} \in \mathbb{Z}_q^{m\times n}$ *and sample a secret vector* $\mathbf{s} \in \mathbb{Z}_q^n$ *from a specific distribution, randomly sample a relatively small noise vector* $\mathbf{e} \in \mathbb{Z}_q^m$ *from Gaussian distribution* D_σ *whose standard deviation is* σ*. The LWE distribution* Ψ *is constructed by the pair* $(\mathbf{A}, \mathbf{b} = \mathbf{As} + \mathbf{e}) \in (\mathbb{Z}_q^{m\times n}, \mathbb{Z}_q^m)$ *sampled as above.*

Definition 9. *(Search LWE$_{m,n,q,D_\sigma}$ problem [28, 30, 31]) Given a pair* $(\mathbf{A}, \mathbf{b})$ *sampled from LWE distribution* Ψ *compute the pair* $(\mathbf{s}, \mathbf{e})$*.*

2.3 G6K and G6K-GPU-Tensor

G6K [15] is an abstract machine for running sieve algorithms and deriving lattice reduction. Its performance is far better than any earlier lattice solving algorithms. Solving SVP with G6K is at least 400 times faster than the previous records in the TU Darmstadt SVP Challenge. G6K's main contribution is building on, generalizing, and extending the previous sieve algorithms. G6K-GPU-Tensor improves the efficiency of G6K through GPU implementations. The above improvement reaches dimension 180 for TU Darmstadt SVP Challenge in 51.6 days on a server with 4 NVIDIA Turing GPUs and 1.5 TB of RAM.

2.4 Sieving Algorithms and Pump in G6K

Sieving Algorithms The first and simplest of practical sieving algorithms by Nguyen and Vidick uses a database of $N_0 = (4/3)^{d/2+o(d)} \approx 2^{0.2075d+o(d)}$ vectors and runs in time $N_0^{2+o(1)} \approx 2^{0.415d+o(d)}$ by repeatedly checking all pairs $\mathbf{v} \pm \mathbf{w}$ [18]. The database size of $(4/3)^{d/2+o(d)}$ is the minimal number of vectors that should be reached to ensure finding enough short pairs constantly, and eventually saturate the ball of radius $\sqrt{4/3} \cdot \mathrm{GH}(L)$. In a line of works [21, 25, 32, 33] the time complexity was gradually improved to $2^{0.292d+o(d)}$ by nearest neighbour searching techniques to find close pairs more efficiently.

Progressive Sieve Progressive sieve [23] is a sieve technique to save the cost of full dimensional sieve. It can be realized by a right-to-left operation. It first calls a sieving algorithm on a lattice projection with small dimension, then uses Babai's nearest plane algorithm [34] to recover the vector to a lattice projection with higher dimension. Repeat the above step until recover the short vectors onto a full dimensional lattice.

Dimension for Free Technique Dimension for free technology [23] is an improvement technology for sieve algorithms which can bring sub-exponential time speedup and memory decreasing. [23] has given two theoretical dimension for free (d4f) estimations for solving β-dimension SVP as $\mathrm{d4f}(\beta) = \frac{\beta \ln(4/3)}{\ln(\beta/2\pi)}$ and $\mathrm{d4f}(\beta) = \frac{\beta \ln(4/3)}{\ln(\beta/2\pi e)}$, while in the code implementation of G6K [15], it gives a more relaxed bound as

$$\mathrm{d4f}(\beta) = \begin{cases} 0, & \beta < 40 \\ \lfloor \frac{\beta - 40}{2} \rfloor, & 40 \leq \beta \leq 75 \\ \lfloor 11.5 + 0.075\beta \rfloor, & \beta > 75. \end{cases} \tag{3}$$

Pump in G6K Martin R. Albrecht *et al.* proposed Pump algorithm in [15], which is improved based on progressive Sieve [22] with dimension for free technique [23] and the insertion tricks in [23]. There are four input parameters for Pump algorithm: lattice basis $\mathbf{B}$, left insertion bound κ, insertion upper bound d_{svp} ($\kappa + d_{\mathrm{svp}} = d$) and dimension-for-free value $\mathrm{d4f}(d_{\mathrm{svp}})$ (the upper bound of sieve dimension is $d_{\mathrm{svp}} - \mathrm{d4f}(d_{\mathrm{svp}})$) as mentioned in section 2.4. After calling a Pump, it will return a basis reduced by Pump.

2.5 Pnj-BKZ in G6K

The pnj-BKZ algorithm is a BKZ-type lattice reduction algorithm that uses Pump as the subroutine for finding the shortest projected vector on the projected sub-lattice. Unlike the classical BKZ algorithm, pnj-BKZ could perform the SVP oracle with an adjustable *jump* no less than 1. More specifically, after executing the SVP oracle on a certain block $\mathbf{B}_{\pi[i:i+\beta']}$, the SVP oracle will be executed on the $\mathbf{B}_{[i+J:i+\beta'+J]}$ block with a *jump* count J rather than $\mathbf{B}_{[i+1:i+\beta'+1]}$. The detailed description of pnj-BKZ is Algorithm 1.

2.6 Slope: the measurement of the lattice basis quality

To measure the quality of lattice basis we use the same averaged quality measurement as that used in [35]. It is the least squares fit coefficient of the slope of $\log \|\mathbf{b}_i^*\|$, which means that the larger(closer to 0) the slope is, the better the basis quality improves.

3 Improved Progressive pnj-BKZ

In this section, we shall first give a sketch of *Improved Progressive pnj-BKZ* in section 3.1, where we use a two-step mode for solving SVP_γ. Next, we give a comparison among BKZ-only mode, the default mode in G6K and two-step mode to explain the benefit of two-step mode in section 3.2.

input : $\mathbf{B}$, β, $J = \text{jump}$, $f_{\text{extra}} = 12$;
output: $\mathbf{B}$;

$f \leftarrow \text{d4f}(\beta)$;
$\beta, f \leftarrow \beta + f_{\text{extra}}, f + f_{\text{extra}}$;
$\mathbf{B} \leftarrow \texttt{LLL(B)}$;
for $i \leftarrow 0$ **to** $\frac{d+2f-\beta}{J}$ **do**
  if $0 \leq i < \frac{f}{J}$ **then**
    $\kappa, \beta', f' \leftarrow 0,\ \beta - f + J \cdot i,\ J \cdot i$;
  else if $\frac{f}{J} \leq i < \frac{d-\beta+f}{J}$ **then**
    $j \leftarrow J \cdot i - f$;
    $\kappa,\ \beta',\ f' \leftarrow j,\ \beta,\ f$;
  else
    $j \leftarrow J \cdot i - (d - \beta + f)$;
    $\kappa,\ \beta',\ f' \leftarrow d - \beta + j,\ \beta - j,\ f - j$;
  $\mathbf{B}_{\pi[k:\beta'+k]} \cdot \mathbf{v}_i \leftarrow \texttt{Pump(B}, \kappa, \beta', f'\texttt{)}$;
  $\mathbf{B} = \texttt{LLL(B)}$;
return $\mathbf{B}$;

Algorithm 1: pnj-BKZ

3.1 Algorithm overview

Our Improved Progressive pnj-BKZ can be described as the following: input a basis $\mathbf{B}$, an optimized blocksize and jump strategy S for reducing $\mathbf{B}$, dimension d and target norm M, reduce $\mathbf{B}$ through a series of pnj-BKZ-(β, J). Each (β, J) is selected from blocksize and jump strategy S. The blocksize and jump strategy is generated from the blocksize and jump strategy selection algorithm (see section 5) and stored as a sequential list. To minimize the total cost, it will first use a series of pnj-BKZ to reduce the lattice basis properly, then at the suitable timing, it will call a Pump algorithm to find the target vector. The parameter selection of Pump follows Algorithm 6, which will lead to finding the approximate shortest vector after Pump. The detailed process is as Algorithm 2.

input : $\mathbf{B}$, S, M, $F(\star, \mathcal{D})$;
output: The approximate shortest vector $\mathbf{v}$;

$\mathbf{B} = \texttt{LLL(B)}$;
for $(\beta, J, \sharp\text{tours}) \in \mathsf{S}$ **do**
  for t **from** 1 **to** $\sharp$tours **do**
    $\mathbf{B} \leftarrow \texttt{pnj-BKZ}(\mathbf{B}, \beta, J)$;
$d_{\text{svp}}, \_ \leftarrow \texttt{ProSieveDimEst(rr(B)}/M,\ F(\star, \mathcal{D})\texttt{)}$; $f \leftarrow \text{d4f}(d_{\text{svp}})$;
$\mathbf{B} \leftarrow \texttt{Pump(B,}d - d_{\text{svp}},\ d_{\text{svp}},\ f\texttt{)}$;
return $\mathbf{v} \leftarrow \mathbf{b}_0$;

Algorithm 2: Improved Progressive pnj-BKZ

Our algorithm improves the heuristic SVP_γ solver in G6K from two aspects: firstly, we use a two-step mode with an optimal blocksize strategy; secondly, we choose the optimized jump(≥ 1) strategy instead of letting the jump value always equals to 1.

The experiments in [35] suggest that compared with the reduction strategy of jump $= 1$, the reduction strategy of jump $= 3$ is not beneficial. More precisely, [35] shows that the reduction strategy of jump $= 3$ requires similar running time to obtain the same quality of lattice basis, which is reduced by the strategy of jump $= 1$, with a larger memory consumption. However, while the jump strategy becomes larger, it can decrease the walltime to reach the same quality of lattice basis. Our experiment result shows that compared with the experiments results in [35], using a more efficient parallel computing implementation in [24], the acceleration effect of the strategy with jump > 1 on the improvement of lattice quality is indeed more obvious. More details can be found in Figure 1, which shows that the pnj-BKZ with the jump > 1 has a smaller time cost ($3 \sim 6$ times faster) while achieving the same reduction quality as that of the pnj-BKZ with the jump $= 1$. Therefore, to find the optimal pnj-BKZ reduction parameters, it is essential to construct a pnj-BKZ simulator to handle the case for jump > 1 which we will discuss in section 4.1.

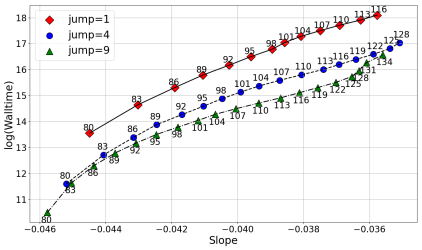


Fig. 1. Efficiency Speedup in Reduction by Jump strategy.
The walltime and slope are averaged over 5 instances for each algorithm. Each instance ran on a machine with 2 GPU and 32 threads. We label the point by pnj-BKZ blocksize β.

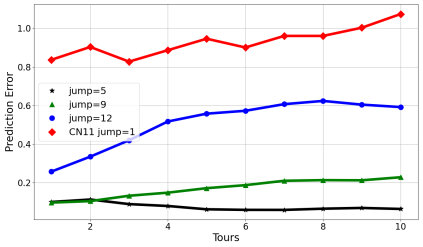


Fig. 2. Simulation error of pnj-BKZ simulator with different jump values
For each pnj-BKZ reduction parameter tested by 20 experiments to obtain the average length of Gram-Schmidt vectors.

3.2 Comparison among BKZ-only Mode, Default Mode in G6K and Two-step Mode

In this part, we introduce the three different modes for solving the SVP_γ problem and give an experiment to prove the comparison result.

BKZ-only Mode BKZ-only mode [27,36] implements multiple loops of BKZ-β/pnj-BKZ-$(\beta, J = 1)$ for solving SVP_γ problem. The pseudocodes of the BKZ-only Mode are as Algorithm 3, in which M denotes the upper bound of the Euclidean norm of the approximate shortest vector. It is a tunable attack since one can change the blocksize strategy to reduce the basis to a specific quality.

input : $\mathbf{B}$, d, M;
output: The approximate shortest vector $\mathbf{v}$;
$\mathbf{B} \leftarrow \texttt{LLL}(\mathbf{B})$;
for $i \leftarrow 1$ to ♯tours **do**
  $\mathbf{B} \leftarrow \texttt{BKZ}(\mathbf{B}, \beta)$ / $\texttt{pnj-BKZ}(\mathbf{B}, \beta, 1)$; // $\mathbf{B} = (\mathbf{b}_0, \cdots, \mathbf{b}_{d-1})$.
  if $\|\mathbf{b}_0\| < M$ **then**
    return $\mathbf{b}_0$;

Algorithm 3: BKZ-only Mode

Default LWE solving Mode in G6K In the default LWE solving mode of G6K [15], it solves the LWE problem by calling progressive pnj-BKZ and a conditional Pump (The algorithm calls Pump only if the estimated time cost of Pump is shorter than an upper bound) repeatedly. In the default LWE solving mode of G6K, it will reduce the basis by a specific blocksize strategy S_0[1]. After each lattice reduction by pnj-BKZ-$(\beta, J = 1)$, $\beta \in S_0$ and jump always equal to 1, the default LWE solving mode will record the time cost of the pnj-BKZ-$(\beta, J = 1)$ process and determine whether a Pump will finish in the same cost. If it does, it will call a Pump; If not, it will skip to the next pnj-BKZ-$(\beta, J = 1)$. The concrete process is as the Algorithm 4.

The benefit of the default LWE solving mode in G6K is that if we do not have an accurate simulator for BKZ/pnj-BKZ and we are not sure of the solvability by a final Pump calling, then a Default LWE solving Mode in G6K will make sure in outputting the required result in a reasonable time. However, without a simulator, it will sometimes enter a Pump with solving failure and waste processing time heavily since a Pump call is costly. Here a failed Pump means that the dimension setting of the sieving in the Pump is too optimistic, which makes the Pump fail to find the target vector. Besides, it might enter a Pump late and waste the processing time of extra cost for several pnj-BKZs with large blocksizes.

Two-step Mode The two-step mode was first proposed by [12], which calls a series of BKZ first for lattice reduction and calls an enumeration algorithm to find the target vector at last. In this paper, we use a two-step mode adapted

[1] $S_0 = \text{list}(\text{range}(10, 50)) + [\text{b} - 20, \text{b} - 17] + \text{list}(\text{range}(\text{b} - 14, \text{b} + 25, 2))$ in default mode

input : $\mathbf{B}$, S_0, d, M, $\gamma = 1.5$;
output: The approximate shortest vector $\mathbf{v}$;

$\mathbf{B} \leftarrow \texttt{LLL}(\mathbf{B})$;
for $\beta \in S_0$ **do**
  $\mathbf{B} \leftarrow \texttt{pnj-BKZ}(\mathbf{B}, \beta, J = 1)$;
  $\mathbf{B} \leftarrow \texttt{LLL}(\mathbf{B})$;
  $T^{\max}_{\text{pump}} \leftarrow T_{\text{pnjBKZ}}(\beta)$;
  $n_{\max}$ is the solution of the equation $T^{max}_{\text{pump}} \cdot \sharp\text{threads} = 2^{\frac{n-58}{2.85}}$;
  $n_{\text{expected}} \in \mathbb{N}$ is the minimum value such that $\sqrt{4/3} \cdot \text{GH}(\mathbf{B}_{[d-n_{\text{expected}}:d]}) \geq M/\gamma \cdot \sqrt{(n_{\text{expected}})/d}$;
  $\kappa \in \mathbb{N}$ is the maximal value such that $\text{GH}(\mathbf{B}_{[\kappa:d]}) \geq M \cdot \sqrt{(d-\kappa)/d}$;
  if $n_{\text{expected}} \leq n_{\max}$ **then**
    $f = \max\{d - \kappa - n_{\text{expected}}, 0\}$;
    $\mathbf{B} = \texttt{Pump}(\mathbf{B}, \kappa, d - \kappa, f)$;
  $\mathbf{B} \leftarrow \texttt{LLL}(\mathbf{B})$;
  if $\|\mathbf{b}_0\| < M$ **then**
    return $\mathbf{b}_0$; //$\mathbf{B} = (\mathbf{b}_0, \cdots, \mathbf{b}_{d-1})$

Algorithm 4: Default LWE Solving Mode in G6K

to pnj-BKZ and Pump. It calls a series of pnj-BKZ to reduce the basis first and finds a good timing to use a Pump algorithm to search the approximate shortest vector in the end, to solve the SVP_γ problem. The concrete process is as Algorithm 2. By our pnj-BKZ simulator Algorithm 5 and Pump sieving dimension estimation Algorithm 6 in section 4.1, we can guarantee that the last Pump outputs the required target vector.

Experiments of Comparison among BKZ-only Mode, Default Mode in G6K, and Two-step Mode In this part, we give an experimental result to illustrate that for solving SVP_γ, the two-step mode is the best mode among the three different modes.

To compare the two-step mode and the BKZ-only mode, we call a Pump-$(\kappa, d_{\text{svp}}, f)$ and a pnj-BKZ-β ($\beta < d_{\text{svp}}$) separately on the same lattice basis, where the Pump and the pnj-BKZ is in the same time cost, i.e. $T_{\text{pump}}(d_{\text{svp}}) = T_{\text{pnjBKZ}}(\beta)$. Table 1 shows that the reduced shortest vector $\mathbf{b}_0$ after a Pump is shorter than that after a pnj-BKZ. Thus, calling a Pump is more likely to find the shortest vector compared to a pnj-BKZ in the same time cost. On this account, it states that two-step mode is better than the BKZ-only mode.

Table 1. Simulated norm of $\mathbf{b}_0$ after a pnj-BKZ and a Pump under the same time cost.

$(n,\alpha)^\dagger$	κ	$\ln(\|\mathbf{b}_0\|^2)$	
		pnj-BKZ	Pump($\mathcal{L}_{[\kappa:d]}$)
(40,0.020)	34	14.64	12.05
(40,0.025)	42	14.73	12.55
(50,0.010)	53	15.49	11.85
(65,0.005)	85	16.70	11.42

† Basis from LWE instance (n, α) in TU Darmstadt LWE challenge.

Table 2. Basis quality estimation after a pnj-BKZ and a Pump under the same time cost.

(n,α)	Cost‡	$\log_2$(PSC)($\log_2$h)	
		pnj-BKZ	Pump($\mathcal{L}_{[\kappa:d]}$)
(50,0.025)	22.01	11.69	12.01
(55,0.020)	17.77	12.63	12.95
(75,0.010)	91.02	20.17	20.48
(90,0.005)	24.18	20.48	22.05

‡ Cost for calling the corresponding algorithm.

For the comparison between the two-step mode and G6K default mode, we show that an early Pump is less helpful in solving SVP_γ than an early pnj-BKZ. Let the time cost of the Pump for finding the target norm on the specific lattice basis, i.e. PSC, be a standard of measuring the quality of lattice reduced by different algorithms. A lattice basis with low PSC can be regarded as high quality. Separately call a pnj-BKZ/Pump on the same lattice basis with the same time cost. Table 2 shows that basis quality after a pnj-BKZ reduction is higher than that after a Pump reduction in the same time cost. The latter basis quality is estimated by the Pump sieving dimension estimation Algorithm 6. Thus, in the G6K-default mode, if it enters a Pump with no solution, the quality of returned lattice basis will be worse than that after a pnj-BKZ reduction in the same time limit. In conclusion, the G6K-default mode is less efficient than the two-step mode.

4 Simulator in Two-Step Solving Mode

In section 4.1, we first give the construction of the pnj-BKZ simulator and give a validation experiment on the performance of pnj-BKZ simulator is shown. Then we give the Pump cost model and the sieving dimension estimation of the last Pump in two-step solving mode in section 4.2.

4.1 Pnj-BKZ Simulator

The first step in the two-step solving mode is using a series well-chosen pnj-BKZ to reduce the lattice basis. To find the optimized reduction strategy of pnj-BKZ with jump > 1, we need an accurate pnj-BKZ simulator.

The pnj-BKZ Simulator Construction Before we give the construct detail of our pnj-BKZ simulator, let's briefly review the main idea of the BKZ simulator proposed in [10]. Suppose the input basis is $\mathbf{B} = (\mathbf{b}_0, \cdots, \mathbf{b}_{d-1})$ and its corresponding Gram-Schmidt basis is $\mathbf{B}^* = (\mathbf{b}_0^*, \cdots, \mathbf{b}_{d-1}^*)$. Let $\mathbf{B}'^* = \left(\mathbf{b}_0'^*, ..., \mathbf{b}_{d-1}'^*\right)$ be the corresponding output of the Gram-Schmidt basis reduced by one tour of BKZ-β. The BKZ simulator proposed in [10] will calculate $\text{Sim}\|\mathbf{b}_1'^*\| = \text{GH}(L_{[1:\beta]})$

$\approx \sqrt{\frac{\beta}{2\pi e}} \left(\prod_{i=1}^{\beta} \|\mathbf{b}_i^*\|\right)^{1/\beta}$ by Gaussian Heuristic. Then it will calculate $\mathrm{Sim}\|\mathbf{b}_2^{'*}\| = \mathrm{GH}(L'_{[2:\beta+1]}) \approx \sqrt{\frac{\beta}{2\pi e}} \left(\frac{\prod_{i=1}^{\beta+1}\|\mathbf{b}_i^*\|}{\mathrm{Sim}\|\mathbf{b}_1^{'*}\|}\right)^{1/\beta}$ by Gaussian Heuristic and the information of $\mathrm{Sim}\|\mathbf{b}_1^{'*}\|$ since the insert of new $\mathbf{b}_1$ will change the value of $\mathrm{Vol}(L_{[2:\beta+1]}) = \prod_{i=2}^{\beta+1}\|\mathbf{b}_i^*\|$ to $\mathrm{Vol}(L'_{[2:\beta+1]}) = \frac{\prod_{i=1}^{\beta+1}\|\mathbf{b}_i^*\|}{\|\mathbf{b}_1^{'*}\|}$. Here $\mathrm{Sim}\|\mathbf{b}_1^{'*}\|$ is a simulated approximate value of $\|\mathbf{b}_1^{'*}\|$ by Gaussian Heuristic. Iteratively calculating all remaining unknown $\mathrm{Sim}\|\mathbf{b}_i^{'*}\|$, such a simulator can predict the length value of each vector in $\mathbf{B}^{'*}$.

The BKZ 2.0 simulator [10] cannot be used directly to simulate the behavior of pnj-BKZ with jump > 1. We observe that when jump > 1, let J equals to the value of jump, every time after new $\mathbf{b}_i^*$ insert at the first position of the block $\mathbf{B}_{\pi_i[i:i+\beta']}$, the $J-1$ norms of Gram-Schmidt vectors $\mathbf{b}_{i+1}^*, ..., \mathbf{b}_{i+J-1}^*$ will change and remain unknown. These unknown norms prevent the BKZ 2.0 simulator [10] from predicting the norm of the first Gram-Schmidt vector in the next block. Our idea is that when jump > 1 we let each projected sub-lattice basis be reduced by a pump satisfying the property of HKZ reduction so that we can predict these unknown norms of Gram-Schmidt vectors between adjacent blocks.

Suppose the logarithms of the Gram-Schmidt norms $l_i = \ln(\|\mathbf{b}_i^*\|)$, $i \in \{0, \cdots, d-1\}$. Let $\mathbf{B}''$ and l_i'' be the corresponding output of the basis and the logarithm of each Gram-Schmidt norm after one tour of pnj-BKZ-(β, J). If the first J vectors in each block reduced by *Pump* satisfies HKZ reduced condition[2], then we can simulate each l_i after a pnj-BKZ-(β, J) reduction by Gaussian Heuristic as

$$\begin{cases} \mathrm{Sim}\left(l_i''\right) = \ln\left(\mathrm{GH}\left(L''_{\pi[i:i-(i \bmod J)+\beta]}\right)\right), & \text{if } i \in [0, d-\beta-1] \\ \mathrm{Sim}\left(l_i''\right) = \ln\left(\mathrm{GH}\left(L''_{\pi[i:d]}\right)\right), & \text{if } i \in [d-\beta, d-1] \end{cases}$$

Here $\ln\left(\mathrm{GH}\left(L''_{\pi[i:i-(i \bmod J)+\beta]}\right)\right) \approx$

$$\frac{1}{2}\ln\left(\frac{\beta-(i \bmod J)}{2\pi e}\right) + \frac{1}{\beta-(i \bmod J)}\left(\sum_{j=0}^{i-(i \bmod J)+\beta-1} l_j - \sum_{j=0}^{i-1}\mathrm{Sim}\left(l_j''\right)\right), \quad (4)$$

[2] To obtain such HKZ reduced basis, we should turn on *pump/down_sieve* during Pump process in pnj-BKZ, delete the condition "(pump.insert_left_bound <= kappa+down_stop)" in the file "https://github.com/WvanWoerden/G6K-GPU-Tensor/blob/main/g6k/algorithms/pump.py" to make sure the output basis projection satisfying HKZ reduction conditions as much as possible.

where $i \in \{0, \cdots, d-\beta-1\}$, $\mathrm{Vol}\left(L''_{\pi[i:i-(i \bmod J)+\beta]}\right) = \frac{\mathrm{Vol}\left(L_{[0:i-(i \bmod J)+\beta]}\right)}{\mathrm{Vol}\left(L''_{[0:i]}\right)}$. And for last β Gram-Schmidt norms

$$\ln\left(\mathrm{GH}\left(L''_{\pi[i:d]}\right)\right) \approx \frac{1}{2}\ln\left(\frac{d-i}{2\pi e}\right) + \frac{1}{d-i}\left(\sum_{j=0}^{d-1} l_j - \sum_{j=0}^{i-1}\mathrm{Sim}\left(l''_j\right)\right). \tag{5}$$

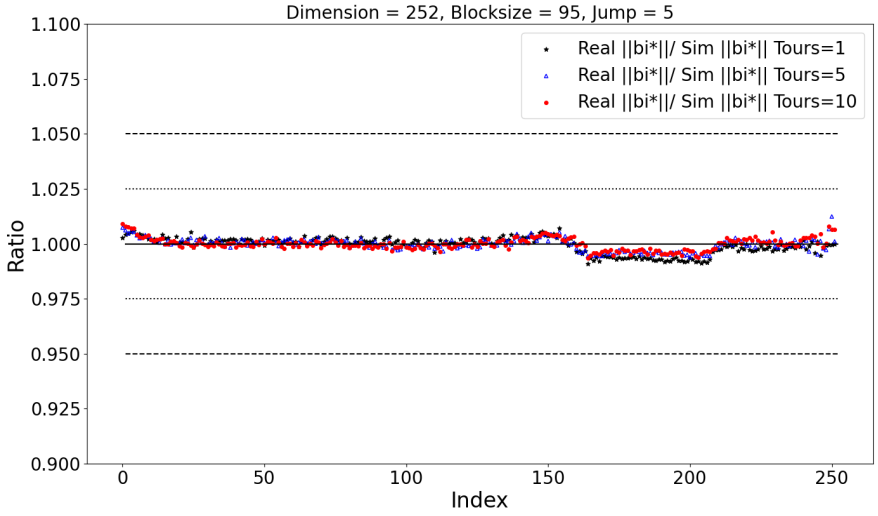
In other words, we only need to input the initial Gram-Schmidt norms $l_i = \ln\left(\|\mathbf{b}_i^*\|\right)$, $i \in \{0, \cdots, d-1\}$ of the lattice basis. Without performing pnj-BKZ-$(\beta,\ J)$ reduction, we can calculate $\mathrm{Sim}\left(l''_i\right)$ which are prediction values of l''_i by (4) and (5). Here l''_i are these actual Gram-Schmidt vector norms of lattice base after reducing by one tour of pnj-BKZ-$(\beta,\ J)$. Therefore, the reduction effect of the pnj-BKZ-$(\beta,\ J)$ algorithm can be predicted without actually running pnj-BKZ algorithm.

To show that it is reasonable for us to use the properties of the HKZ reduction basis to simulate the actual reduction effect of pnj-BKZ-$(\beta,\ J)$, we first illustrate that the Pump[2] output lattice basis is an almost HKZ reduced basis. For each tour, for $i \in [1, d]$, calculate the ratio $\frac{l''_i}{\mathrm{Sim}(l''_i)}$, see Figure 3. Here l''_i for $i \in [1, d]$ are the average logarithms of these Gram-Schmidt vector lengths obtained from 20 experiments, and $\mathrm{Sim}(l''_i)$ are logarithm of lengths of Gram-Schmidt vector which are calculated by (4) and (5).

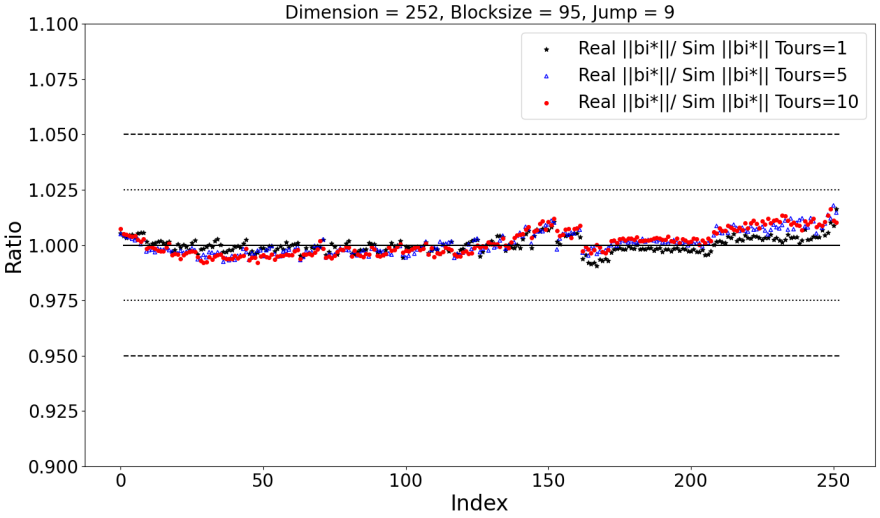
We calculate the simulation length of the Gram-Schmidt vector strictly according to the property of the HKZ reduction basis and Gaussian Heuristic. Therefore, in addition to being one of the criteria for measuring the accuracy of the pnj-BKZ simulator, this ratio $\frac{l''_i}{\mathrm{Sim}(l''_i)}$ can also be used as a criterion for judging whether the output basis of the Pump satisfies the property of HKZ reduction[2]. However, it can be seen from Figure 3 that when jump ≤ 9 even the tours increase to 10, the ratios $\frac{l''_i}{\mathrm{Sim}(l''_i)}$ are all between 0.975 and 1.025, indicating that the pnj-BKZ simulator uses (4) and (5) as the approximate estimate of the actual value $\|\mathbf{b}_i''^*\|$ and can already reflect how the average of the norms of Gram-Schmidt vectors change during each tour's reduction of pnj-BKZ-$(\beta,\ J)$.

Besides, when simulating the length value of Gram-Schmidt vector, we note that while $i \equiv 1\ (mod\ J)$ the index i of $\mathrm{GH}\left(L''_{[i:i+\beta-(i \bmod J)]}\right)$ is the same as that in $\mathrm{GH}\left(L'_{[i:i+\beta-1]}\right)$, however, the simulation volumes of projected sublattice $L'_{[i:i+\beta-1]}$ and $L''_{[i:i+\beta-1]}$ are different. Because $\mathrm{Vol}(L'_{[i:i+\beta-1]}) = \frac{\prod_{j=1}^{i+\beta-1}\|\mathbf{b}_j^*\|}{\prod_{j=1}^{i-1}\|\mathbf{b}_j'^*\|}$ here $\|\mathbf{b}_j'^*\| := \mathrm{GH}\left(L'_{[j:j+\beta-1]}\right)$ in BKZ 2.0 simulator [10]. While $\mathrm{Vol}(L''_{[i:i+\beta-1]}) = \frac{\prod_{j=1}^{i+\beta-1}\|\mathbf{b}_j^*\|}{\prod_{j=1}^{i-1}\|\mathbf{b}_j''^*\|}$ here $\|\mathbf{b}_j''^*\|$ calculated by (4) and (5) in pnj-BKZ simulator. We give a detailed algorithm description of the pnj-BKZ simulator in the Algorithm 5.

Performance of pnj-BKZ simulator We give an experiment to verify the effectiveness of our pnj-BKZ simulator in this part.



(a) Jump=5



(b) Jump=9

Fig. 3. ratio $\frac{l_i''}{\mathrm{Sim}(l_i'')}$, $\beta = 95$.

Run 10 tours ($\beta = 95$, $J = 5$ and $\beta = 95$, $J = 9$ respectively) of pnj-BKZ reduction on a 252 dimension lattice basis, and record the output of Gram-Schmidt vector lengths each tour. For each pnj-BKZ reduction parameter, we did experiments 20 times to obtain the average length of Gram-Schmidt vector.

input : $\mathsf{rr} = (l_0, \cdots, l_{d-1})$, blocksize $\beta \in \{45, \cdots, d\}$, number of tours t and size of jump J to simulate.

output: A prediction for the logarithms of the Gram-Schmidt norms $l'_i = \ln(\|\mathbf{b}'^*_i\|)$ after t tours pnj-BKZ-β reduction with jump is J.

for $i \leftarrow 0$ **to** 44 **do**
  $r_i \leftarrow$ average $\ln(\|\mathbf{b}^*_i\|)$ of a HKZ reduced random unit-volume 45-dimensional lattice;
for $i \leftarrow 45$ **to** β **do**
  $c_i \leftarrow \ln\left(\mathrm{V}_i(1)^{-1/i}\right) = \ln\left(\frac{\Gamma(i/2+1)^{1/i}}{\pi^{1/2}}\right)$;
for $j \leftarrow 0$ **to** $t-1$ **do**
  $\mathsf{flag} \leftarrow$ true; //flag to store whether $L_{[k,d]}$ has changed
  for $k \leftarrow 0$ **to** $d-\beta-1$ **do**
    $\beta' \leftarrow \min(\beta, d-k+1)$; //Dimension of local block
    $h \leftarrow \min(k-(k \bmod \mathrm{J})+\beta-1, d)$;
    $\ln(\mathrm{V}) \leftarrow \sum_{i=0}^{h} l_i - \sum_{i=0}^{k-1} l'_i$;
    if $\mathsf{flag} = \mathsf{True}$ **then**
      if $\ln(\mathrm{V})/(\beta'-(k \bmod \mathrm{J})) + c_{\beta'-(k \bmod \mathrm{J})} < l_k$ **then**
        $l'_k \leftarrow \ln(\mathrm{V})/(\beta'-(k \bmod \mathrm{J})) + c_{\beta'-(k \bmod \mathrm{J})}$;
        $\mathsf{flag} \leftarrow \mathsf{False}$;
    else
      $l'_k \leftarrow \ln(\mathrm{V})/(\beta'-(k \bmod \mathrm{J})) + c_{\beta'-(k \bmod \mathrm{J})}$;
  for $k \leftarrow d-\beta$ **to** $d-46$ **do**
    $\beta' \leftarrow d-k$; $h \leftarrow d$; $\ln(\mathrm{V}) \leftarrow \sum_{i=1}^{h} l_i - \sum_{i=1}^{k-1} l'_i$;
    if $\mathsf{flag} = \mathsf{True}$ **then**
      if $\ln(\mathrm{V})/\beta' + c_{\beta'} < l_k$ **then**
        $l'_k \leftarrow \ln(\mathrm{V})/\beta' + c_{\beta'}$; $\mathsf{flag} \leftarrow false$;
    else
      $l'_k \leftarrow \ln(\mathrm{V})/\beta' + c_{\beta'}$;
  $\ln(\mathrm{V}) \leftarrow \sum_{i=1}^{h} l_i - \sum_{i=1}^{k-1} l'_i$;
  for $k \leftarrow d-45$ **to** $d-1$ **do**
    $l'_k \leftarrow \frac{\ln(\mathrm{V})}{45} + r_{k+45-d}$;
  for $k \leftarrow 0$ **to** $d-1$ **do**
    $l_k \leftarrow l'_k$;
return $l_0, \cdots, l_{d-1}$;

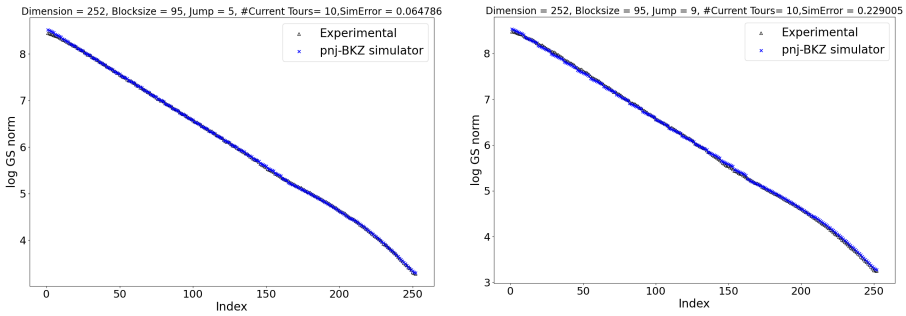
Algorithm 5: pnj-BKZ Simulator

To measure the accuracy of pnj-BKZ simulator, SimError calculated as the equation (6)

$$\text{SimError}(\sharp\text{tours}) = \sum_{i=0}^{d-1} \left(\|\mathbf{b}_i^*\|_{(\sharp\text{tours})} - \text{Sim}(\|\mathbf{b}_i^*\|)_{(\sharp\text{tours})} \right)^2, \tag{6}$$

where $\sharp$tours represents the number of current tours and $\text{Sim}(\|\mathbf{b}_i^*\|)_{(\sharp\text{tours})}$ are the lengths of Gram-Schmidt vectors predicted by pnj-BKZ simulator with $\sharp$tours.

As can be seen in the Figure 4, pnj-BKZ simulator can well predict the behavior of the pnj-BKZ algorithm when jump within $[2, 9] \cap \mathbb{Z}$ since the corresponding SimError($\sharp$tours) is small. Figure 2 shows that as the jump value increases, the prediction error increases, and the accuracy of pnj-BKZ simulator decreases. However, the prediction error SimError($\sharp$tours) is still within 1 especially when the jump takes within $[2, 12]$ and the SimError($\sharp$tours) of pnj-BKZ simulator is not bigger than that of BKZ 2.0 simulator. Therefore the pnj-BKZ simulator can predict how the average norms of Gram-Schmidt vectors change during each tour reduction of pnj-BKZ-(β, J).



(a) $\beta = 95, jump = 5, \sharp\text{tours} = 10$ (b) $\beta = 95, jump = 9, \sharp\text{tours} = 10$

Fig. 4. Prediction effect of pnj-BKZ simulator.

To verify the effectiveness of our pnj-BKZ simulator, we perform the following experiments, reducing $(n = 70, \alpha = 0.005)$ LWE challenge lattice basis by pnj-BKZ with reduction parameter: blocksize $\beta = 95$, jump size $J \in [1, \cdots, 16]$, $\sharp$tours $\in [1, \cdots, 10]$. Here under the same reduction parameters, we do 20 times experiments.

See Appendix B for more details about the prediction effect of the simulator under different jump values and different tours.

4.2 Pump Cost Model and Dimension Estimation of the Last Pump

Pump Cost Model We set T_{pump} as the computational cost of a Pump since the main cost of a Pump can be regard as a $(\beta - f)$-dimensional progressive

sieve, i.e.

$$\begin{aligned} T_{\text{pump}}(\beta) &= \sum_{j=\beta_0}^{\beta} T_{\text{sieve}}(j) = \sum_{j=\beta_0}^{\beta-f} 2^{c\cdot j+o(j)} = 2^{c\beta_0}\left(1+2^c+\cdots+2^{c(\beta-f-\beta_0)}\right) \\ &\leq 2^{c\beta_0}\cdot\frac{2^{c(\beta-f+1)+o(\beta-f+1)}}{1-2^c} = O\left(2^{c(\beta-f)}\right) \approx 2^{c(\beta-f)+c_1}, \end{aligned} \tag{7}$$

where β_0 is the dimension of initial sieving in Pump (In G6K β_0 is set to 30, and in G6K-GPU, it is set to 50), c and c_1 are the coefficients of the full sieve cost related to sieve dimension, $T_{\text{sieve}}(j)$ is the sieve cost in dimension j with dim-for-free value $f = \text{d4f}(j)$. More detail about our practical Pump cost model can be seen in the Appendix C.

Dimension Estimation of the Last Pump Except for the pnj-BKZ simulator, to solve LWE with high success probability we need to determine the dimension of sieving in the last Pump. In G6K, a Pump sieving dimension estimation method has been given. Let $M_{d_{\text{svp}}} = M\cdot\sqrt{d_{\text{svp}}/d}$ be the expected norm of the required approximate shortest vector projection in the sub-lattice $\mathcal{L}_{\pi[d-d_{\text{svp}}:d]}$ and d_{svp} be the maximal dimension to sieve while calling Pump, where $\text{GH}(rr_{[d-d_{\text{svp}}:d]}) = \text{GH}(\mathbf{B}_{\pi[d-d_{\text{svp}}:d]}) \leq M_{d_{\text{svp}}}$.

We give a new dimension of sieving estimation of Pump named ProSieveDimEst (Algorithm 6) here to estimate the selected dimension of the progressive sieve in the end. In this algorithm, $F(\beta,\mathcal{D})$ be a distribution function related to the projected lattice $\mathcal{L}_{\pi[d-\beta:d]}$ and $\mathcal{D}$ describes the distribution of the elements in the projected target vector $\mathbf{v}\in\mathcal{L}_{\pi[d-\beta:d]}$. For instance, when solving standard form LWE the target vector contains the noise vector $\mathbf{e}\in\mathbb{Z}_q^m$ from a discrete Gaussian distribution D_σ whose standard deviation is σ, then $F(\beta,\mathcal{D}) = \sigma^2\chi_\beta^2$. Algorithm 6 will evaluates the maximal sieve dimension d_{svp} for calling Pump within a high probability(≥ 0.999) to find the target vector, then give a cost estimation by d_{svp}. Let $T_{\text{pump}}(\beta) = \sum_{j=\beta_0}^{\beta} T_{\text{sieve}}(j)$ as equation (7). Since Pump contains the progressive sieve process, we should consider the failure/success probability during the process. Let the success probability of sieve-β be $P_{\text{suc}}(\beta)$, then its failure probability is $1-P_{\text{suc}}(\beta)$. We call the sieve-β only if sieve-$(\beta-1)$ didn't find the target vector, so the expected cost of sieve-β is $T_{\text{sieve}}(\beta)(1-P_{\text{suc}}(\beta-1))$. Iterate β from β_0 to d_{svp}, the expected Pump cost

$$\begin{aligned} \text{PSC}(d_{\text{svp}}) &= \sum_{\beta=\beta_0}^{d_{\text{svp}}} \left[T_{\text{sieve}}(\beta)\cdot(1-P_{\text{suc}}(\beta-1))\right] \\ &= \sum_{\beta=\beta_0}^{d_{\text{svp}}} \left[T_{\text{pump}}(\beta)\cdot(P_{\text{suc}}(\beta)-P_{\text{suc}}(\beta-1))\right], \end{aligned} \tag{8}$$

wher $P_{\text{suc}}(d_{\text{svp}}) \geq 0.999$.

input : rr, $F(\star, \mathcal{D})$, $P_{\text{success}} = 0.999$;
output: d_{svp}, PSC;
for $d_{\text{svp}} \leftarrow d_{\text{start}}$ **to** d **do**
  $P_{\text{suc}}(d_{\text{svp}}) \leftarrow \Pr\left[x \leftarrow F(d_{\text{svp}}, \mathcal{D}) \middle| x \leq \left(\text{GH}(rr_{[d-d_{\text{svp}}:d]})\right)^2\right]$;
  if $P_{\text{suc}}(d_{\text{svp}}) > P_{\text{success}}$ **then**
    return d_{svp}, PSC(d_{svp});

Algorithm 6: ProSieveDimEst

4.3 Estimator Stability Assumption

In this part, we give a heuristic assumption about our strategy estimator which will be discussed in detail in Section 5. In practice, although the sieving algorithm has a certain randomness, the time cost of the Pump, pnj-BKZ-(β, J) and the reduction effect of a series of pnj-BKZ-(β, J) are stable. Same as the length of Gram-Schmidt vectors of BKZ-β fully reduced basis will follow GSA (Definition 4) and BKZ 2.0 simulator can well simulate how the length of Gram-Schmidt vectors changes during every tour of BKZ-β's reducing. Therefore, we give the Heuristic 1.

Heuristic 1 *The practical time cost model (Appendix C) and pnj-BKZ simulator (Alg.5) fit the actual time cost of the pump algorithm and pnj-BKZ algorithm and the reduction effect of a series of pnj-BKZ-(β, J) for almost all lattices.*

5 Improved Progressive pnj-BKZ: Blocksize and Jump Strategy Optimization

In this section, we describe the two blocksize and jump strategy selection algorithms in detail. The first is *blocksize and jump strategy selection algorithm based on pro-BKZ* (BSSA), based on the blocksize selection strategy in Improved Progressive BKZ [12]. The second is a new algorithm called *blocksize and jump strategy enumeration* (EnumBS), through which we can get an optimal blocksize and jump strategy. We give both formal proof and Experimental results to show that our new strategy selection algorithm is better than the algorithm based on pro-BKZ.

5.1 Blocksize and Jump Strategy Selection Algorithm based on Pro-BKZ

The *blocksize and jump strategy selection algorithm based on pro-BKZ* (BSSA) can be described as an application of the shortest path algorithm on blocksize and jump strategy selection. To store the variables in the BSSA process, we define a blocksize and jump strategy dictionary BS, in which the key is for each middle β^{goal} node and the value is a tuple of $\mathsf{bs} = (\mathsf{rr}, \mathsf{S}, G_{\text{BKZ}})$, where

rr is the length of Gram-Schmidt vector which is fully BKZ-β^{goal} reduced, S means the blocksize and jump selection strategy which will improve the quality of lattice basis from fully BKZ-β^{start} reduced to fully BKZ-β^{goal} reduced and G_{BKZ} denotes simulated time cost for strategy S.

BSSA firstly sets several middle nodes of β_i as a measure of basis quality and expects to find the optimal blocksize and jump $(\beta^{\text{alg}}, J^{\text{alg}})$ with tours t between each node whose reduction cost is G_{BKZ}, which can be used to minimize the time cost of improving the quality of lattice basis from fully BKZ-β^{start} reduced to fully BKZ-β^{goal} reduced and store the reduction strategy $(\beta^{\text{alg}}, J^{\text{alg}})$, simulated reduced GS-lengths and cost for the strategy as $\mathsf{BS}[\beta^{\text{start}}+1] = ([(\beta^{\text{alg}}, J^{\text{alg}}, t)], \mathsf{rr}, G_{\text{BKZ}})$. Given β^{start}, β, β^{goal} and $\beta^{\text{start}} < \beta < \beta^{\text{goal}}$, suppose we have found the best single pnj-BKZ strategy from β^{start} to β, then we could try to find the strategy with minimal cost from from β^{start} to β^{goal} by looking for another pnj-BKZ-$(\beta^{\text{alg}}, J^{\text{alg}})$ from β^{start} to β^{goal} or considering the minimal cost of pnj-BKZ-$(\beta^{\text{alg}}, J^{\text{alg}})$ from β^{start} to β and β to β^{goal} separately and find the minimal total cost from β^{start} to β^{goal}. Let $\beta^{\text{start}} = \beta_0$, repeat the above steps from $\beta_0 + 1$ to $\beta^{\text{goal}} = d$, we will find the minimal cost from β_0 to each β that $\beta_0 + 1 \leq \beta \leq d$.

To find the efficient two-step solving strategy, we should consider the cost of the last Pump, i.e. PSC, after several pnj-BKZ reductions. By setting different final β^{goal}, we can get different reduction strategy BS which improving the quality of lattice basis from β^{start} to β^{goal} and different sieving dimension of the last Pump corresponding to different quality of lattice which is fully β^{goal} reduced. Then we set multiple different final β^{goal} to choose the two-step solving strategy whose total time cost is the minimal. Here total time cost includes the time cost of improving the quality of lattice by a series pnj-BKZ-$(\beta, J) \in \mathsf{BS}$ and the time cost of final Pump. See Algorithm 7 for more detail about BSSA.

Since BSSA uses the quality of the BKZ-β fully reduced basis as the intermediate node to find the optimized block strategy, the BSSA strategy with more flexible intermediate node settings can be constructed by the slope of the length of Gram-Schmidt vectors or combining the root Hermit factor value δ and geometry series assumption. We plan to finish this work in the future.

5.2 Blocksize and Jump Strategy Enumeration Algorithm

In this part we will introduce another blocksize and jump strategy selection algorithm.

In the case where Heuristic 1 holds we can prove that there is an algorithm that can find the optimal Blocksize and Jump Strategy of a series of pnj-BKZ-(β, J) algorithm for improving the quality of lattice basis and the optimal dimension of Pump to find the target vector with a high probability. We design such an algorithm named *blocksize and jump strategy enumeration algorithm* (EnumBS) which is a pruning enumeration algorithm that enumerates all the possible blocksize and jump strategies and we will describe the proposed algorithm in detail and prove it optimality in this section. The detailed description of EnumBS can be seen in Algorithm 8.

input : $\mathsf{rr}_0 = (l_0, \cdots, l_{d-1})$, $F(\star, \mathcal{D})$, $\beta_0 = 50$, $J_{\max}(\star) \leftarrow \min\{\mathrm{d4f}(\star), 0.1 \cdot (\star)\}$;
output: $G_{\min}$, $B_{\min}$, $S_{\min}$;

$d \leftarrow \mathrm{len}(\mathsf{rr}_0)$; $\mathsf{BS}[\beta^{\mathrm{start}}] = (\mathsf{rr}_0, [\,], 0)$;
for $\beta \leftarrow \beta_0$ **to** d **do**
  $G^{\mathrm{BKZ}}_{\min} \leftarrow +\infty$;
  for $\beta^{\mathrm{sstart}} \leftarrow \beta^{\mathrm{start}}$ **to** $\beta - 1$ **do**
    $\mathsf{bs}^{\mathrm{sstart}} \leftarrow \mathsf{BS}\,[\beta^{\mathrm{sstart}}]$; $\mathsf{bs} \leftarrow (\emptyset, \emptyset, +\infty)$;
    $\sharp$tours $\leftarrow$ the maximal tours for pnj-BKZ-β;
    $\mathsf{rr}^* \leftarrow$ `PnjBKZSim`$(\mathsf{bs}^{\mathrm{sstart}}.\mathsf{rr}, \beta, 1, \sharp\mathrm{tours})$;
    $(d^*_{\mathrm{svp}}, \mathrm{PSC}^*) \leftarrow$ `ProSieveDimEst`$(\mathsf{rr}^*, F(\star, \mathcal{D}))$;
    for $\beta^{\mathrm{alg}} \leftarrow \beta + 1$ **to** d **do**
      for $j \leftarrow J_{\max}(\beta^{\mathrm{alg}})$ **to** 1 **do**
        if $\frac{d-\beta^{\mathrm{alg}}}{j} \cdot T_{\mathrm{pump}}(\beta^{\mathrm{alg}}) > \mathsf{bs}.G_{\mathrm{BKZ}}$ **then**
          break; // It means a larger beta cannot decrease cost.
        $G' \leftarrow +\infty$;
        $\sharp$tours $\leftarrow$ the maximal tours for pnj-BKZ-$(\beta^{\mathrm{alg}}, j)$;
        for $t \leftarrow 1$ **to** $\sharp$tours **do**
          $\mathsf{rr}' \leftarrow$ `PnjBKZSim`$(\mathsf{bs}^{\mathrm{sstart}}.\mathsf{rr},\ \beta^{\mathrm{alg}},\ j,\ t)$;
          $G_{\mathrm{BKZ}}(\beta, j) \leftarrow t \cdot \frac{d-\beta}{j} \cdot T_{\mathrm{pump}}(\beta)$;
          $(d'_{\mathrm{svp}}, \mathrm{PSC}') \leftarrow$ `ProSieveDimEst`$(\mathsf{rr}^*, F(\star, \mathcal{D}))$;
          if $\mathrm{PSC}' \leq \mathrm{PSC}^*$ **then**
            $G' \leftarrow G_{\mathrm{BKZ}}(\beta, j)$;
            break;
        if $\mathsf{bs}.G_{\mathrm{BKZ}} > G'$ **then**
          $\mathsf{bs} \leftarrow (\mathsf{bs}^{\mathrm{sstart}}.\mathsf{S} \cup [(\beta^{\mathrm{alg}}, j, t)], \mathsf{rr}', \mathsf{bs}^{\mathrm{sstart}}.G_{\mathrm{BKZ}} + G')$;
    if $G^{\mathrm{BKZ}}_{\min} > \mathsf{bs}.G_{\mathrm{BKZ}}$ **then**
      $G^{\mathrm{BKZ}}_{\min} \leftarrow \mathsf{bs}.G_{\mathrm{BKZ}}$; $\mathsf{BS}\,[\beta] \leftarrow \mathsf{bs}$;
$G_{\min} \leftarrow +\infty$, $S_{\min} \leftarrow \emptyset$;
for $\beta \leftarrow \beta^{\mathrm{start}}$ **to** d **do**
  $\mathsf{bs} \leftarrow \mathsf{BS}[\beta]$; $(d_{\mathrm{svp}}, \mathrm{PSC}) \leftarrow$ `ProSieveDimEst`$(\mathsf{bs}.\mathsf{rr},\ F(\star, \mathcal{D}))$;
  $G \leftarrow \mathsf{bs}.G_{\mathrm{BKZ}} + \mathrm{PSC}$; $\sharp\mathsf{S} \leftarrow \mathrm{len}(\mathsf{bs}.\mathsf{S})$;
  if $G < G_{\min}$ **then**
    $B \leftarrow \mathrm{bit}_{\mathrm{pump}}(\max\{\mathsf{bs}.\mathsf{S}[\sharp\mathsf{S}], d_{\mathrm{svp}}\})$;
    $G_{\min}$, $B_{\min}$, $\mathsf{S}_{\min} \leftarrow G$, B, S;
return $G_{\min}$, $B_{\min}$, $\mathsf{S}_{\min}$;

Algorithm 7: BSSA

In EnumBS, we use BS to store the information of each reduction strategy which might be the optimal strategy or will become the optimal strategy after adding more (β, J) nodes. BS is a list and each element bs in the BS is a tuple of values $\mathsf{bs} = (\mathsf{rr}, \mathsf{S}, G_{\mathrm{BKZ}}, d_{\mathrm{svp}}, \mathrm{PSC})$. It is important to note that each bs in BS should be in order that increases by its G_{BKZ} value and decreases by its PSC value. In bs, S is a list storing the blocksize and jump strategy calling pnj-BKZ, G_{BKZ} is the time cost for calling such a series of pnj-BKZ, rr stores the current simulated gs-lengths after calling pnj-BKZ following strategy S by calling pnj-BKZ simulator (Alg.5), d_{svp} and PSC is the parameters related to last Pump by calling a Pump dimension estimation method (Alg.6), in which d_{svp} denotes the maximal dimension to be selected in last Pump and PSC is estimated time cost for last Pump. Each element in S is a tuple (β, J, t), where β is block size value of pnj-BKZ, J is the jump value of pnj-BKZ and t is the number of tours for calling pnj-BKZ-(β, J). For the sake of narrative simplicity, we will use $\mathsf{bs}.\star$ to denote each element in bs, e.g. $\mathsf{bs}.\mathsf{S}$. Let $\sharp\mathsf{S}$ and $\sharp\mathsf{BS}$ be the length of S and BS.

At the start of EnumBS, there is only one tuple $\mathsf{bs}^{(0)}$ in BS, where $\mathsf{bs}^{(0)}.\mathsf{S} = [\,]$ denotes a no pnj-BKZ block size strategy with a pure Pump sieve. The total cost of $\mathsf{bs}^{(0)}$ is the Pump cost. Then, to generate more strategies and try to find the optimal strategy, we can regard $\mathsf{bs}^{(0)}$ as the root node and expand the strategy list from $\mathsf{bs}^{(0)}$. Let the relation between $\mathsf{bs}^*.\mathsf{S} = \mathsf{bs}.\mathsf{S} \cup [(\beta, J, t)]$ and $\mathsf{bs}.\mathsf{S}$ be the child strategy and the father strategy, where the (β, J) value should be larger than the largest block size strategy in $\mathsf{bs}.\mathsf{S}$ (i.e. $\beta > \mathsf{bs}.\mathsf{S}[\sharp S]$ or $(\beta = \mathsf{bs}.\mathsf{S}[\sharp S].\beta$ and $j < \mathsf{bs}.\mathsf{S}[\sharp S].J)$) to ensure the reducibility after adding a new (β, J, t). We call such a bs^* a child node of bs and the corresponding bs is the father node. Considering each child strategy $\mathsf{bs}^*.\mathsf{S}$ of $\mathsf{bs}^{(0)}.\mathsf{S}$ for all possible (β, J, t), compute the other values in bs^*, i.e. $\mathsf{bs}^*.G_{\mathrm{BKZ}}$, $\mathsf{bs}^*.\mathsf{rr}$, $\mathsf{bs}^*.d_{\mathrm{svp}}$ and $\mathsf{bs}^*.\mathrm{PSC}$. When we try to add a bs^* into BS, we should first determine whether it exists a $\mathsf{bs} \in \mathsf{BS}$ so that $\mathsf{bs}^*.\mathrm{PSC} \geq \mathsf{bs}.\mathrm{PSC}$ and $\mathsf{bs}^*.G_{\mathrm{BKZ}} \geq \mathsf{bs}.G_{\mathrm{BKZ}}$. If so, we cannot add such bs^* into BS, because the child strategies generated by bs^*(including bs^* itself) won't have a shorter time overhead than which generated by the corresponding bs. If not, then we should first add bs^* and then delete the bad strategy in BS whose PSC value and G_{BKZ} value are both larger than bs^*. Iterate each $\mathsf{BS}[k]$ and its child nodes sequentially, and we'll end up with a BS containing the optimal block size strategy. Iterate through BS and return the optimal strategy in the end.

Proof for the Optimality of EnumBS As we mention at start of Section 5.2, EnumBS is an algorithm that can find the optimal block size strategy in two-step mode. Theorem 1 proves its optimality.

Theorem 1. *Let* $\mathsf{bs}.G = \mathsf{bs}.G_{\mathrm{BKZ}} + \mathsf{bs}.\mathrm{PSC}$ *be the total cost of a two-step reduction using parameters in* bs *and* $\mathcal{S}$ *be a set containing all possible strategies. It exists an optimal strategy* $\mathsf{bs}_{\mathsf{op}}.\mathsf{S} \in \mathcal{S}$ *s.t.* $bs_{\mathsf{op}}.G$ *is minimal. The output strategy* $\mathsf{bs}_{\mathsf{EnumBS}}.\mathsf{S}$ *of EnumBS is the optimal strategy, i.e.* $\mathsf{bs}_{\mathsf{EnumBS}}.\mathsf{S} = \mathsf{bs}_{\mathsf{op}}.\mathsf{S}$.

Proof. (Proof by contradiction) Suppose there is a strategy $\mathsf{bs}'.\mathsf{S} \in \mathcal{S}$ such that $\mathsf{bs}'.G < \mathsf{bs}_{\mathsf{EnumBS}}.G$. Then, there are four different situations:

If $\mathsf{bs}'.\mathrm{PSC} < \mathsf{bs}_{\mathsf{EnumBS}}.\mathrm{PSC}$, i.e. the cost of last Pump in bs' is shorter than which of S_{EnumBS}, then it exists the following two situations.

(1) If $\mathsf{bs}'.G_{\mathrm{BKZ}} \leq \mathsf{bs}_{\mathsf{EnumBS}}.G_{\mathrm{BKZ}}$, then $\mathsf{bs}_{\mathsf{EnumBS}}$ will be replaced by bs' in the EnumBS algorithm, contradictory.

(2) If $\mathsf{bs}'.G_{\mathrm{BKZ}} > \mathsf{bs}_{\mathsf{EnumBS}}.G_{\mathrm{BKZ}}$, one of the child strategies of bs' or bs' it self. Since $\mathsf{bs}'.G < \mathsf{bs}_{\mathsf{EnumBS}}.G$, the EnumBS won't output the strategy $\mathsf{bs}_{\mathsf{EnumBS}}$, contradictory.

If $\mathsf{bs}'.\mathrm{PSC} \geq \mathsf{bs}_{\mathsf{EnumBS}}.\mathrm{PSC}$, i.e. the cost of last Pump in bs' is larger than which of S_{EnumBS}, then it exists the following two situations.

(3) If $\mathsf{bs}'.G_{\mathrm{BKZ}} \geq \mathsf{bs}_{\mathsf{EnumBS}}.G_{\mathrm{BKZ}}$, then $\mathsf{bs}'.G \geq \mathsf{bs}_{\mathsf{EnumBS}}.G$, contradictory to the assumption.

(4) If $\mathsf{bs}'.G_{\mathrm{BKZ}} < \mathsf{bs}_{\mathsf{EnumBS}}.G_{\mathrm{BKZ}}$, then one of the child strategies of bs' or bs' it self will appear in the BS to the last. Since $\mathsf{bs}'.G < \mathsf{bs}_{\mathsf{EnumBS}}.G$, EnumBS won't output the strategy $\mathsf{bs}_{\mathsf{EnumBS}}.\mathsf{S}$, contradictory.

Consequently, $\mathsf{bs}_{\mathsf{EnumBS}}.\mathsf{S}$ is the optimal strategy. □

6 Applying Pro-pnj-BKZ to LWE

In section 6.1, we will explain the method of choosing the upper bound of the sieving dimension of the last Pump in two-step mode for solving LWE. It is different from that used in the default G6K's LWE solving strategy, which we think is over-optimistic. The over-optimistic estimation for the sieving dimension of the last Pump in default G6K's LWE solving strategy will make the Pump fail to find the target vector and lead to a bad reduction. In section 6.2 we give the optimized blocksize and jump strategy for solving TU Darmstadt LWE challenge. In section 6.3, we apply our algorithm to solve TU Darmstadt LWE Challenges and we will give a comparison of different LWE-solving modes by experiments and simulation respectively. In section 6.4, we show these new TU Darmstadt LWE Challenges we solved. In section 6.5, we give a new security estimation of LWE in NIST schemes [37] based on the optimized blocksize and jump selection and two-step solving mode strategy .

6.1 Estimation of sieving dimension in Pump for solving LWE

In this part, we reconsider the relationship between the dimension of sieving in the last Pump and the probability of finding the target vector in LWE. We present a new method for determining these parameters in the final high-dimensional sieving in Pump based on the distribution of the square of length of LWE error vector, which is a chi-squared distribution.

The value estimation of d_{svp} in default G6K cannot always guarantee finding the target vector $\mathbf{t}$. It uses the inequality

$$\sigma\sqrt{d_{\mathrm{svp}}} \approx \left\|\pi_{d-d_{\mathrm{svp}}}(\mathbf{t})\right\| \leq \mathrm{GH}(\mathbf{B}_{[d-d_{\mathrm{svp}}]}), \tag{9}$$

which is relevant to the expected length of the target vector and current shortest vector in the projection of the sub-lattice, to determine the upper bound of

input : $\mathsf{rr}_0 = (l_0, \cdots, l_{d-1})$, $F(\star, \mathcal{D})$, $J_{\max}(\star) \leftarrow \min\{\mathrm{d4f}(\star), 0.1 \cdot (\star)\}$;
output: $S_{\min}$, $G_{\min}$, $B_{\min}$;

$k \leftarrow 1$; $d \leftarrow \mathrm{len}(\mathsf{rr}_0)$; $d^{(0)}_{\mathrm{svp}}, \mathrm{PSC}^{(0)} \leftarrow$ `ProSieveDimEst`$(\mathsf{rr}_0,\ F(\star, \mathcal{D}))$;
$\mathsf{BS} \leftarrow \{(\mathsf{rr}_0, [\,], 0, d^{(0)}_{\mathrm{svp}}, \mathrm{PSC}^{(0)})\}$; $\sharp\mathsf{BS} \leftarrow \mathrm{len}(\mathsf{BS})$; $\mathsf{bs}^* \leftarrow \mathsf{BS}[0]$;
while $k < \sharp\mathsf{BS}$ **do**
  $\mathsf{bs} \leftarrow \mathsf{BS}\ [k]$;
  if $\mathsf{bs.S} = [\,]$ **then**
    $(\beta^{\mathrm{start}}, j^{\mathrm{start}}) \leftarrow (\beta_0, J_{\max}(\beta_0)$;
  else
    $\sharp\mathsf{S} = \mathrm{len}(\mathsf{bs.S})$; $(\beta^{\mathrm{start}}, j^{\mathrm{start}}) \leftarrow \mathsf{bs.S}[\sharp\mathsf{S}]$;
    if $j^{\mathrm{start}} = 1$ **then**
      $\beta^{\mathrm{start}}, j^{\mathrm{start}} \leftarrow \beta^{\mathrm{start}} + 1, J_{\max}(\beta^{\mathrm{start}} + 1)$;
    else
      $j^{\mathrm{start}} \leftarrow j^{\mathrm{start}} - 1$;
  $\mathsf{tmpBS} = \{\}$;
  do in parallel
    for $\beta \leftarrow \beta^{\mathrm{start}} + 1$ **to** d **do**
      for $j \leftarrow j^{\mathrm{start}}$ **to** 1 **do**
        $\sharp\mathrm{tours} \leftarrow$ the maximal tours for pnj-BKZ-β with jump j;
        $T_{\mathrm{pnjBKZ}} \leftarrow \frac{d-\beta}{j} \cdot T_{\mathrm{pump}}(\beta)$;
        for $t \leftarrow 1$ **to** $\sharp\mathrm{tours}$ **do**
          $\mathsf{bs}^*.\mathsf{S} \leftarrow \mathsf{S} \cup [(\beta, j, t)]$; $\mathsf{bs}^*.\mathsf{rr} \leftarrow$ `PnjBKZSim` $(\mathsf{bs.rr},\ \beta,\ j,\ t)$;
          $\mathsf{bs}^*.G_{\mathrm{BKZ}} \leftarrow \mathsf{bs}^*.G_{\mathrm{BKZ}} + t \cdot T_{\mathrm{pnjBKZ}}$;
          $\mathsf{bs}^*.d_{\mathrm{svp}}, \mathsf{bs}^*.\mathrm{PSC} \leftarrow$`ProSieveDimEst`$(\mathsf{bs}^*.\mathsf{rr},\ F(\star, \mathcal{D}))$;
          $\mathsf{tmpBS} \leftarrow \mathsf{tmpBS} \cup \{\mathsf{bs}^*\}$;
      $j^{\mathrm{start}} \leftarrow J_{\max}(\beta + 1)$;
  for $\mathsf{bs}^* \in \mathsf{tmpBS}$ **do**
    $\mathsf{BS} \leftarrow \mathsf{BS} \cup \{\mathsf{bs}^*\}$;
    if $\exists\ \mathsf{bs} \in \mathsf{BS}$ *s.t.* $\mathsf{bs}^*.\mathrm{PSC} \geq \mathsf{bs.PSC}$ **and** $\mathsf{bs}^*.G_{\mathrm{BKZ}} \geq \mathsf{bs}.G_{\mathrm{BKZ}}$ **then**
      $\mathsf{BS} \leftarrow \mathsf{BS} \setminus \{\mathsf{bs}^*\}$;
    else
      for $\forall \mathsf{bs} \in \mathsf{BS}$ *s.t.* $\mathsf{bs}^*.\mathrm{PSC} \leq \mathsf{bs.PSC}$ **and** $\mathsf{bs}^*.G_{\mathrm{BKZ}} \leq \mathsf{bs}.G_{\mathrm{BKZ}}$ **do**
        $\mathsf{BS} \leftarrow \mathsf{BS} \setminus \{\mathsf{bs}\ \}$;
  $k \leftarrow k + 1$;
$S_{\min}, G_{\min}, B_{\min} \leftarrow [\,], +\infty, +\infty$;
for $\mathsf{bs} \in \mathsf{BS}$ **do**
  $G \leftarrow \mathsf{bs}.G_{\mathrm{BKZ}} + \mathsf{bs.PSC}$; $\sharp\mathsf{S} = \mathrm{len}(\mathsf{bs.S})$; $B \leftarrow \mathrm{bit}_{\mathrm{pump}}(\max\{\mathsf{bs.S}[\sharp\mathsf{S}], d_{\mathrm{svp}}\})$;
  if $G_{\min} > G$ **then**
    $\mathsf{S}_{\min}, G_{\min}, B_{\min} \leftarrow \mathsf{bs.S}, G, B$;
return $\mathsf{S}_{\min}$, $G_{\min}$, $B_{\min}$;

Algorithm 8: EnumBS

sieving in the Pump. In other words, the upper bound of the sieving dimension used in default G6K supposes that the length of the target vector is fixed as its expected value and is over-optimistic. In fact, the square of the length of the LWE error vector is a randomly positive variable rather than a fixed value. Instead of giving a univariate parameter for evaluating the upper bound of the sieving dimension, it is more reasonable to make sure the success probability of recovering the target vector is high enough with a suitable sieving dimension.

New upper bound of sieving dimension in Pump. We wish to describe the relationship between the success probability of finding $(\mathbf{e}, 1)$ and the quality of lattice basis using different upper bound of sieving dimensions of Pump. Then, we can set the suitable upper bound of sieving dimension in Pump to make the probability of finding the approximate shortest vector close to 1. Through that, we can avoid executing an invalid Pump which fail to find the target vector due to an inaccurate prediction function.

The Algorithm 6 represents the condition of selecting the upper bound of sieving dimension in a progressive sieve according to a certain solving probability threshold. This idea was first proposed in [38] to optimize the dimension selection for BKZ, and [26] further accelerated the algorithm. We use the same method in [26] to reduce the cost of evaluation.

In order to calculate the value $\Pr\left[y \leftarrow \sigma^2\chi^2_{d_{\text{svp}}} \middle| y \leq \left(\text{GH}\left(\mathbf{B}_{[d-\beta:d]}\right)\right)^2\right]$, one can integrate the probability density function of the chi-squared distribution from 0 to the $\frac{1}{\sigma^2}\left(\text{GH}\left(\mathbf{B}_{[d-\beta:d]}\right)\right)^2$ value:

$$C_\beta\left(x = \frac{1}{\sigma^2}\left(\text{GH}\left(\mathbf{B}_{[d-\beta:d]}\right)\right)^2\right) = \int_0^x \frac{t^{\frac{\beta}{2}-1}e^{-\frac{t}{2}}}{2^{\frac{\beta}{2}}\Gamma\left(\frac{\beta}{2}\right)}\text{d}t,$$

then $\Pr\left[y \leftarrow \sigma^2\chi^2_\beta \middle| y \leq \text{GH}\left(\mathbf{B}_{[d-\beta:d]}\right)^2\right] = \sigma^2 \cdot C_\beta\left(x = \frac{1}{\sigma^2}\left(\text{GH}\left(\mathbf{B}_{[d-\beta:d]}\right)\right)^2\right)$.

When solving LWE instead of SVP_γ, Let $F(d_{\text{svp}}, \mathcal{D}) = \sigma^2\chi^2_{d_{\text{svp}}}$ and call the Algorithm 6. In addition, we can combine pnj-BKZ simulator (Alg.5) and Pump sieving dimension estimator (Alg.6) to give a probabilistic simulator for solving LWE based on our optimization strategy as Algorithm 7 and Algorithm 8, which has been introduced in Section 5.

6.2 Optimized Blocksize and Jump Strategy for Solving TU Darmstadt LWE Challenge

We use a machine with Intel Xeon 5128 16c 32@2.3GHz, 1.48T RAM, and NVIDIA GTX 3090 * 2 to construct a practical time-cost model by experiments, see more detail in the Appendix C. Based on this practical time-cost model we give a comparison of different LWE-solving algorithms to show our improvement.

Table 3 shows the blocksize and jump strategy selected by Algorithm 8 for solving TU Darmstadt LWE challenge instances with $P_{\text{success}} = 0.999$ using the practical cost model. From Table 3 we can also see that the time cost of generating the blocksize and jump strategy by Algorithm 8 is acceptable. To generate the optimal strategy for other lattice can implement our code shown in public. We solved TU Darmstadt LWE challenge instances of $(n, \alpha) \in \{(40, 0.035), (40, 0.040), (50, 0.025), (55, 0.020), (90, 0.005)\}$ successfully using these strategies.

Table 3. Blocksize and Jump strategy generated by EnumBS.

(n, α)	Strategy (β, jump, ♯tours)	EnumBSGen/s
(40,0.025)	[(91, 8, 1),(104, 8, 1)]	6.5
(45,0.020)	[(79, 8, 1),(91, 8, 1),(108, 8, 1)]	18
(50,0.015)	[(79, 8, 1),(91, 8, 1),(106, 8, 1)]	20
(60,0.010)	[(79, 8, 1),(89, 8, 1),(104, 8, 1),(112, 8, 1)]	57
(80,0.005)	[(91, 9, 1),(111, 11, 2),(117, 11, 2),(120, 11, 1)]	390
(40,0.035)	[(117, 11, 2),(117, 4, 1),(129, 4, 1),(142, 4, 1)]	180
(40,0.040)	[(102, 10, 1),(116, 11, 1),(120, 11, 1), (120, 4, 1), (134, 4, 1),(146, 4, 1),(154, 2, 1)]	320
(50,0.025)	[(94, 9, 1),(112, 11, 1),(117, 11, 1),(118, 11, 1), (118, 4, 1),(129, 4, 1),(140, 4, 1),(146, 4, 1),(154, 2, 1)]	690
(55,0.020)	[(101, 10, 1),(111, 11, 1),(117, 11, 1),(117, 4, 1), (119, 4, 1),(128, 4, 1),(140, 4, 1),(147, 4, 1),(155, 4, 1)]	740
(90,0.005)	[(91, 9, 1),(103, 10, 1),(111, 11, 1),(117, 11, 3), (117, 4, 1),(118, 4, 1),(120, 4, 1),(128, 4, 1), (133, 4, 1),(141, 4, 1),(143, 4, 1),(148, 4, 1)]	180

6.3 Practical Experimental Test Comparison of LWE-solving algorithms

In this part we firstly test the actual walltime cost of different LWE solving algorithms through actual experiments, so as to prove that our algorithm is indeed improved compared with the default LWE solving algorithm in G6K. From Table 4 we can see that compared with the default LWE solving strategy in G6K we decreased the walltime cost by about $8.4 \sim 10.5$ times by EnumBS strategy and $5.3 \sim 10$ times by BSSA strategy.

Meanwhile to show the accuracy of our optimized blocksize and jump selection strategy, we compare the quality of lattice basis and walltime in each middle node predicted by our optimized blocksize and jump selection strategy with that of actual experiments. Table 5 illustrates that both the quality of actual lattice basis and the actual walltime of each tour of pnj-BKZ-(β, J) are almost the same as our prediction. This also proves that Heuristic 1 is established in an experimental way.

Table 4. Experimental result of different LWE-solving algorithms[§].

(n, α)	Walltime/s			Memory/GB			Walltime Acceleration Ratio	
	G6K	BSSA	EnumBS	G6K	BSSA	EnumBS	BSSA	EnumBS
$(40, 0.025)$	11864	1314	1307	4.5	10.0	6.5	9	9.1
$(45, 0.020)$	22400	2227	2124	1.7	7.2	4.0	10	10.5
$(50, 0.015)$	20185	2518	2241	11.2	11.2	4.2	8	9
$(80, 0.005)$	147464	27595	17544	23.1	9.0	20.8	5.3	8.4

§ Here BSSA represents the two-step LWE solving algorithm whose blocksize and jump strategy is generated by alg 7 and EnumBS represents the two-step LWE solving algorithm whose blocksize and jump strategy is generated by alg 8.

Table 5. The simulated and actual quality and walltime of lattice basis in reduction process using LWE instance $(n, \alpha) = (40, 035)$.

(β, J, tours)	Simulation		Practical	
	Slope	log(Walltime/s)	Slope	log(Walltime/s)
(83,8,1)	−0.050127	8.74	−0.04768	8.83
(93,8,1)	−0.04388	9.24	−0.04279	10.08
(108,8,1)	−0.03994	10.03	−0.03957	11.06
(117,8,1)	−0.038133	10.48	−0.03801	11.96
(119,4,1)	−0.03669	11.87	−0.03709	12.99
(133,4,1)	−0.0351	14.66	−0.03553	14.92

6.4 New LWE Records

The TU Darmstadt LWE challenge website presents sample instances for testing algorithms that solve the LWE problem. The main goal of this challenge is to help in assessing the hardness of the LWE problem in practice. Furthermore, it can be used to compare different types of LWE solvers.

By our new algorithm, i.e. pro-pnj-BKZ, we have solved the LWE instances $(n, \alpha) \in \{(40, 0.035), (90, 0.005), (50, 0.025), (55, 0.020), (40, 0.040)\}$ in TU Darmstadt LWE challenge website. (We solved the LWE instance $(n, \alpha) = (80, 0.005)$ earlier.). Specifically we denoted a service with AMD EPYC$^{\text{TM}}$ 7002 Series 128@2.6GHz, NVIDIA RTX 3090 * 8, 1.5T RAM as Machine A, and denoted a service with AMD EPYC$^{\text{TM}}$ 7002 Series 64@2.6GHz, a100 *4, 512 GB RAM as Machine B. Then we listed the walltime and RAM cost in solving the above LWE challenges in Table 6. The unit of running time in Table 6 is hour and the unit of RAM in it is GB.

6.5 Security Estimation for NIST schemes

In this part, we estimated security bits of LWE in NIST schemes [37] under consideration of the influence of the optimized blocksize and jump selection and two-step mode strategy. Our new hardness estimation of LWE tries to answer Question 7 in Section 5.3 of [2] and narrows the security estimation error interval.

Table 6. Actual running time, RAM cost and simulation cost.

(n,α)	Machine	Walltime (h)	RAM (GB)
(40,0.035)	B	233	184
(50,0.025)	A	592	184
(55,0.020)	A	611	890
(90,0.005)	B	370	332
(40,0.040)	A	683	1120

For more details about our evaluation code can be seen in the open source code[1]. Under the RAM model, i.e, it assumes that access into even exponentially large memory is free, the estimated security bits of LWE in NIST schemes [37] can be reduced by 8.5 $\sim$ 11.9 bits in the case of paying memory cost increased by at most 6 bits compared to the current estimation generated by Leaky-Estimator[2] in [39]. See Table 7 for details. Here G and B in Table 7 respectively represent the total log number of logic circuits for these LWE instances in NIST schemes [37] being solved and the maximal memory needed for solving these LWE instances, that both are calculated by Gate-count algorithm [40].

Table 7. Security Estimation results of different estimator for NIST schemes♮.

	G/log2(gates)		B/log2(bit)		ΔG	ΔG+ΔB
	Leaky-Estimator	Two-Step	Leaky-Estimator	Two-Step		
Kyber512	151.5	139.6	93.8	95.7	11.9	10
Kyber768	215.1	206.4	138.5	143.2	8.7	4.1
Kyber1024	287.3	278.2	189.7	194.1	9.1	4.7
Dilithium-I	158.6	150.1	97.8	102.9	8.9	4
Dilithium-II	216.7	207.9	138.7	144.0	9.1	4.1
Dilithium-III	285.4	276.5	187.4	193.1	8.5	2.4

♮ In the column of G/log2(gates) and B/log2(bit), respectively reflect the security bit estimations and memory bit estimations of LWE instances in NIST schemes with different estimators. Here under the influence of the optimized blocksize and jump selection and two-step mode strategy, ΔG means the security bits decreasing under the RAM model and ΔG+ΔB means the security bits decreasing by considering the sum of time complexity and space complexity.

In practice, our strategy using larger memory cost will indeed lead to an extra overhead of accessing exponentially large memory, which will somewhat offset the above-claimed decreasing of security hardness. But even including the additional overhead caused by accessing higher memory, the optimized blocksize and jump selection and two-step mode strategy still have smaller sum of time complexity and space complexity compared with that of Leaky-Estimator. More preciously, considering the sum of time complexity and space complexity, the

[1] https://github.com/Summwer/lwe-estimator-with-pnjbkz.git
[2] https://github.com/lducas/leaky-LWE-Estimator

optimized blocksize and jump selection and two-step mode strategy will decrease the security bits of LWE by $2.4 \sim 10$ bits claimed in NIST schemes [37]. See Table 7 for more detail.

7 Conclusion and Future Work

7.1 Conclusion

In this paper, we propose Improved Progressive pnj-BKZ, which combines pnj-BKZ and Pump algorithm to solve SVP_γ problem based on two new simulating algorithms (pnj-BKZ simulator and Pump estimator). Experimental results show that our simulators can accurately predict the behavior of pnj-BKZ even if jump ≥ 1. We design two new blocksize and jump strategy selection algorithms: BSSA and EnumBS, and demonstrate the optimality of the EnumBS strategy. Meanwhile, applying the blocksize and jump strategy generated from EnumBS to solve the LWE Challenge results in at most 10.5 times improvement compared to default G6K mode and help us to solve the TU Darmstadt LWE challenges $(n, \alpha) \in \{(40, 0.035), (40, 0.040), (50, 0.025), (55, 0.020), (90, 0.005)\}$. In addition, using our new hardness estimator of LWE for security evaluation of these NIST lattice-based schemes shows that our optimized blocksize and jump strategy selection and two-step mode will cause the security strength to drop by $2.4 \sim 10$ bits.

7.2 Future Work

When constructing the practical time cost model of the Pump to find the optimal reduction parameter, we found that the time cost of each Pump increases linearly with the increase of the index of the Pump (index $< d - f$) in each tour of pnj-BKZ. Currently, we are only sure that this phenomenon is caused by the early termination condition (See the Appendix C for more detail), but we don't know how the early termination condition affects the time cost of the pump, and we plan to find out the reason in the future.

A good insertion in Pump could decrease the time cost and increase the quality of basis, the insert function in the Pump in the G6K is Heuristic. So we plan to study the insert function and try to design a more fittable function in our future work.

In addition, we noticed that when using G6K to solve the high-dimension lattice challenge, the program often displays a saturation warning. We don't know what causes this warning, however, in most cases, it may result in longer time overhead and we plan to solve this problem.

Besides, we give a new security estimation of the hardness of LWE in the NIST schemes by considering the influence of our optimized blocksize and jump selection and two-step mode strategy. We plan to give details of our new security estimation of NIST schemes in future work.

Although the EnumBS could obtain the optimal blocksize and jump strategy, its theoretical complexity is exponential. In future work, we hope to accelerate

the speed of generating the optimal blocksize and jump strategy for estimating the security of lattice-based schemes by improving the efficiency of EnumBS.

References

1. L. Ducas, T. L. Eike Kiltz, V. Lyubashevsky, P. Schwabe, G. Seiler, and D. Stehlé, *Dilithium(Round 3)*. NIST PQC probject, 2020.
2. R. Avanzi, J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, P. Schwabe, G. Seiler, and D. Stehlé, "Kyber(Round 3)," p. 42, 2020.
3. O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," *Journal of the ACM*, vol. 56, pp. 34:1–34:40, Sept. 2009.
4. V. Lyubashevsky, C. Peikert, and O. Regev, "On Ideal Lattices and Learning with Errors over Rings," in *Advances in Cryptology – EUROCRYPT 2010* (H. Gilbert, ed.), Lecture Notes in Computer Science, (Berlin, Heidelberg), pp. 1–23, Springer, 2010.
5. S. Bai and S. D. Galbraith, "An Improved Compression Technique for Signatures Based on Learning with Errors," in *Topics in Cryptology – CT-RSA 2014* (J. Benaloh, ed.), (Cham), pp. 28–47, Springer International Publishing, 2014.
6. R. Kannan, "Improved algorithms for integer programming and related lattice problems," in *Proceedings of the fifteenth annual ACM symposium on Theory of computing*, STOC '83, (New York, NY, USA), pp. 193–206, Association for Computing Machinery, Dec. 1983.
7. A. K. Lenstra, H. W. Lenstra, and L. Lovász, "Factoring polynomials with rational coefficients," *Mathematische Annalen*, vol. 261, pp. 515–534, Dec. 1982.
8. C. P. Schnorr and M. Euchner, "Lattice basis reduction: Improved practical algorithms and solving subset sum problems," in *Fundamentals of Computation Theory* (L. Budach, ed.), Lecture Notes in Computer Science, (Berlin, Heidelberg), pp. 68–85, Springer, 1991.
9. N. Gama, P. Q. Nguyen, and O. Regev, "Lattice Enumeration Using Extreme Pruning," in *Advances in Cryptology – EUROCRYPT 2010* (D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, J. C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, M. Sudan, D. Terzopoulos, D. Tygar, M. Y. Vardi, G. Weikum, and H. Gilbert, eds.), vol. 6110, pp. 257–278, Berlin, Heidelberg: Springer Berlin Heidelberg, 2010. Series Title: Lecture Notes in Computer Science.
10. Y. Chen and P. Q. Nguyen, "BKZ 2.0: Better Lattice Security Estimates," in *Advances in Cryptology – ASIACRYPT 2011* (D. H. Lee and X. Wang, eds.), Lecture Notes in Computer Science, (Berlin, Heidelberg), pp. 1–20, Springer, 2011.
11. M. R. Albrecht, S. Bai, J. Li, and J. Rowell, "Lattice Reduction with Approximate Enumeration Oracles," in *Advances in Cryptology – CRYPTO 2021* (T. Malkin and C. Peikert, eds.), Lecture Notes in Computer Science, (Cham), pp. 732–759, Springer International Publishing, 2021.
12. Y. Aono, Y. Wang, T. Hayashi, and T. Takagi, "Improved Progressive BKZ Algorithms and Their Precise Cost Estimation by Sharp Simulator," in *Advances in Cryptology – EUROCRYPT 2016* (M. Fischlin and J.-S. Coron, eds.), Lecture Notes in Computer Science, (Berlin, Heidelberg), pp. 789–819, Springer, 2016.
13. G. Hanrot, X. Pujol, and D. Stehlé, "Analyzing Blockwise Lattice Algorithms Using Dynamical Systems," in *Advances in Cryptology – CRYPTO 2011* (P. Rogaway, ed.), Lecture Notes in Computer Science, (Berlin, Heidelberg), pp. 447–464, Springer, 2011.

14. S. Bai, D. Stehlé, and W. Wen, "Measuring, Simulating and Exploiting the Head Concavity Phenomenon in BKZ," in *Advances in Cryptology – ASIACRYPT 2018* (T. Peyrin and S. Galbraith, eds.), Lecture Notes in Computer Science, (Cham), pp. 369–404, Springer International Publishing, 2018.
15. M. R. Albrecht, L. Ducas, G. Herold, E. Kirshanova, E. W. Postlethwaite, and M. Stevens, "The General Sieve Kernel and New Records in Lattice Reduction," in *Advances in Cryptology – EUROCRYPT 2019* (Y. Ishai and V. Rijmen, eds.), (Cham), pp. 717–746, Springer International Publishing, 2019.
16. D. Micciancio and P. Voulgaris, "Faster exponential time algorithms for the shortest vector problem," in *Proceedings of the 2010 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, Proceedings, pp. 1468–1480, Society for Industrial and Applied Mathematics, Jan. 2010.
17. R. Fitzpatrick, C. Bischof, J. Buchmann, Ö. Dagdelen, F. Göpfert, A. Mariano, and B.-Y. Yang, "Tuning gausssieve for speed," in *Progress in Cryptology - LATINCRYPT 2014* (D. F. Aranha and A. Menezes, eds.), (Cham), pp. 288–305, Springer International Publishing, 2015.
18. P. Q. Nguyen and T. Vidick, "Sieve algorithms for the shortest vector problem are practical," *Journal of Mathematical Cryptology*, vol. 2, Jan. 2008.
19. G. Herold and E. Kirshanova, "Improved Algorithms for the Approximate k-List Problem in Euclidean Norm," in *Public-Key Cryptography – PKC 2017* (S. Fehr, ed.), Lecture Notes in Computer Science, (Berlin, Heidelberg), pp. 16–40, Springer, 2017.
20. G. Herold, E. Kirshanova, and T. Laarhoven, "Speed-Ups and Time–Memory Trade-Offs for Tuple Lattice Sieving," in *Public-Key Cryptography – PKC 2018*, pp. 407–436, Springer, Cham, Mar. 2018.
21. A. Becker, N. Gama, and A. Joux, "Speeding up lattice sieving without increasing the memory, using sub-quadratic nearest neighbor search," 2015.
22. T. Laarhoven and A. Mariano, "Progressive Lattice Sieving," in *Post-Quantum Cryptography* (T. Lange and R. Steinwandt, eds.), (Cham), pp. 292–311, Springer International Publishing, 2018.
23. L. Ducas, "Shortest Vector from Lattice Sieving: A Few Dimensions for Free," in *Advances in Cryptology – EUROCRYPT 2018* (J. B. Nielsen and V. Rijmen, eds.), (Cham), pp. 125–145, Springer International Publishing, 2018.
24. L. Ducas, M. Stevens, and W. van Woerden, "Advanced Lattice Sieving on GPUs, with Tensor Cores," in *Advances in Cryptology – EUROCRYPT 2021* (A. Canteaut and F.-X. Standaert, eds.), Lecture Notes in Computer Science, (Cham), pp. 249–279, Springer International Publishing, 2021.
25. A. Becker, L. Ducas, N. Gama, and T. 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*, SODA '16, (USA), pp. 10–24, Society for Industrial and Applied Mathematics, Jan. 2016.
26. E. W. Postlethwaite and F. Virdia, "On the Success Probability of Solving Unique SVP via BKZ," in *Public-Key Cryptography – PKC 2021* (J. A. Garay, ed.), vol. 12710, pp. 68–98, Cham: Springer International Publishing, 2021. Series Title: Lecture Notes in Computer Science.
27. P.-Q. Chen, Yuanmi; Nguyen, *Réduction de réseau et sécurité concrète du chiffrement complètement homomorphe*. PhD Thesis, 2013.
28. C. Peikert, "A Decade of Lattice Cryptography," *Found. Trends Theor. Comput. Sci.*, vol. 10, pp. 283–424, Mar. 2016. Place: Hanover, MA, USA Publisher: Now Publishers Inc.

29. V. Lyubashevsky and D. Micciancio, "On Bounded Distance Decoding, Unique Shortest Vectors, and the Minimum Distance Problem," in *Advances in Cryptology - CRYPTO 2009* (S. Halevi, ed.), vol. 5677, pp. 577–594, Berlin, Heidelberg: Springer Berlin Heidelberg, 2009. Series Title: Lecture Notes in Computer Science.
30. M. R. Albrecht, R. Player, and S. Scott, "On the concrete hardness of Learning with Errors," *Journal of Mathematical Cryptology*, vol. 9, Jan. 2015.
31. K. Xagawa, "Cryptography with Lattices," p. 244, 2010.
32. T. Laarhoven, "Sieving for Shortest Vectors in Lattices Using Angular Locality-Sensitive Hashing," in *Advances in Cryptology – CRYPTO 2015* (R. Gennaro and M. Robshaw, eds.), Lecture Notes in Computer Science, (Berlin, Heidelberg), pp. 3–22, Springer, 2015.
33. A. Becker and T. Laarhoven, "Efficient (Ideal) Lattice Sieving Using Cross-Polytope LSH," in *Progress in Cryptology – AFRICACRYPT 2016* (D. Pointcheval, A. Nitaj, and T. Rachidi, eds.), (Cham), pp. 3–23, Springer International Publishing, 2016.
34. L. Babai, "On Lovász' lattice reduction and the nearest lattice point problem," *Combinatorica*, vol. 6, pp. 1–13, Mar. 1986.
35. M. R. Albrecht, L. Ducas, G. Herold, E. Kirshanova, E. W. Postlethwaite, and M. Stevens, "The General Sieve Kernel and New Records in Lattice Reduction," in *Advances in Cryptology – EUROCRYPT 2019* (Y. Ishai and V. Rijmen, eds.), vol. 11477, pp. 717–746, Cham: Springer International Publishing, 2019. Series Title: Lecture Notes in Computer Science.
36. E. Alkim, L. Ducas, T. Pöppelmann, and P. Schwabe, "Post-quantum Key Exchange—A New Hope," pp. 327–343, 2016.
37. I. T. L. C. S. R. CENTER, "Post-quantum cryptography pqc selected algorithms 2022." https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022.
38. D. Dachman-Soled, L. Ducas, H. Gong, and M. Rossi, "Lwe with side information: Attacks and concrete security estimation," in *Advances in Cryptology – CRYPTO 2020* (D. Micciancio and T. Ristenpart, eds.), (Cham), pp. 329–358, Springer International Publishing, 2020.
39. D. Dachman-Soled, L. Ducas, H. Gong, and M. Rossi, "LWE with Side Information: Attacks and Concrete Security Estimation," in *Advances in Cryptology – CRYPTO 2020: 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17–21, 2020, Proceedings, Part II*, (Berlin, Heidelberg), pp. 329–358, Springer-Verlag, Aug. 2020.
40. M. R. Albrecht, V. Gheorghiu, E. W. Postlethwaite, and J. M. Schanck, "Estimating quantum speedups for lattice sieves," in *Advances in Cryptology – ASIACRYPT 2020* (S. Moriai and H. Wang, eds.), (Cham), pp. 583–613, Springer International Publishing, 2020.
41. M. R. Albrecht, F. Göpfert, F. Virdia, and T. Wunderer, "Revisiting the Expected Cost of Solving uSVP and Applications to LWE," in *Advances in Cryptology – ASIACRYPT 2017* (T. Takagi and T. Peyrin, eds.), (Cham), pp. 297–322, Springer International Publishing, 2017.

A Pump Simulator

To simulate the default G6K mode in the high dimension, we should use a Pump Simulator to simulate the change of the lattice basis. Our Pump simulator is designed under the assumption of HKZ, which means that the Gram-Schmidt lengths after Pump obeys HKZ assumption. We have compared the Gram-Schmidt lengths after Pump with the simulated one, and find out that the square error between them is close to 0. It shows that the Pump simulator is accurate while it takes dimension-for-free value as $f = \frac{d\ln(4/3)}{\ln(d/2\pi)}$ mentioned in section 2.4. Figure 5 shows the pump simulation result and the pseudo code is as Alg.9.

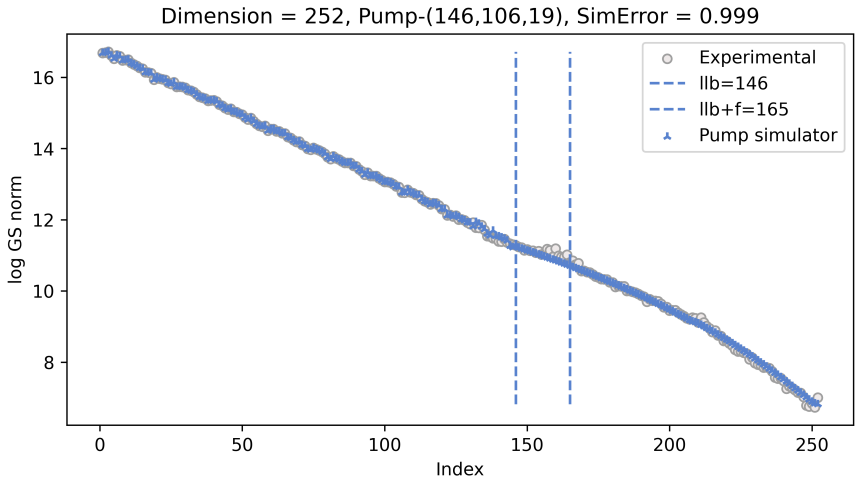


Fig. 5. Pump Simulation

B More experimental details about pnj-BKZ simulator

Here we give more verification experiments of our pnj-BKZ simulator, reducing ($n = 70, \alpha = 0.005$) LWE challenge lattice basis by pnj-BKZ with reduction parameter: ($\beta = 95, J = 9$) and ($\beta = 100, J = 12$) respectively, $\sharp$tours $\in [1, \cdots, 10]$. Here under the same reduction parameters, we do 20 times experiments. Figure 4 shows that our pnj-BKZ simulator fits well to the actual pnj-BKZ reduction result.

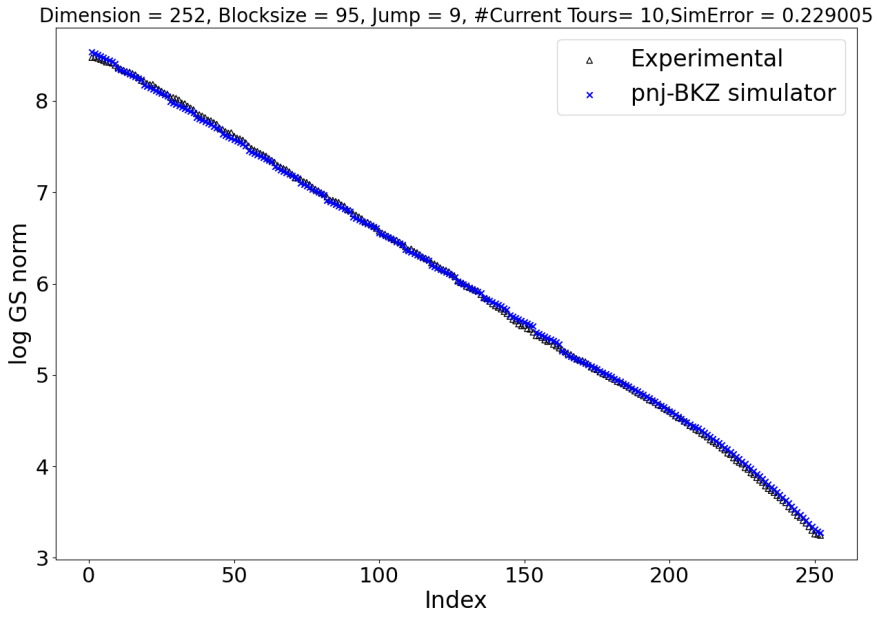
C Practical time cost model of pnj-BKZ and Pump

To find the optimal progressive blocksize and jump size selection strategy for solving TU Darmstadt LWE challenges, it is necessary to construct pnj-BKZ and pump time cost models. However, the asymptotic complexity of the sieving does not match the

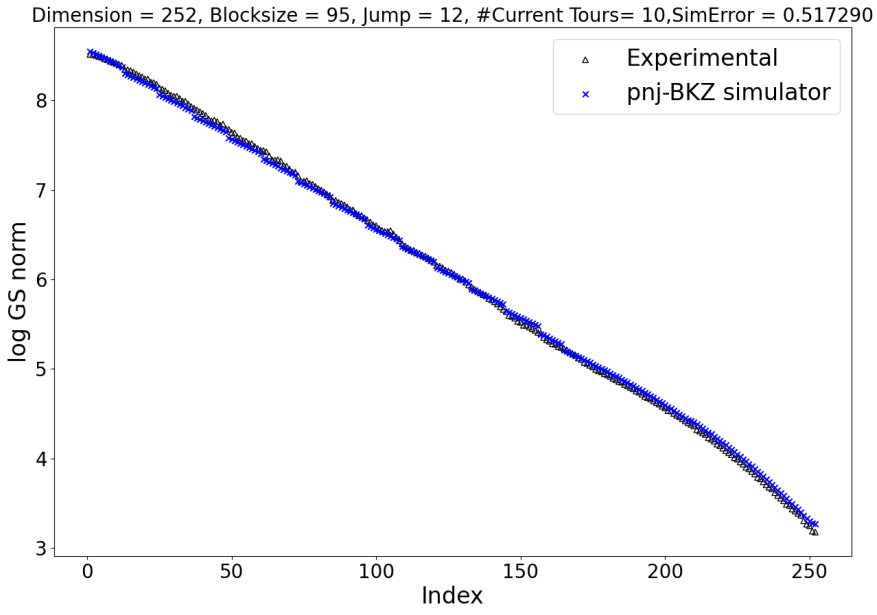
input : $(l_0, \cdots, l_{d-1})$,d, κ, β, f.
output: A prediction for the logarithms of the Gram-Schmidt norms $l'_i = \ln(\|\mathbf{b}'^*_i\|)$ after `Pump(`κ, β, f`)`.

$d_{\text{sieve}} \leftarrow \beta - f$;
for $\beta' \leftarrow d_{\text{sieve}}$ **to** d **do**
  if $\beta' < 40$ **then**
    $f' \leftarrow 0$;
  else
    $f' \leftarrow \frac{\beta' \ln 4/3}{\ln(\beta'/2\pi)}$;
  if $\beta' - f' \geq d_{\text{sieve}}$ **then**
    $\beta \leftarrow \beta'$; break;
for $i \leftarrow 0$ **to** 44 **do**
  $r_i \leftarrow$ average $\ln(\|\mathbf{b}^*_i\|)$ of a HKZ reduced random unit-volume 45-dimensional lattice;
for $i \leftarrow 45$ **to** β **do**
  $c_i \leftarrow \ln\left(\mathrm{V}_i(1)^{-1/i}\right) = \ln\left(\frac{\Gamma(i/2+1)^{1/i}}{\pi^{1/2}}\right)$;
for $k \leftarrow 0$ **to** $d - \beta - 1$ **do**
  $l'_k \leftarrow l_k$;
flag $\leftarrow$ **True**; //flag to store whether $L_{[k,d]}$ has changed
for $k \leftarrow d - \beta$ **to** $d - 46$ **do**
  $\beta' \leftarrow d - k$; $h \leftarrow d$; $\ln(\mathrm{V}) \leftarrow \sum_{i=1}^{h} l_i - \sum_{i=1}^{k-1} l'_i$;
  if *flag* = **True then**
    if $\ln(\mathrm{V})/\beta' + c_{\beta'} < l_k$ **then**
      $l'_k \leftarrow \ln(\mathrm{V})/\beta' + c_{\beta'}$; *flag* $\leftarrow$ *false*;
  else
    $l'_k \leftarrow \ln(\mathrm{V})/\beta' + c_{\beta'}$;
$\ln(\mathrm{V}) \leftarrow \sum_{i=1}^{h} l_i - \sum_{i=1}^{k-1} l'_i$;
for $k \leftarrow d - 45$ **to** $d - 1$ **do**
  $l'_k \leftarrow \frac{\ln(\mathrm{V})}{45} + r_{k+45-d}$;
for $k \leftarrow 0$ **to** $d - 1$ **do**
  $l_k \leftarrow l'_k$;
return $l_0, \cdots, l_{d-1}$;

Algorithm 9: Pump Simulator

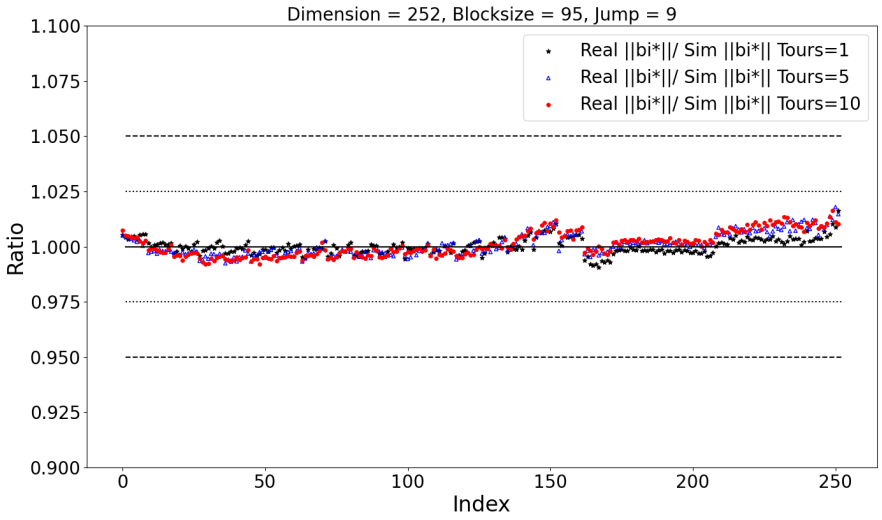


(a) $\beta = 95, jump = 9, \sharp\text{tours} = 10$

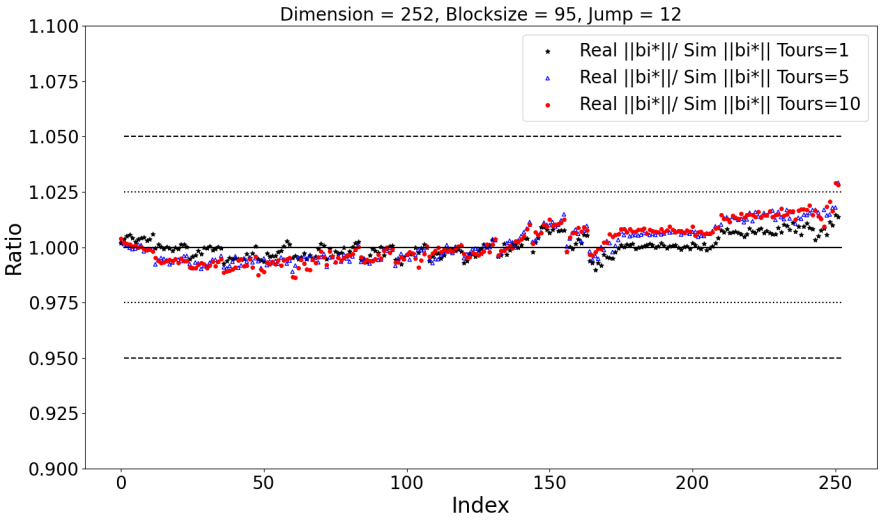


(b) $\beta = 95, jump = 12, \sharp\text{tours} = 10$

Fig. 6. Prediction effect of pnj-BKZ simulator. To verify the effectiveness of our pnj-BKZ simulator, we perform the following experiments, reducing $(n = 70, \alpha = 0.005)$ LWE challenge lattice basis by pnj-BKZ with reduction parameter: blocksize $\beta = 95$, jump size $J \in [1, \cdots, 16]$, $\sharp\text{tours} \in [1, \cdots, 10]$. Here under the same reduction parameters, we do 20 times experiments to obtain the average length of Gram-Schmidt vector.



(a) Jump=9



(b) Jump=12

Fig. 7. ratio $\frac{l_i''}{\mathrm{Sim}(l_i'')}$, $\beta = 95$. Run 10 tours ($\beta = 95$, $J = 9$ and $\beta = 95$, $J = 12$ respectively) of pnj-BKZ reduction on a lattice basis, and record the output of Gram-Schmidt vector lengths each tour. For each pnj-BKZ reduction parameter, we did experiments 20 times to obtain the average length of Gram-Schmidt vector.

actual cost well in the low-dimensional case[3] (dimension $\leq$ 128). The multithreading technology used in Pump will balance part of the time cost increases when the dimension of sieving increases. Therefore, we construct a practical time cost model by using the experimental method to test the running time of the Pump on a different lattice basis for finding the optimal reduction parameters of solving TU Darmstadt LWE challenges.

Here we need to point out that although the time-cost model based on the results of experiments can well fit the actual cost of running pnj-BKZ, using testing machines with different configurations will inevitably lead to changes in the time-cost model in low-dimensional cases. Therefore, we only use this experimentally constructed time-cost model when looking for the optimal progressive blocksize and jump size selection strategy for solving LWE challenges.

Besides, when we construct the actual time cost model by testing the time cost of pnj-BKZ on the specific machine, we find that each Pump in pnj-BKZ takes a different time cost as the figure 8(a) shown. Especially, the Pump cost increases under the incremental index smaller than $d-\beta+f$ and decreases after $d-\beta+f$ indices. It infers that for a fixed blocksize β, the average Pump cost in pnj-BKZ will increase with the growth of dimension d.

We can regard T_{pump} as a computational cost model of the $(\beta - f)$-dimensional progressive sieve, i.e.

$$\begin{aligned} T_{\text{pump}}(\beta) &= \sum_{j=\beta_0}^{\beta} T_{\text{sieve}}(j) = \sum_{j=\beta_0}^{\beta-f} 2^{c\cdot j+o(j)} = 2^{c\beta_0}\left(1+2^{c}+\cdots+2^{c(\beta-f-\beta_0)}\right) \\ &\leq 2^{c\beta_0}\cdot\frac{2^{c(\beta-f+1)+o(\beta-f+1)}}{1-2^{c}} = O\left(2^{c(\beta-f)}\right) \approx 2^{c(\beta-f)+c_1}, \end{aligned} \tag{10}$$

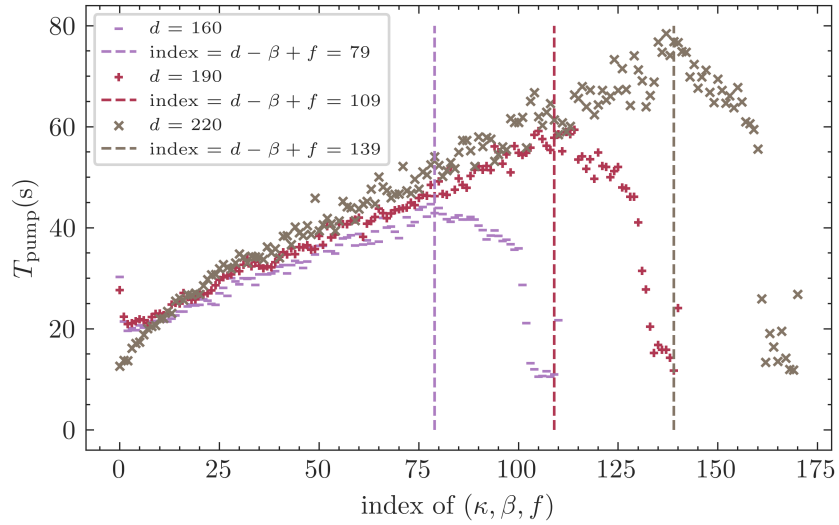
where β_0 is the dimension of initial sieving in Pump (In G6K β_0 is set to 30, and in G6K-GPU, it is set to 50), c and c_1 are the coefficients of the full sieve cost related to sieve dimension, $T_{\text{sieve}}(j)$ is the sieve cost with dimension j in dim-for-free.

However, we find that the asymptotic complexity of the sieving does not match the actual cost well in the low-dimensional case. While dimension is low, the number of threads used in Pump increases with the dimension, which balance out part of the time cost increase. So in low dimension, c might be much lower than the theoretical result.

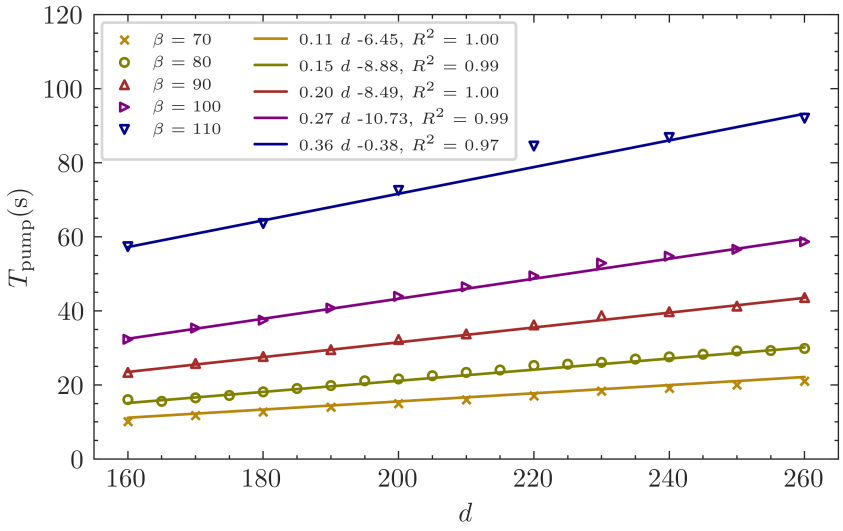
In order to accurately predict the unknown coefficients c and c_1 in the computational cost model, we use the experimental method to test the running time of Pump on different lattice basis corresponding to different TU Darmstadt LWE challenges and with different blocksizes β. The experimental results show that our computational cost model above can fit well with the actual cost of Pump.

Take $(\beta - f)$ as the independent variable, where f selected from equation (3) in section 2.4(Sieving Algorithms and Pump in G6K) of our main body. $\log_2 T$ is obtained from the experimental test as the dependent variable, and we use the least squares fitting to find c and c_1. We use R^2 to denote the coefficient of determination (R squared) value above linear regression model. The coefficient of determination (R^2 or R squared) is a statistical measure in a regression model that determines the proportion of variance in the dependent variable that can be explained by the independent variable.

[3] While dimension exceeds 128, the time cost for pump and pnj-bkz fits the theoretical value well, we can directly use the time cost model of trple_gpu sieve declared in [24].

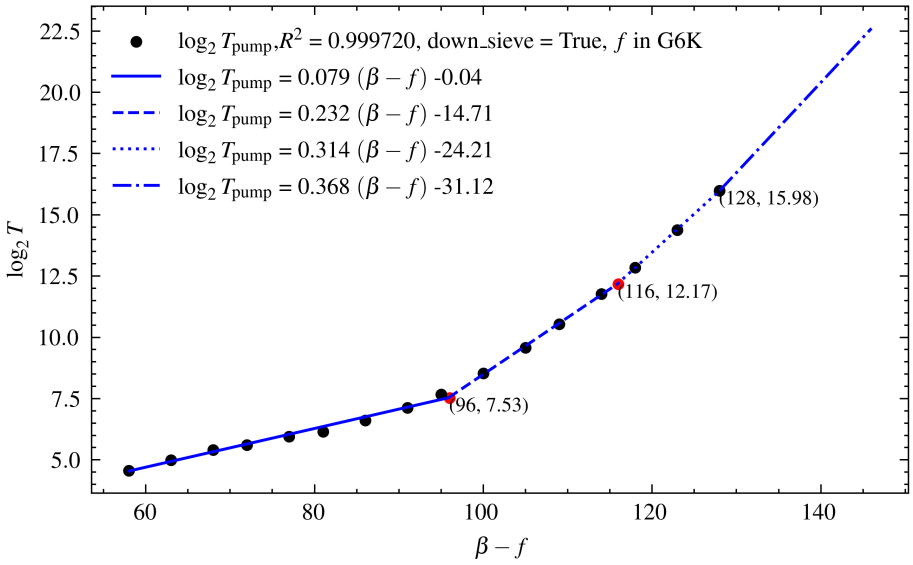


(a) Cost for each Pump under different index in pnj-BKZ-100 with one tour

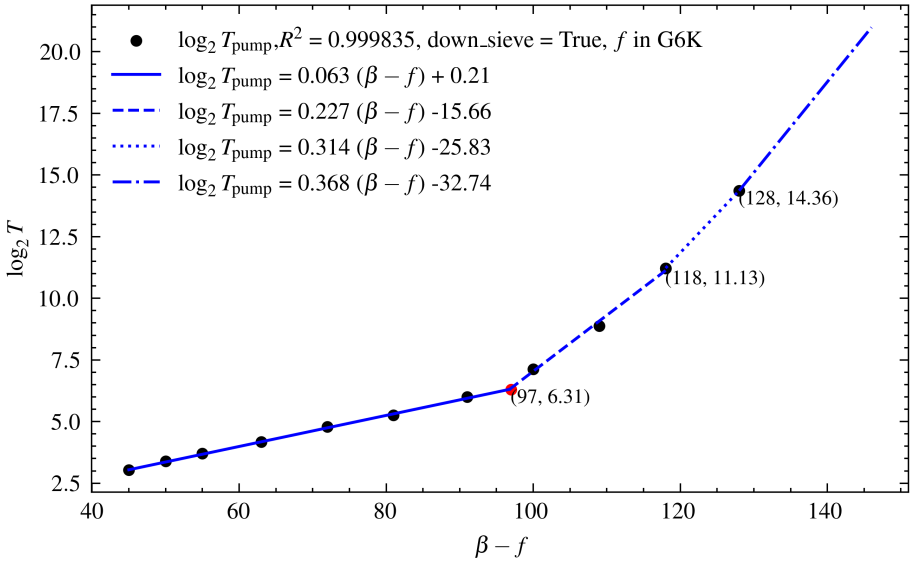


(b) Average Pump cost in pnj-BKZ-β grows linearly with d

Fig. 8. Pump cost under different indices in pnj-BKZ and average Pump costs in pnj-BKZ with change of d using Machine C.



(a) $\log_2 T_{\text{pump}}$: Test Pump independently.



(b) $\log_2 T_{\text{pump}}$: Average Pump Cost in pnj-BKZ.

Fig. 9. Pump and pnj-BKZ Cost Figure while $d = 180$, Pump Oracle = gpu_sieve, using Machine C: average T_{pump} in pnj-BKZ is lower than T_{pump} test directly, since in pnj-BKZ, the T_{pump} cost distributes uneven. The different functions from $\beta - f$ to $\log_2 T_{\text{pump}}$ result from the saturation of threads in CPU/GPU.

Generally, the range of R^2 is $[0, 1]$ and when R^2 closer is to 1, the better the model fits the data.

From Figure 9(a), we can see that R^2 is close to 1. It means that the fitting effect is good. Figure 9(a) also shows that the logarithm of the computational cost of *Pump* is linearly correlated to $\beta - f$, where f is selected from dim4free function mentioned in equation (3) in section 2.4(Sieving Algorithms and Pump in G6K) of our main body.

Pnj-BKZ consists of a series of Pumps. If we regard pnj-BKZ as a combination of Pumps with equal cost, the computational cost of pnj-BKZ can be calculated by the sum cost of $\frac{d+2f-\beta}{J}$ progressive sieves on the $(\beta - f)$-dimension projection sublattice with $J = jump$. However, as the figure 8(a) shows, each Pump in pnj-BKZ takes a different cost. Especially, the Pump cost increases under the incremental index smaller than $d - \beta + f$ and decreases after $d - \beta + f$ indices. It infers that for a fixed blocksize β, the average Pump cost in pnj-BKZ will increase with the growth of dimension d. As Figure 8(b) shown, we have tested on 5 fixed blocksizes and proven that the average Pump cost in pnj-BKZ grows linearly with d.

We suggest that the average T_{pump} in pnj-BKZ is linear in d when d is small, and independent with d when d is large. Combining the functions among T_{pump}, blocksize β and dimension d, we can get the average Pump cost equation as

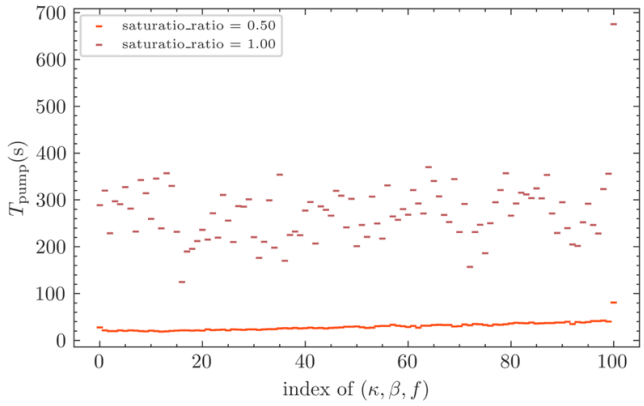
$$T_{\mathrm{pump}} = \min\left\{2^{c(\beta-f)+c_1} \cdot (c_2 \cdot d + c_3), 2^{c'(\beta-f)+c_1'}\right\} \tag{11}$$

where $2^{c\cdot(\beta-f)+c_1}$ is the Pump cost related to blocksize β and f satifsies the equation (3) in section 2.4(Sieving Algorithms and Pump in G6K) of our main body. For a lattice with dimension $d = 180$, the average Pump cost in pnj-BKZ can be simulated as Figure 9(b) and c_2 and c_3 can be computed by Figure 8(b) shown.

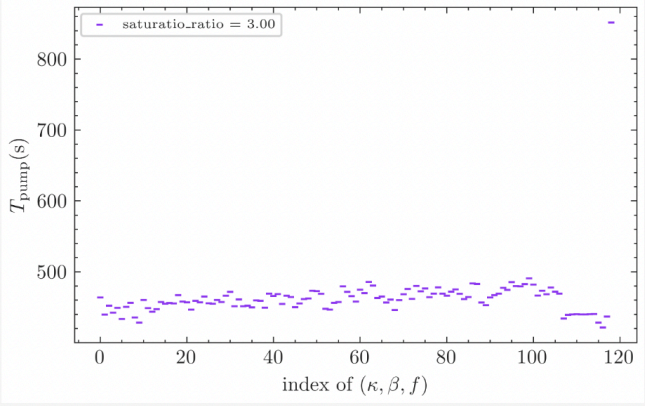
We believe that the relationship between average T_{pump} in pnj-BKZ and d is affected by the early termination condition in the implementation of sieving algorithms in Pump, where the algorithm stops when enough short vectors are generated. And we conducted the following experiments to verify our suspect.

We know that the key parameter controlling the early termination condition is the size of saturation ratio (saturation ratio is set to 0.5 by default in G6K). Therefore, we test the time cost of the pump on each projection sub-lattice under different saturation ratio. In order to remove the possible impact of the lift operation in the d4f technology, we set both d4f and f_{extra} to 0 in Figure 10(a). From Figure a we can see that the phenomenon that the pump time cost increases with the increase of the initial index of the projected sub-lattice is no longer obvious when saturation ratio is set to 1. In other words, when the early termination condition is removed, the time cost of the pump increasing phenomenon will disappear. In fact even if the d4f technique is used, as long as the early termination condition is removed, the the time cost of the pump increasing phenomenon will disappear. See Figures 10(b) and 10(c) for details.

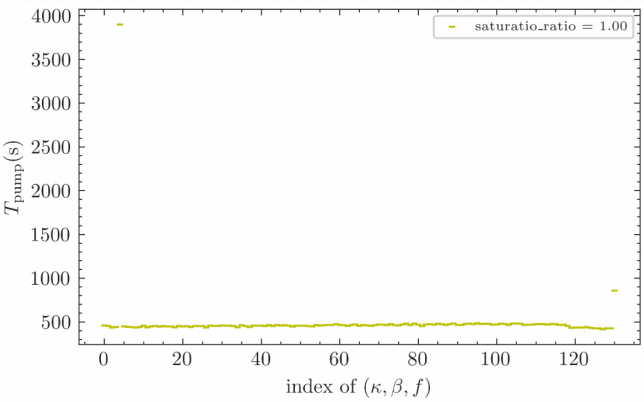
If d is large enough, we suppose that the algorithm stops only when all vectors are reduced, then the time cost of the Pump will achieve the theoretical complexity, so the average Pump cost with the growth of d will also achieve an upper bound $T_{\mathrm{pnj-BKZ}}^{(max)} = 2^{c'(\beta-f)+c_1'}$. However, we are currently not able to calculate the exact value of c' and c_1'. Besides, We believe that the reason for the gradual decrease in pump cost corresponding to the last $d - \beta + f$ is the gradual decreasing dimension sieving in these pumps.



(a) $d = 180$, $\beta = 80$, d4f $= 0$, $f_{\text{extra}} = 0$.



(b) saturation ratio=3.0, $d = 180$, $\beta = 100$, d4f $= 19$, $f_{\text{extra}} = 0$.



(c) saturation ratio=1.0 $d = 180$, $\beta = 100$, d4f $= 19$, $f_{\text{extra}} = 12$.

Fig. 10. The influence of early termination condition.

D Choosing the number of LWE Samples

BKZ-only mode is the mainstream method for estimating the security of an LWE-based crytosystem at current. It uses Kannan's Embedding technique to reduce the LWE problem to the uSVP$_\gamma$ problem and uses the GSA assumption to simulate the change after a BKZ-β reduction. Its evaluation method was firstly proposed by Erdem Alkim *et al.*. in [36] and has been proved the correctness in [41], which has both given a lower bound of LWE samples and a blocksize β. We rename it "2016 Estimation from GSA for LWE" (refer to as 2016 Estimate).

In order to solve the LWE problem, the first thing we need to do is to determine the number of LWE instances to construct the lattice basis described in the primal attack. The strategy to select the number of LWE instances in 2016 Estimate is to find the number of LWE instances m so that the following inequality holds and the value of β is minimal. Let $d = m + 1$, n be the dimension of LWE instance, then

$$\min_{\beta \in \mathbb{N}} \left\{ T_{\mathrm{bkz}}(\beta) : \sigma\sqrt{\beta} \leq \delta(\beta)^{2\beta - d - 1} \cdot q^{\frac{d-n-1}{d}} \right\}. \tag{12}$$

The strategy in 2016 Estimate is to find m so that the LWE problem can be solved with the least time cost when using a fixed blocksize of BKZ-β algorithm to solve it.

In G6K, its estimation method simulates a two-stage strategy. Their main difference from ours is that its two-stage strategy contains two tours of pnj-BKZ with a fixed blocksize β simulated from GSA assumption and a progressive sieve algorithm in dimension d_{svp}. It simulates the above scenario and try to find the minimal cost of $(\beta, d_{\mathrm{svp}})$ from

$$\min_{\beta, d_{\mathrm{svp}} \in \mathbb{N}} \left\{ 2 \cdot T_{\mathrm{bkz}}(\beta) + \mathrm{PSC}(d_{\mathrm{svp}}) : \|\pi_{d-d_{\mathrm{svp}}}(\mathbf{v})\| \leq \mathrm{GH}(\mathcal{L}_{\pi[d-d_{\mathrm{svp}}]}) \right\}, \tag{13}$$

where $c = 0.349$ in G6K-CPU and $c = 0.292$ in G6K-GPU.

Our strategy for solving the LWE problem is also simulating a two-stage strategy. In the first stage, it will call the pnj-BKZ simulator to simulate the basis after a series of pnj-BKZ. In the second stage, it tries to find the approximate shortest vector by Pump. Based on the estimation scheme in the default G6K described above, we modify the time cost of two pnj-BKZs and a progressive sieve to the time cost of serial pnj-BKZs following the blocksize strategy and a progressive sieve. Besides, we use the new Pump estimation scheme (as described in Algorithm: Pump estimation in LWE) to simulate the norm of the target vector. Let $P(d_{\mathrm{svp}}) = \Pr\left[y \leftarrow \sigma^2 \chi^2_{d_{\mathrm{svp}}} \middle| y \leq \left(\mathrm{GH}\left(\mathcal{L}_{\pi[d-d_{\mathrm{svp}}:d]} \right) \right)^2 \right]$. Thus, the inequality becomes

$$\min_{\beta, d_{\mathrm{svp}} \in \mathbb{N}} \left\{ T_{\mathrm{pnjBKZs}}(\mathbf{B}) + \mathrm{PSC}(d_{\mathrm{svp}}) : P(d_{\mathrm{svp}}) \geq P_{\mathrm{success}} \right\}, \tag{14}$$

where δ is the basis quality after pnj-BKZs. $T_{\mathrm{pnjBKZs}}(\mathbf{B})$ will respectively call BSSA and EnumBS to calculate the corresponding computational cost. To minimize the number of attempts, we narrow the range of m to $[m_0 - \tau, m_0 + \tau]$, where m_0 is the number of samples chosen in the estimation of default G6K and set a maximum search field range τ. We use a dichotomization to find an m with minimal β and d_{svp} satisfying the inequality (14). Furthermore, the concrete process is as the Algorithm 10.

Using the optimization strategy for LWE instance number selection, we can solve challenges faster than G6K default strategy. See the table 8.

input: n, q, α, m_{all}, β_{bound}, $d^{(\text{svp})}_{\text{bound}}$, τ, $\mathbf{A}^{m_{all}\times n}$, $\mathbf{b}^{m_{all}\times 1}$;
output: $S_{\min}$, $T_{\min}$, m;

$\sigma, T_{\min}, \mathsf{mRange} \leftarrow \alpha q, +\infty, \{\}$;
$m_0 \leftarrow$ LWE samples estimation in G6K as formula (13);
$m_{\max}, m_{\min} \leftarrow \max\{m \textbf{ satisfies } \text{equation (13)}\}, \min\{m \textbf{ satisfies } \text{equation (13)}\}$;
while $\tau \neq 0$ **do**
    Construct $\mathbf{B}$ by $\left(\mathbf{A}^{m_0\times n}, \mathbf{b}^{m_0\times 1}, q\right)$;
    $S_{\min}, T_{\min} \leftarrow \texttt{EnumBS(rr(}\mathbf{B}\texttt{))}, m_0+1, \sigma^2 m_0+1, J)$;
    $m_1 \leftarrow m_0$;
    for $m \in \{\max\{n, m_0-\tau\}, \min\{m_{all}, m_0+\tau\}\}$ **do**
        if $m \geq m_{\min}$ **and** $m \leq m_{\max}$ **then**
            $d \leftarrow m+1$, $M \leftarrow \sigma^2 m+1$;
            Construct $\mathbf{B}$ by $\left(\mathbf{A}^{m\times n}, \mathbf{b}^{m\times 1}, q\right)$;
            $S, T_{\text{total}} \leftarrow \texttt{EnumBS(rr(}\mathbf{B}\texttt{))}, d, M, J)$;
            if $T_{\min} < T_{\text{total}}$ **then**
                $S_{\min}, T_{\min}, m_1 \leftarrow S, T_{\text{total}}, m$;
    if $m_1 = m_0$ **then**
        $\tau \leftarrow \lfloor \frac{\tau}{2} \rfloor$;
    $m_0 \leftarrow m_1$;
return $S_{\min}, T_{\min}, m_0$;

Algorithm 10: Our LWE Samples Selection Algorithm

Table 8. LWE samples improvement simulated result with $jump = 1$.

(n,α)	G6K's m	Our m	$\text{Cost}_{\text{new}}/\text{Cost}_{\text{old}}$
(50,0.025)	218	216	99.98%
(55,0.020)	229	234	98.76%
(60,0.015)	240	246	99.30%
(90,0.005)	305	312	95.11%