Improved Progressive BKZ with Lattice Sieving

Wenwen Xia[1,⋆], Leizhang Wang[1,⋆], Geng Wang[2], Dawu Gu[1,2,3], and Baocang Wang[1]

[1] Xidian University
[2] Shanghai Jiao Tong University
[3] xiawenwen@stu.xidian.edu.cn
[4] lzwang_2@stu.xidian.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 choice in BKZ is the blocksize β, which greatly affects its efficiency. In 2016, Aono *et al.* presented *Improved Progressive BKZ* (pro-BKZ). Their work designed a blocksize strategy selection algorithm so that 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 strategy selection algorithm for BKZ with sieving, which relies on accurate time cost models and simulating algorithms. We propose the following approaches:

- New simulators and time cost models for sieving and BKZ with sieving. A simulator is used for simulating lattice reduction process without running the lattice reduction algorithm itself. We give new simulators for sieving and BKZ, to simulate the cases where blocks in BKZ with sieve oracle jump by more than one dimension. We also give more accurate time cost models for both sieving and BKZ with sieving by experiments. Specifically, we discover new relationships among time cost, blocksize and lattice dimension, which cannot be explained by the existing theoretical results, and discuss the reason.
- New two-step mode for solving SVP_γ problem 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 problem more efficiently, one should first repeatedly run BKZ to reduce the lattice basis and finally run lattice 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 timing where the algorithm ends the BKZ routine and begins sieving.
- New blocksize strategy selection algorithm for BKZ with sieving. Since the blocksize strategy selection algorithm in pro-BKZ is not optimal, we design a new blocksize strategy selection algorithm to ensure an optimal strategy output. We implement both blocksize strategy selection algorithms in pro-BKZ and ours, along with our new BKZ simulator and two-step mode to generate the blocksize strategies. Simulation results show that the strategy generated by our new algorithm are more efficient in solving SVP_γ problem. By generated strategy, we improve the efficiency nearly 24.6 times compared with heuristic blocksize strategy in G6K.

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

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 [DEKL+20], Kyber [ABD+20] which have been accepted as NIST standards. Lattice-based structures appear to be immune from attacks by both classical and quantum computers. 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 is built upon the LWE problem [Reg09] and its variants [Reg09, LPR10, BG14, ABD+20]. One of the best-known cryptanalytic technique against these problems is primal attack [Kan83], 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$_\gamma$ problem). It then calls an approximate Shortest Vector Problem (SVP$_\gamma$) solver to find the approximate shortest vector which can be used to recover the solution of LWE.

The SVP$_\gamma$ problem is a basic lattice hard problem. In recent years, substantial improvements have been made on solving the SVP$_\gamma$ problem. In 1982, the first polynomial-time lattice reduction algorithm named the LLL [LLL82] was proposed to solve the SVP$_\gamma$ problem with an exponential approximate factor γ. To solve the problem with a smaller approximate factor, Schnorr and Euchner [SE91] 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 [GNR10] to speed up enumeration, BKZ 2.0 [CN11] based on [GNR10], approximate enumeration oracle [ABLR21] on speeding up enumeration, and parameters optimization in BKZ such as Improved Progressive BKZ (pro-BKZ) [AWHT16].

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 [HPS11] and provided a BKZ simulator in BKZ 2.0 [CN11]. Using the properties that the last β vectors in BKZ-β reduction basis satisfy HKZ reduction and Gaussian heuristic, [AWHT16] 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.* [BSW18] considered the norm of shortest vector as a random variable rather than a fixed value, and brought randomness into the BKZ 2.0 simulator. The new simulator 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 [AWHT16] solves the SVP$_\gamma$ problem by calling a series of BKZs with different blocksizes first to optimize the basis quality, and an SVP oracle to find the approximate shortest vector at last (we call it a two-step mode). The main contribution 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 mid 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 become more commonplace. A lattice sieving algorithm requires more memory but less time than enumeration.

In 2019, Albrecht *et al.* [ADH+19] 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, which can selectively call the Gauss sieve [MV10, FBB+15], NV sieve [NV08], k-list sieve [HK17, HKL18] or BGJ1 sieve [BGJ15]. Pump is a generic design based on sieving algorithms using the progressive sieve introduced in [LM18] with dimension-for-free technique [Duc18], which makes the sieving process more efficient and allows a higher solving rate. Ducas *et al.* [DSvW21] improved the efficiency of G6K using GPU and implemented the fastest sieving algorithm BDGL16 [BDGL16] 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 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. Like what has been shown in pro-BKZ [AWHT16], 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 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. Besides, we also give time cost models for pnj-BKZ and Pump. In particular, we illustrate new relationships among time cost, blocksize and lattice dimension, which previous theoretical results fail to explain, and we also discuss the reason. Furthermore, we design a new simulator to simulate the quality of lattice basis after Pump.

- We propose a new model for solving SVP_γ problem by the simulating algorithms and time cost models. 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 strategies. Besides, we avoid wasting time due to failed Pumps in G6K's default strategy by using a high solving probability Pump which executes only once. Eventually, we solve the SVP_γ problem more efficiently than G6K's default strategy through the improvement during these two stages (lattice basis pre-processing stage and target vector searching stage).

- Based on the simulating algorithms, the time cost models and two-step mode, we give two new blocksize 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 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 24.6 (611, respectively) times in solving the TU Darmstadt LWE challenges compared with the default LWE solver in G6K (the LWE solver in BKZ with a fixed blocksize, respectively).

- Since the solving process has changed, we also give a new samples estimation for the LWE problem. Using this new samples estimation, we further increase the efficiency in solving LWE challenges by around 5%.

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

Organization. The paper is organized as follows. In Section 2, we present basic notations and preliminaries. In section 3, we present the sketch of pro-pnj-BKZ, design a pnj-BKZ simulator and a Pump estimator for simulation, and give cost models for Pump and pnj-BKZ. In section 4, we propose our two improved blocksize strategy selection algorithms in detail and compare them both in time cost and optimization effect. In section 5, we apply our algorithm to solving LWE problem and estimate the LWE samples afresh. In section 6, we present the simulating results and actual walltime of cost for solving the LWE challenge by pro-pnj-BKZ, and compare it with other lattice solving algorithms. In section 7, we give 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 $\|\mathbf{v}\|$.

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 $0 \le i < j \le d-1$, 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 $rr(\mathbf{B}) = (l_0, \cdots, l_{d-1})$, abbreviate to rr, $rr_{[i:j]} = (l_i, \cdots, l_{j-1})$.

Notations for algorithms description. Let BKZ-β/pnj-BKZ-β be an abbreviation of a one-tour BKZ/pnj-BKZ with blocksize β. 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

[1] (www.latticechallenge.org/lwe_challenge)

a BKZ-β reduced basis. Let $\sharp$tours be the minimum tours for BKZ-β/pnj-BKZ-β to reach a BKZ-β/pnj-BKZ-β reduced basis. Denote t as the number of tours for implementing BKZ/pnj-BKZ with a fixed blocksize β.

Let $T_{\mathrm{BKZ}}(\beta)/T_{\mathrm{pnjBKZ}}(\beta)$ be time assumption of one-tour BKZ/pnj-BKZ with blocksize β. 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 to $T_{\mathrm{BKZs}}/T_{\mathrm{pnjBKZs}}$.

Denote $T_{\mathrm{pump}}(d_{\mathrm{svp}})$ as the time cost of Pump with sieve dimension equal to d_{svp}, abbreviate it to T_{pump}. Let $\mathrm{PSC}(rr, M)$ be the Pump Solvable Cost to find the vector, whose norm is shorter than M, in lattice $\mathcal{L}(\mathbf{B})$ and $rr = rr(\mathbf{B})$, abbreviate it to PSC.

Definition 1. *(The Gaussian Distribution [PV21]) 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*

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

Definition 2. *(Chi-Squared Distribution [PV21]) 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 [Duc18]) 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 3. *(HKZ reduction and BKZ reduction [Duc18]) 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 4. *(Root Hermite Factor [Che13]) 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 5. *(Geometric Series Assumption [ADH+19]) 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 6. *(Shortest Vector Problem(SVP) [Pei16]) Given an arbitrary basis* $\mathbf{B}$ *of some lattice* $\mathcal{L} = \mathcal{L}(\mathbf{B})$*, find a shortest non-zero vector, i.e. a vector* $\mathbf{v} \in \mathcal{L}$ *for which* $\|\mathbf{v}\| = \lambda_1(\mathcal{L})$*.*

Definition 7. *(Approximate Shortest Vector Problem(SVP_γ) [Pei16]) Given an arbitrary basis* $\mathbf{B}$ *of some lattice* $\mathcal{L} = \mathcal{L}(\mathbf{B})$*, find a shortest non-zero vector, i.e. a vector* $\mathbf{v} \in \mathcal{L}$ *for which* $\|\mathbf{v}\| \leq \gamma(d) \cdot \lambda_1(\mathcal{L})$*. Besides,* $\gamma(d)$ *is an approximate factor greater than or equal to 1 and a function approximate to the lattice dimension* d*.*

Definition 8. *(unique Shortest Vector Problem($uSVP_\gamma$) [LM09]) Given an arbitrary basis* $\mathbf{B}$ *of some lattice* $\mathcal{L} = \mathcal{L}(\mathbf{B})$ *and* $\mathcal{L}$ *satisfies 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 9. *(LWE_{m,n,q,D_σ} Distribution [APS15, Pei16, Xag10]) 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 10. *(Search LWE_{m,n,q,D_σ} problem [APS15, Pei16, Xag10]) Given a pair* $(\mathbf{A}, \mathbf{b})$ *sampled from LWE distribution* Ψ *(described in Definition 9), to compute the pair* $(\mathbf{s}, \mathbf{e})$*.*

2.3 Primal Attack

Martin R. Albrecht *et al.*. [AFG14] firstly presented the primal attack for the LWE problem, which reduced Standard Form LWE problem to an SVP_γ problem by Kannan's embedding technique [Kan83]. $(\mathbf{A}, \mathbf{b})$ are LWE instances and the form of the embedding lattice basis is as

.

$$\mathbf{B}_{\mathbf{A}',\mathbf{b}} = \begin{pmatrix} \mathbf{A}' & \mathbf{b} \\ \mathbf{0}^T & 1 \end{pmatrix}, \tag{3}$$
$$\mathbf{A}' = \mathbf{P}^{-1} \begin{pmatrix} q\mathbf{I}_{m-n} & \bar{\mathbf{A}} \\ \mathbf{O} & \mathbf{I}_n \end{pmatrix},$$

in which $\mathbf{P} \in \mathbb{Z}^{m\times m}$ is a permutation matrix such that $\mathbf{P} \cdot \mathbf{A} = \begin{pmatrix} \bar{\mathbf{A}} \\ \mathbf{I}_n \end{pmatrix}$.

2016 Estimation from GSA for LWE [AGVW17, ADPS16] If β in the BKZ algorithm (or its variant) satisfies the inequality

$$\sqrt{\beta/d}\|\mathbf{v}\| \leq \delta^{2\beta-d}\mathrm{Vol}(\mathcal{L}(\mathbf{B}))^{1/d}, \tag{4}$$

then BKZ can obtain vector $\mathbf{v}$ in time $T(\beta)$ (time cost relative to β, which is an exponential function of β).

2.4 G6K and G6K-GPU-Tensor

General Sieve Kernel (G6K, [ADH+19]) was proposed by Martin R. Albrecht *et al.* in 2019 as 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.

2.5 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 $\mathbf{v} \pm \mathbf{w}$ [NV08]. 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 [Laa15, BGJ15, BL16, BDGL16] the time complexity was gradually improved to $2^{0.292d+o(d)}$ by nearest neighbour searching techniques to find close pairs more efficiently. Instead of checking all pairs, they first apply some bucketing strategy in which close vectors are more likely in the same bucket. Considering the close pairs inside each bucket are enough for reduction, the cost of a full search in the set for finding close pairs can be decreased. In order to lower the memory requirement of $2^{0.2075d+o(d)}$, one can also reduce triplets of vectors in addition to pairs. It leads to a time-memory trade-off, which lowers the memory cost while increasing the computational cost. The current best triple sieve with minimal memory $2^{0.1887d+o(d)}$ takes time $2^{0.3588d+o(d)}$ [HKL18].

Progressive Sieve Progressive sieve [LM18] 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 [Bab86] 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 The output of the sieve heuristically contains all vectors which length less than $\sqrt{\frac{4}{3}} \cdot \mathrm{GH}(\mathcal{L})$, not only the shortest vector of $\mathcal{L}$. Thus, we can denote the output of the sieve for specific sub-lattice $\mathcal{L}_{\pi[f]}$ of dimension $d_0 - f$ as

$$L \leftarrow \mathrm{Sieve}(\mathcal{L}_{\pi[f]}) = \left\{ \mathbf{x} \in \mathcal{L}_{\pi[f]} \setminus \{0\} : \|\mathbf{x}\| \leq \sqrt{\frac{4}{3}} \cdot \mathrm{GH}(\mathcal{L}_{\pi[f]}) \right\}. \tag{5}$$

Suppose the shortest vector of lattice $\mathcal{L}$ is $\mathbf{v}$, from the property of short vector set after sieving, the projection of the shortest vector $\mathbf{v}$ has been contained in L, i.e. $\pi_f(\mathbf{v}) \in L$. Since $\mathrm{GH}(\mathcal{L}) = \|\mathbf{v}\| \geq \|\pi_f(\mathbf{v})\|$, it is sufficient for containing $\mathbf{v}$ while

$$\mathrm{GH}(\mathcal{L}) \leq \sqrt{\frac{4}{3}} \cdot \mathrm{GH}(\mathcal{L}_f). \tag{6}$$

In general, if $\mathbf{v}$ is uniform, then $\|\pi_f(\mathbf{v})\| \approx \sqrt{\frac{d-f}{d}}\|\mathbf{v}\|$. Thus, we obtain a more relaxed condition for a sieve in the projected lattice is that

$$\sqrt{\frac{d_0 - f}{d_0}} \cdot \mathrm{GH}(\mathcal{L}) \leq \sqrt{\frac{4}{3}} \cdot \mathrm{GH}(\mathcal{L}_f). \tag{7}$$

Ducas has given two theoretical dimension for free estimations in [Duc18] as

$$f = \frac{d_0 \ln(4/3)}{\ln(d_0/2\pi)} \text{ and } f = \frac{d_0 \ln(4/3)}{\ln(d_0/2\pi e)},$$

while in G6K [ADH+19], it gives a more relaxed bound as

$$f = \begin{cases} 0, & d_0 < 40 \\ \lfloor \frac{d_0 - 40}{2} \rfloor, & 40 \leq d_0 \leq 75 \\ \lfloor 11.5 + 0.075 d_0 \rfloor, & d_0 > 75 \end{cases} \tag{8}$$

Pump in G6K Martin R. Albrecht *et al.* proposed Pump algorithm in [ADH+19], which is improved based on progressive Sieve [LM18] with dimension for free technique [Duc18] and the insertion tricks in [Duc18]. There are four input parameters for Pump algorithm: lattice basis $\mathbf{B}$, left insertion bound κ, insertion upper bound d_{svp} ($\kappa + d_{\text{svp}} = d$) and dimension-for-free value f (the upper bound of sieve dimension is $d_{\text{svp}} - f$) as mentioned in section 2.5. After calling a Pump, it will return a basis reduced by Pump.

2.6 Plain BKZ and Pnj-BKZ

BKZ algorithm is an adaptive algorithm for solving the SVP_γ problem, which can handle different approximate factors by adjusting the blocksize β to balance the solving time and the accuracy of the solution. Here we list two different versions of BKZ: plain BKZ and pnj-BKZ. The main difference is that they use different SVP oracles, where plain BKZ uses an enumeration algorithm as its SVP oracle while pnj-BKZ uses a progressive sieve algorithm Pump as its SVP oracle.

Plain BKZ Block Korkine-Zolotarev (BKZ) reduction algorithm is a lattice reduction algorithm whose performance between LLL reduction algorithm and HKZ reduction algorithm introduced by Schnorr and Euchner [Sch91].

The main idea of BKZ algorithm is using β' dimensional SVP oracle to find a lattice vector $\mathbf{v}_i \in \mathcal{L}_d$ s.t. $\|\pi_i(\mathbf{v}_i)\| = \mathrm{GH}\left(\mathcal{L}_{\pi[i:i+\beta']}\right)$, then insert it in i-th position of lattice basis, where $\beta' = \min(\beta, d-i)$, $i = 0, \cdots, d-1$. After each insertion, it will improve the quality of the lattice basis. Besides, it calls an LLL algorithm before lattice reduction to vanish the linear dependence and make a rough reduction after the insertion.

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 plain 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 as Algorithm 1.

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

$f \leftarrow$ equality (8) with $d_0 = \beta$;
$\beta, f \leftarrow \beta + f_{\text{extra}}, f + f_{\text{extra}}$;
$\mathbf{B} \leftarrow \texttt{LLL(B)}$;
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

2.7 Blocksize Strategy in Improved Progressive BKZ

Yoshinori Aono *et al.* proposed a blocksize strategy selection method in [AWHT16] to find an optimized blocksize strategy and speed up the efficiency of BKZ 2.0 for solving the SVP_γ problem and LWE problem. In this section, we will describe the concrete scheme of their blocksize strategy selection method.

The basic idea of Improved Progressive BKZ's optimized blocksize strategy selection algorithm is to select a sequence of blocksizes by repeatedly calling their BKZ simulator to minimize the total cost of BKZ reduction for obtaining a BKZ-β reduced basis.

Notations on blocksize strategy in pro-BKZ. A tuple in blocksize strategy $(\beta^{alg}, \beta^{start}, \beta^{goal})$ satisfying $2 \leq \beta^{start} < \beta^{goal} \leq \beta^{alg}$, $\beta^{start} \overset{\beta^{alg}}{\rightarrow} \beta^{goal}$ means that we input a β^{start} reduced basis $\mathbf{B}$, call the algorithm to update $\mathbf{B}$ using enough tours of BKZ-β^{alg} algorithm and terminate until the quality of the basis $\mathbf{B}$ is better than a BKZ-β^{goal} reduced basis. $T_{\text{BKZ}}(\beta^{start} \overset{\beta^{alg}}{\rightarrow} \beta^{goal})$ implies computing time in seconds of the above operation.

The blocksize strategy to obtain a BKZ-β reduced basis from an LLL reduced basis is denoted as follows,

$$\{(\beta_j^{alg}, \beta_j^{goal})\}_{j=1,\cdots,D} \leftarrow \left(LLL(= \beta_0^{goal}) \overset{\beta_1^{alg}}{\rightarrow} \beta_1^{goal} \overset{\beta_2^{alg}}{\rightarrow} \beta_2^{goal} \overset{\beta_3^{alg}}{\rightarrow} \cdots \overset{\beta_D^{alg}}{\rightarrow} \beta_D^{goal} (= \beta) \right). \quad (9)$$

Blocksize Strategy Optimization Method in Improved Progressive BKZ To find an optimized sequence $\{(\beta_j^{alg}, \beta_j^{goal})\}_{j=1,\cdots,D}$ that minimizes the total computing time in seconds, i.e. try

to find a sequence of $\{(\beta_j^{alg}, \beta_j^{goal})\}_{j=1,\cdots,D}$ to minimize the following formula:

$$\min \sum_{i=1}^{D} T_{\mathrm{BKZ}} \left(\beta_{i-1}^{goal} \overset{\beta_i^{alg}}{\rightarrow} \beta_i^{goal} \right). \tag{10}$$

Let $T_{\mathrm{BKZs}}(\beta_0^{goal} \rightarrow \beta')$ be the minimal time in seconds from an LLL reduced basis to a $\beta'-$reduced basis, then

$$T_{\mathrm{BKZs}}(\beta_0^{goal} \rightarrow \beta^{goal}) = \min_{\beta', \beta^{alg}} \left\{ T_{\mathrm{BKZs}}(\beta_0^{goal} \rightarrow \beta') + T_{\mathrm{BKZ}} \left(\beta' \overset{\beta^{alg}}{\rightarrow} \beta^{goal} \right) \right\}, \tag{11}$$

where $\beta' < \beta^{goal} \leq \beta^{alg}$. It is similar to the Shortest Path Problem. We can use a Shortest Path Algorithm to solve it.

3 Improved Progressive pnj-BKZ: Sketch and Basic Variants

In this section, we shall first give a sketch of *Improved Progressive pnj-BKZ* in section 3.1. In section 3.2, we give a simulator for pnj-BKZ and a dimension estimation for Pump. In section 3.3, we give a cost estimation for Pump and pnj-BKZ. At the end of this section, 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 3.4.

3.1 Our Improved Progressive pnj-BKZ

Our Improved Progressive pnj-BKZ can be described as the following: input a basis $\mathbf{B}$, an optimized blocksize strategy S for reducing $\mathbf{B}$, dimension d and target norm M, reduce $\mathbf{B}$ through a series of pnj-BKZ-β. Each β is selected from blocksize strategy S. The blocksize strategy is generated from the blocksize strategy selection algorithm (see section 4) and stored as a sequential list. After a series of lattice reductions, it will call a Pump algorithm. The parameter selection of Pump follows Algorithm 4, which will lead to finding the approximate shortest vector after Pump. The detailed process is as Algorithm 2.

input : $\mathbf{B}$, S, M, J;
output: The approximate shortest vector $\mathbf{v}$;
$\mathbf{B} = \mathtt{LLL}(\mathbf{B})$;
for β *in* S **do**
  $\mathbf{B} = \mathtt{pnj\text{-}BKZ}\ (\mathbf{B}, \beta, J)$;
$d_{\mathrm{svp}}, \_ \leftarrow \mathtt{PumpEstimation}(rr(\mathbf{B}),\ M)$;
$f \leftarrow$ Equation (8) with $d_0 = d_{\mathrm{svp}}$;
$\mathbf{B} \leftarrow \mathtt{Pump}(\mathbf{B}, d - d_{\mathrm{svp}},\ d_{\mathrm{svp}},\ f)$; // $\mathbf{B} = (\mathbf{b}_0, \cdots, \mathbf{b}_{d-1})$.
return $\mathbf{b}_0$;

Algorithm 2: Improved Progressive pnj-BKZ

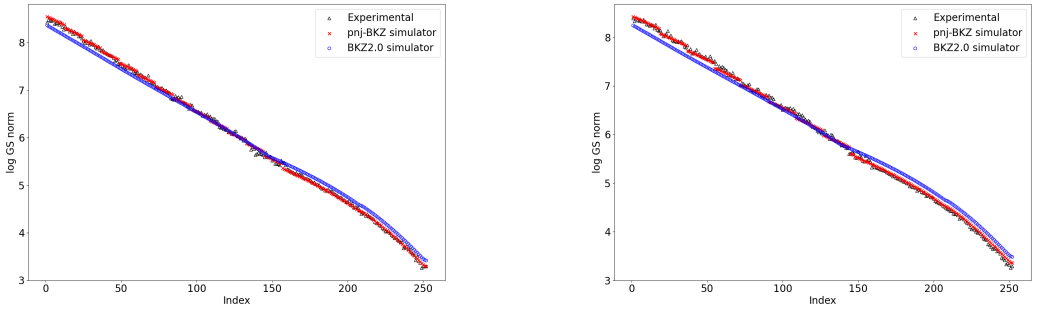
3.2 The pnj-BKZ Simulator and Pump estimation

The pnj-BKZ Simulator The pnj-BKZ is a BKZ-type lattice reduction algorithm that includes the *jump* strategy and uses Pump as the SVP oracle. The BKZ 2.0 simulator [CN11] cannot be used directly to simulate the behavior of pnj-BKZ with $jump > 1$. When $jump > 1$, let $J = jump$, after the reduction of block $\mathbf{B}_{\pi_i[i:i+\beta']}$, the $J-1$ norms of GS vectors $\mathbf{b}_{i+1}, ..., \mathbf{b}_{i+J-1}$ remain unknown. These unknown norms prevent the BKZ 2.0 simulator from predicting the norm of the first GS vector in the next block. So we give a new BKZ simulator to predict the case of $jump > 1$. To simulate pnj-BKZ while $jump > 1$, we let these vectors satisfy the HKZ reduced condition, so that we can predict the behavior of the pnj-BKZ.

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 $\mathbf{B}'$ and l_i' be the corresponding output of the basis and the logarithm of each Gram-Schmidt norm after a pnj-BKZ-β. 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 reduction with $jump = J$ by Gaussian Heuristic as

$$\begin{aligned} l_i' &= \ln\left(\mathrm{GH}\left(\mathbf{B}'_{\pi[i:i-(i \bmod J)+\beta]}\right)\right) \\ &\approx \frac{1}{2}\ln\left(\frac{\beta-(i \bmod J)}{2\pi e}\right) + \frac{1}{\beta-(i \bmod J)}\left(\sum_{j=0}^{i-(i \bmod J)+\beta-1} l_j - \sum_{j=0}^{i-1} l_j'\right), \end{aligned} \tag{12}$$

where $i \in \{0, \cdots, d-\beta-1\}$, $\mathrm{Vol}\left(\mathbf{B}'_{\pi[i:i-(i \bmod J)+\beta]}\right) = \frac{\mathrm{Vol}\left(\mathbf{B}_{[0:i-(i \bmod J)+\beta]}\right)}{\mathrm{Vol}\left(\mathbf{B}'_{[0:i]}\right)}$. As for the norm prediction of last GS norms l_i: $i \in \{d-\beta, d-1\}$, we use the same method as the BKZ 2.0 simulator.



(a) $\beta = 105$, J = 12, our square error = 0.482, simulator in BKZ 2.0 square error = 3.067

(b) $\beta = 115$, J = 18, our square error = 0.656, simulator in BKZ 2.0 square error = 5.325

Fig. 1. Gram–Schmidt log-norms for pnj-BKZ with $Jump > 1$

[2] To obtain the HKZ basis, we should turn on *pump/down_sieve* during Pump process in pnj-BKZ and delete the condition "(pump.insert_left_bound <= kappa+down_stop)" to make sure the output basis projection is HKZ reduced.

input : $(l_0, \cdots, l_{d-1})$, the desired blocksize $\beta \in \{45, \cdots, d\}$, the number of tours t and the 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 jumpsize 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**
  $flag \leftarrow$ 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
    if $k \equiv 0\,(mod\ \mathrm{J})$ **then**
      $h \leftarrow \min(k+\beta-1, d)$; //End index of local block
      $\ln(\mathrm{V}) \leftarrow \sum_{i=1}^{h} l_i - \sum_{i=1}^{k-1} l'_i$;
      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
        $l'_k \leftarrow \ln(\mathrm{V})/\beta' + c_{\beta'}$;
    else
      $h \leftarrow \min(k - (k \bmod \mathrm{J}) + \beta, d)$; $\ln(\mathrm{V}) \leftarrow \sum_{i=1}^{h} l_i - \sum_{i=1}^{k-1} l'_i$;
      if $flag =$ **True then**
        if $\ln(\mathrm{V})/(\beta' - (k \bmod \mathrm{J})) + c_{\beta'-(k \bmod \mathrm{J})} < l_k$ **then**
          $l'_k \leftarrow \ln(\mathrm{V})/(\beta' - (k \bmod \mathrm{J})) + c_{\beta'-(k \bmod \mathrm{J})}$; $flag \leftarrow false$;
      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 $flag =$ **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 3: pnj-BKZ Simulator

We give a detailed algorithm description of the pnj-BKZ simulator in the algorithm 3. 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 different jump sizes. Then BKZ 2.0 simulator and our jump BKZ simulator are used respectively to simulate the reduction effective, and finally the prediction effect of different simulator is compared. From Figure 1, it can be intuitively seen that as the jump value gradually increases, the simulation effect of BKZ 2.0 simulator becomes more and more inaccurate, however our pnj-BKZ simulator can well predict the actual behavior of pnj-BKZ with $jump > 1$. To illustrate the difference between the two simulators more clearly, we calculate the sum of squares value of residuals between the simulator simulated values and the mean of multiple experiments separately, which can be seen in Figure 1. When the value of jump increased, the square error of BKZ 2.0 simulator is increasing, while the square error of our simulator is still keeping in a low level which is much smaller than that in BKZ 2.0 simulator.

Pump Estimation Besides the simulator, we need a Pump estimator to determine the dimension of the final Pump. In G6K, a Pump estimation method has been given. Let $M_{d_{\text{svp}}} = M \cdot \sqrt{\frac{d_{\text{svp}}}{d}}$ 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 maximum 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 Pump estimation method here, which regards $M_{d_{\text{svp}}}$ as a distribution $M_{d_{\text{svp}}} = M_{[L=d-d_{\text{svp}}:R=d]} = F(L, R, \mathcal{D})$ that is related to the projected lattice, and the distribution $\mathcal{D}$ is the uniform distribution of all vectors with norm $\leq M$. In our Pump estimation method (as in Algorithm 4), it first evaluates the maximum sieve dimension d_{svp} for calling Pump to guarantee a high solving probability, then give a cost estimation by d_{svp}, where Pump cost will be mentioned in section 3.3.

input : rr, M, $P_{\text{success}} = 0.8$;
output: d_{svp}, PSC;
for $d_{\text{svp}} \leftarrow d_{\text{start}}$ **to** d **do**
  $p \leftarrow \Pr\left[M_{d_{\text{svp}}} \leftarrow F(d - d_{\text{svp}}, d, \mathcal{D}) | M_{d_{\text{svp}}} \leq \text{GH}(rr_{[d-d_{\text{svp}}:d]})\right]$;
  if $p > P_{\text{success}}$ **then**
    PSC $\leftarrow$ Equality (14) with $\beta = d_{\text{svp}}$;
    return d_{svp}, PSC;

Algorithm 4: PumpEstimation

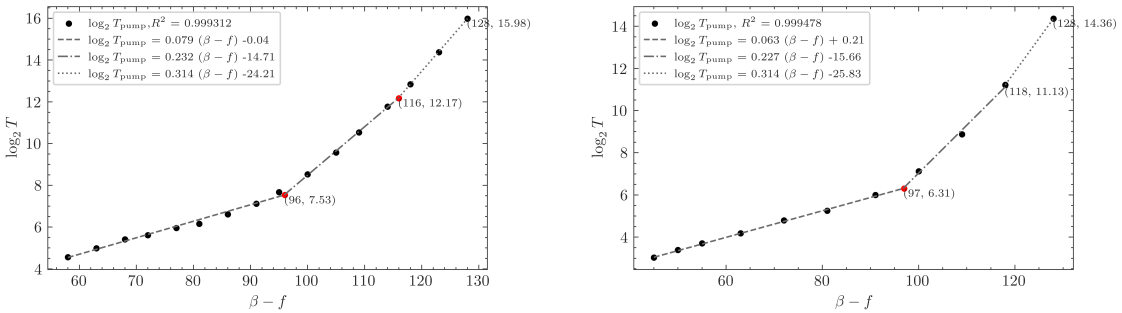
3.3 Time Cost Model for Pump, pnj-BKZ and BKZ

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

$$\begin{aligned} T_{\text{pump}} &= \sum_{j=\beta_0}^{\beta-f} 2^{c \cdot j + o(j)} = 2^{c\beta_0}\left(1 + 2^c + \cdots + 2^{c(\beta-f-\beta_0)}\right) \\ &\leq 2^{c\beta_0} \cdot \frac{2^{c(\beta-f+1)+o(\beta-f+1)}}{1-2^c} = O\left(2^{c(\beta-f)}\right) \approx 2^{c(\beta-f)+c_1}, \end{aligned} \tag{13}$$

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.

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.



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

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

Fig. 2. 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.

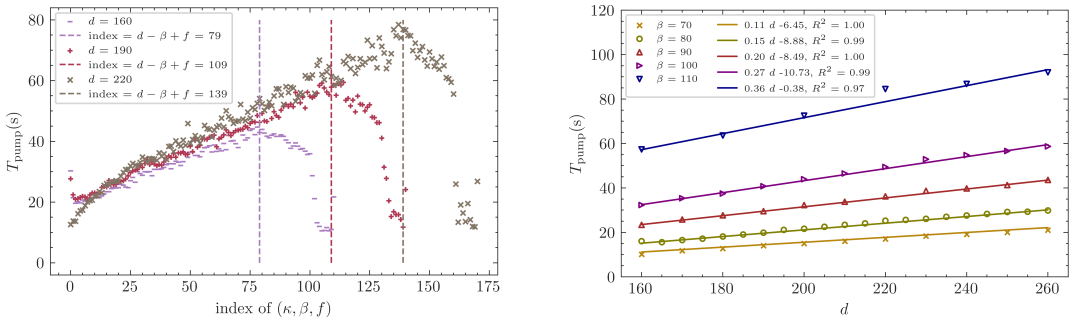
In order to accurately predict the unknown coefficients c and c_1 in the computational cost model, we use the experimental method to test the running time of Pump on different lattice basis corresponding to different TU Darmstadt LWE challenges and with different blocksizes β. The experimental results 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 (8). $\log_2 T$ 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. 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 2(a), we can see that R^2 is close to 1. It means that the fitting effect is good. Figure 2(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 section 2.5.

As described in Algorithm 1, 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 3(a) shown, each Pump in pnj-BKZ takes 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 3(b) shown, we have tested on 5 fixed blocksizes and proven that the average Pump cost in pnj-BKZ grows linearly with d.



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

We suggest that 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

$$T_{\text{pump}} = \min\left\{2^{c(\beta-f)+c_1} \cdot (c_2 \cdot d + c_3), 2^{c'(\beta-f)+c'_1}\right\} \tag{14}$$

where $2^{c\cdot(\beta-f)+c_1}$ is the Pump cost related to blocksize β for a fixed dimension d_0 and f satifsies the equation (8). For a fixed dimension $d = 180$, the average Pump cost in pnj-BKZ can be simulated as Figure 2(b) and c_2 and c_3 can be computed by Figure 3(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. If d is large enough, we suppose that the algorithm stops only when all vectors are reduced, then the time cost of Pump will achieve the theoretical complexity, so average Pump cost with growth of d will also achieve an upper bound $T^{(max)}_{\text{pnj-BKZ}} = 2^{c'(\beta-f)+c'_1}$. However, we are currently not able to calculate the exact value of c' and c'_1.

3.4 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$_\gamma$ problem and give an experiment to prove the comparison result.

BKZ-only Mode BKZ-only mode [Che13, ADPS16] implements multiple loops of BKZ-β/pnj-BKZ-β for solving SVP$_\gamma$ problem. The pseudocodes of the BKZ-only Mode are as algorithm 5, 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 *in* ♯tours **do**
  $\mathbf{B} \leftarrow \texttt{BKZ}(\mathbf{B}, \beta)$ / 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 5: BKZ-only Mode

Default Mode in G6K In the default mode of G6K [ADH+19], it solves the SVP_γ 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 mode of G6K, it will reduce the basis by a specific blocksize strategy S_0. After each lattice reduction by pnj-BKZ-β, $\beta \in S_0$, the default mode will record the time cost of the pnj-BKZ-β 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. The concrete process is as the Algorithm 6.

The benefit of the default mode in G6K is that if we do not have an accurate simulator for plain BKZ/pnj-BKZ and we are not sure of the solvability by a final *Pump* calling, then a Default 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. Besides, it might enter a Pump late and waste the processing time of extra cost for several pnj-BKZs with large blocksizes.

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

Algorithm 6: Default Mode in G6K

Two-step Mode Two-step mode was firstly proposed by [AWHT16], which calls a series of BKZ first for lattice reduction and calls an enumeration algorithm to find the target vector at last, without an optimal proof. In this paper, we use a two-step mode adapted to pnj-BKZ and Pump. It calls a series of pnj-BKZ to reduce the basis firstly and uses a Pump algorithm to search the approximate shortest vector in the final, to solve the SVP_γ problem. The concrete process is as Algorithm 2. By our pnj-BKZ simulator and Pump estimation algorithm, we can guarantee that the last Pump outputs the required target vector and the process for solving SVP_γ problem in two-step mode is optimal.

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 the two-step mode is the best mode among the three different modes.

For the comparison between two-step mode and BKZ-only mode, we can see that the reduced shortest vector in projected lattice after a Pump is shorter than that after a pnj-BKZ in the same time cost in Table 1. Thus, if the lattice basis is already well-reduced by pnj-BKZ, then calling a Pump is more likely to find the target 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 $\mathbf{b}_\kappa^*$ after a pnj-BKZ and a Pump under the same time cost.

$(n,\alpha)^\dagger$	κ	$\ln(\|\mathbf{b}_\kappa^*\|^2)$	
		pnj-BKZ	Pump in $\mathcal{L}_{[\kappa:d]}$
(50,0.020)	70	13.53	12.54
(55,0.020)	82	13.39	12.22
(75,0.010)	123	13.04	11.30
(90,0.005)	156	12.14	9.94

† 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‡	$\log_2(\mathrm{PSC})(\log_2 \mathrm{h})$	
		pnj-BKZ	Pump in $\mathcal{L}_{[\kappa:d]}$
(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 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 Pump for finding the target norm on the specific lattice basis, i.e. PSC, be a standard of reduced quality. A lattice basis with low PSC can be regarded as high quality. Separately call a pnj-BKZ/Pump on the same lattice basis with same time overhead. Table 2 shows that basis quality after a pnj-BKZ reduction is higher than that after a Pump reduction in the same time overhead. (The latter basis quality is estimated using the Pump estimator in Appendix A.) 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, G6K-default mode is less efficient than two-step mode.

4 Improved Progressive pnj-BKZ: Blocksize Strategy Optimization

In this section, we describe the two blocksize strategy selection algorithms in detail. The first is *blocksize strategy selection algorithm based on pro-BKZ* (BSSA), based on the blocksize strategy in Improved Progressive BKZ mentioned in section 2.7. The second is a new algorithm called *blocksize strategy enumeration* (EnumBS), through which we can get an optimal blocksize strategy. We give both formal proof and experiment results to show that our new strategy selection algorithm is better than the algorithm based on pro-BKZ.

input : rr_0, BS,β_0^{start}, β_0^{goal}, d, M, J;
output: BS for $\beta_0^{start} \to \beta_0^{goal}$;
for $\beta \leftarrow \beta_0^{start}$ **to** β_0^{goal} **do**
  $T_{\min} \leftarrow +\infty$;
  for $\beta^{sstart} \leftarrow \beta_0^{start}$ **to** β **do**
    if $\beta^{sstart} = \beta_0^{start}$ **then**
      $T_{\text{pnjBKZs}}(\beta_0^{start} \to \beta^{sstart})$, $rr \leftarrow 0$, rr_0;
    else if $\beta^{sstart} > \beta_0^{start}$ **then**
      if β^{sstart} **not in** BS **then**
        BS $\leftarrow$ GenBSproBKZ (rr_0, BS, β_0^{start}, β^{sstart}, d, M, J);
      $rr \leftarrow$ BS $[\beta^{sstart}]$.value (rr);
      $T_{\text{pnjBKZs}}(\beta_0^{start} \to \beta^{sstart}) \leftarrow$ BS $[\beta^{sstart}]$.value $(T_{\text{pnjBKZs}}(\beta_0^{start} \xrightarrow{t\cdot\beta^{alg}} \beta^{sstart}))$;
    rr', β^{alg}, t, $T_{\text{pnjBKZs}}(\beta^{sstart} \xrightarrow{t\cdot\beta^{alg}} \beta) \leftarrow$ SimTimeOpt(rr, β^{sstart}, β, d, M, J);
    $T \leftarrow T_{\text{pnjBKZs}}(\beta_0^{start} \to \beta^{sstart}) + T_{\text{pnjBKZs}}(\beta^{sstart} \xrightarrow{t\cdot\beta^{alg}} \beta)$;
    if $T_{\min} > T$ **then**
      $T_{\min}$, BS$[\beta] \leftarrow T$, $\left(rr', \beta^{sstart} \xrightarrow{t\cdot\beta^{alg}} \beta, T_{\text{pnjBKZs}}(\beta^{sstart} \xrightarrow{t\cdot\beta^{alg}} \beta)\right)$;
return BS **of** $\beta_0^{start} \to \beta_0^{goal}$;

Algorithm 7: Blocksize strategy generation in pro-BKZ (GenBSproBKZ)

4.1 Blocksize Strategy Selection Algorithm based on Pro-BKZ

The *blocksize strategy selection algorithm based on pro-BKZ* (BSSA) can be described as an application of the shortest path algorithm on blocksize strategy selection. It firstly sets several nodes of β_i as a measure of basis quality and expects to find the optimal blocksize β^{alg} with tours t between each nodes, which can be used to minimize the time cost from β^{start} to β^{goal} and the pseudocode is as the Algorithm 7. The goal of function SimTimeOpt is to find the best β^{alg} such that after calling t tours of pnj-BKZ-β^{alg} with jump J, the quality of rr which has reached quality of β^{start} will elevate to the β^{goal} quality. To store the variables in blocksize generation process, we define a blocksize strategy dictionary BS, in which the key is each β^{goal} node, and the value is a tuple of $\left(rr(\beta^{goal}), \beta^{start} \xrightarrow{t\cdot\beta^{alg}} \beta^{goal}, T_{\text{pnjBKZs}}(\beta^{start} \xrightarrow{t\cdot\beta^{alg}} \beta^{goal})\right)$, where $rr(\beta^{goal})$ is the GS-lengths' norm in pnj-BKZ-β^{goal} reduction, $t = \sharp\text{tours}(\beta^{start} \xrightarrow{\beta^{alg}} \beta^{goal})$ means the minimum number of tours from $rr(\beta^{start})$ to $rr(\beta^{goal})$ and $T_{\text{pnjBKZs}}(\beta^{start} \xrightarrow{t\cdot\beta^{alg}} \beta^{goal})$ denotes simulated time cost for t tours of pnj-BKZ-β^{alg}.

To find the shortest vector's minimum cost, we should also consider the cost of the last Pump, i.e. PSC , after several pnj-BKZ reductions. Thus, we choose multiple number of β^{goal}s to find the minimum total cost of pnj-BKZ from β^{start} to β^{goal} and Pump from β^{goal} to finding the short vector. The pseudocodes about BSSA is Algorithm 8.

input : $rr_0 = (l_0, \cdots, l_{d-1})$, β^{start}, β^{goal}, d, M, J;
output: BS of $\beta^{start} \to \beta_{\min}$, $T_{\min}$;

```
BS = {};
BS = GenBSproBKZ(rr_0, BS, β^start, β^goal, d, M, J);
T_min, β_min ← +∞, β^start;
for β ← β^start + 1 to β^goal do
    T_pnjBKZs ← BS[β].value(T_pnjBKZs(β^start --t·β^alg--> β));
    rr ← BS [β].value(rr);
    _, PSC ← PumpEstimation(rr, M);
    T ← T_pnjBKZs + PSC;
    if T < T_min then
        T_min, β_min ← T, β;
return BS of β^start → β_min, T_min;
```

Algorithm 8: *blocksize strategy selection algorithm based on pro-BKZ(BSSA)*

4.2 Blocksize Strategy Enumeration Algorithm

The *blocksize strategy enumeration algorithm* (EnumBS) is a pruning enumeration algorithm that enumerates all the possible blocksize strategies below the maximum time cost. Let each blocksize strategy be stored as a tuple $([\beta_1, ..., \beta_n], T_{\text{pnjBKZs}} = \sum_{i=1}^{n} T_{\text{pnj-BKZ}}(\beta_i), \text{PSC}(rr, M), rr)$, where $50 \leq \beta_1 \leq ... \leq \beta_n \leq d$, T_{pnjBKZs} is the total cost of pnj-BKZs under blocksize list $[\beta_1, ..., \beta_n]$, and $\text{PSC}(rr, M)$/PSC is the Pump Solvable Cost, which can be viewed as the quality of basis after n tours of pnj-BKZ reduction.

We consider all possible strategies as a dynamic enumeration tree, traverse through the tree and store each node in a list of blocksize strategies. A strategy is discarded from the list, only if there is another strategy which the total time cost of pnj-BKZ is shorter and the quality of basis after reduction is also better. Finally, we compare all strategies in the list to find the optimal strategy. The pseudocode of EnumBS is in Algorithm 9.

Proof for the Optimality of EnumBS

Theorem 1. *Let $rr(S)$ be the Gram-Schmidt Lengths of basis after calling a series of pnj-BKZ simulator following the strategy S. Suppose all stratgies form set $\mathcal{S}$, it exists an optimal strategy $S_{op} \in \mathcal{S}$ s.t. $T(S_{op}) = T_{\text{pnjBKZs}}(S_{op}) + \text{PSC}(rr(S_{op}), M)$ is minimum. The output strategy S_{EnumBS} of EnumBS is the optimal strategy, i.e. $S_{\text{EnumBS}} = S_{op}$.*

Proof. (Reduction to Absurdity) Suppose there's a strategy $S' \in \mathcal{S}$ such that $T(S') < T(S_{\text{EnumBS}})$. Then there are four different situations:

(1) $\text{PSC}(rr(S'), M) < \text{PSC}(rr(S_{\text{EnumBS}}), M)$, i.e. the basis quality of series of pnj-BKZ with strategy of S' is better than the basis quality of S_{EnumBS}.

(1.1) If $T_{\text{pnjBKZs}}(S') \leq T_{\text{pnjBKZs}}(S_{\text{EnumBS}})$, then the strategy S_{EnumBS} will be replaced by S', contradictory.

(1.2) If $T_{\text{pnjBKZs}}(S') > T_{\text{pnjBKZs}}(S_{\text{EnumBS}})$, S' will appear in the BS at last. Since $T(S') < T(S_{\text{EnumBS}})$, EnumBS won't output the strategy S_{EnumBS}, contradictory.

(2) $\text{PSC}(rr(S'), M) \geq \text{PSC}(rr(S_{\text{EnumBS}}), M)$, i.e. the basis quality of series of pnj-BKZ with strategy of S' is worse than the basis quality of S_{EnumBS}.

(2.1) If $T_{\text{pnjBKZs}}(S') \geq T_{\text{pnjBKZs}}(S_{\text{EnumBS}})$, then $T(S') \geq T(S_{\text{EnumBS}})$, contradictory to the assumption.

(2.2) If $T_{\text{pnjBKZs}}(S') < T_{\text{pnjBKZs}}(S_{\text{EnumBS}})$. Then, S' will appear in the BS at last. Since $T(S') < T(S_{\text{EnumBS}})$, EnumBS won't output the strategy S_{EnumBS}, contradictory. □

Table 3 shows the blocksize strategy selected by Algorithm 9 in TU Darmstadt LWE challenge instances with $P_{\text{success}} = 0.8$. In this paper the threshold for judging whether a simulated solving strategy can successfully find target vector is set to 0.8.

4.3 Practical Time cost Comparison between BSSA and EnumBS

Although the time cost of EnumBS is exponential, it is still acceptable in selecting a strategy since it takes only about an hour while running the algorithm on dimension $d = 313$ and even faster than BSSA when the lattice dimension is low. Table 4 shows the actual time cost of BSSA and EnumBS while running them to obtain the optimized blocksize strategy for LWE instances in the LWE challenge. Besides, the simulated time cost of strategy generated by EnumBS is lower than the time cost of BSSA as the Table 5, thus the generating cost for EnumBS is valuable.

Table 4. Running time of BSSA and EnumBS for strategy generation.

(n,α)	Cost(s)	
	BSSA	EnumBS
(45,0.030)	1100.57	181.8
(50,0.025)	1837.74	373.24
(55,0.020)	1404.99	462.67
(60,0.015)	1224.61	686.07
(90,0.005)	2344.02	3864.899

Table 5. Cost estimation of strategy from BSSA and EnumBS.

(n,α)	Cost(h)	
	BSSA	EnumBS
(45,0.030)	476.83	315.6
(50,0.025)	2132.63	867.57
(55,0.020)	1029.44	834.87
(60,0.015)	136.28	130.0
(90,0.005)	278.6	277.85

5 Applying Pro-pnj-BKZ to LWE

In this section, we apply our algorithm to LWE and present a new rule for choosing LWE samples based on our optimized blocksize selection strategy in section 4, which is different from that in 2016 Estimate [ADPS16]. In section 5.1, we will explain the method in choosing a dimensional upper bound for the last Pump in two-step mode, which is different from the chosen rule in default G6K since the chosen rule in default G6K is over-optimistic. An over-optimistic estimation for dimension in last Pump will lead to a bad reduction, through which the output basis will exclude the predicted short vector. In section 5.2, we will give a scheme for selecting an optimized number of LWE samples.

5.1 Pump Estimation in LWE

In this section, we reconsider the relationship between the dimension of final 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 Pump based on the distribution of the length of LWE error term, 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

$$\sigma\sqrt{d_{\text{svp}}} \approx \left\|\pi_{d-d_{\text{svp}}}(\mathbf{t})\right\| \leq \text{GH}(\mathbf{B}_{[d-d_{\text{svp}}]}), \tag{15}$$

input : $rr_0 = (l_0, \cdots, l_{d-1})$, d, M, J;
output: $S_{\min}$, $T_{\min}$;
$\_$, $\mathrm{PSC}^{(0)} \leftarrow$ `PumpEstimation` (rr_0, M);
$\mathsf{BS} \leftarrow \{(\emptyset, 0, \mathrm{PSC}^{(0)}, rr_0)\}$;
$i = 0$; // $\sharp\mathsf{BS}$ is the number of elements in BS.
while $i < \sharp\mathsf{BS}$ **do**
  $\mathsf{BS} \leftarrow$ Sort BS in order PSC from large to small;
  $S_{start} \leftarrow \mathsf{BS}\,[i][0]$; // Blocksize strategy list.
  if $S_{start} = \emptyset$ **then**
    $\beta^{start} \leftarrow 50$;
  else
    $\beta^{start} \leftarrow \max(S_{start})$;
  for $\beta \leftarrow \beta^{start} + 1$ **to** d **do**
    $\sharp$tours $\leftarrow$ the maximum tours for pnj-BKZ-β;
    for $t \leftarrow 1$ **to** $\sharp$tours **do**
      $S^* \leftarrow S \cup \underbrace{[\beta, \cdots, \beta]}_{t}$;
      $T^* \leftarrow T_{\mathrm{pnjBKZs}}(S^*)$;
      $rr^* \leftarrow$ GS-lengths after calling `pnj-BKZ simulator` with jump J by strategy S^*;
      $\_$, $\mathrm{PSC}^* \leftarrow$ `PumpEstimation`(rr^*, M);
      $\mathsf{BS} \leftarrow \mathsf{BS} \cup \{(S^*, T^*, \mathrm{PSC}^*, rr^*)\}$;
      if $\exists\ (S', T', \mathrm{PSC}', rr') \in \mathsf{BS}$ *s.t.* $\mathrm{PSC}^* \geq \mathrm{PSC}'$ **and** $T^* \geq T'$ **then**
        $\mathsf{BS} \leftarrow \mathsf{BS} \setminus \{(S^*, T^*, \mathrm{PSC}^*, rr^*)\}$;
      else
        for $\forall\ (S, T, \mathrm{PSC}, rr) \in \mathsf{BS}$ *s.t.* $\mathrm{PSC}^* \leq \mathrm{PSC}$ **and** $T^* < T$ **do**
          $\mathsf{BS} \leftarrow \mathsf{BS} \setminus \{(S\ , T\ , \mathrm{PSC}, rr)\}$;
  $i \leftarrow i + 1$;
$S_{\min}, T_{\min} \leftarrow [\,], +\infty$;
for $\mathsf{bs} \in \mathsf{BS}$ **do**
  $(S, T_{\mathrm{pnjBKZs}}, \mathrm{PSC}\ , rr) \leftarrow \mathsf{BS}\,[\mathsf{bs}\,]$;
  $\_$, $\mathrm{PSC} \leftarrow$ `PumpEstimation`(rr, M);
  $T \leftarrow T_{\mathrm{pnjBKZs}} + \mathrm{PSC}$;
  if $T_{\min} > T$ **then**
    $S_{\min}, T_{\min} \leftarrow S, T$;
return $S_{\min}$, $T_{\min}$;

Algorithm 9: Blocksize Strategy Enumeration (EnumBS)

Table 3. Blocksize strategy generated by EnumBS in TU Darmstadt LWE challenges.

n	α	Strategies	$jump$	$d_{\mathrm{pump}}(\kappa, \beta, f)$
45	0.030	[87, 87, 117, 117, 137, 143]	2	(36,152,22)
50	0.025	[87, 87, 107, 117, 119, 129, 140]	2	(53,167,24)
55	0.020	[87, 87, 107, 107, 117, 123, 140, 145]	2	(66,165,23)
60	0.015	[87, 107, 107, 107, 117, 121, 130, 135]	2	(84,158,23)
85	0.005	[87, 87, 88, 107, 107, 107, 117, 117, 120, 124, 127]	2	(141,146,22)
90	0.005	[87, 87, 107, 107, 117, 117, 117, 119, 127, 129, 133, 140]	2	(147,160,23)

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

In fact, the length of LWE error vector is a randomly positive variable rather than a fixed value. Instead of giving a univariate parameter for evaluating the sieving dimension, it is more reasonable to calculate the success probability in recovering the target vector with different dimensions.

New upper bound of dimension in Pump. We wish to describe the relationship between the length of the target vector $(\mathbf{e}, 1)$ and the success probability of finding $(\mathbf{e}, 1)$ from different Pump dimensions. Then, we can set the upper dimension bound in Pump to make the probability of finding the approximate shortest vector large. Through that, we will save the time of executing an invalid Pump with a low dimension due to an inaccurate prediction function and find the approximate shortest vector in one Pump with a probability of success close to 1.

The algorithm 10 represents the condition of selecting the upper dimension bound in a progressive sieve according to a certain solving probability threshold. This idea was first proposed in [DSDGR20] to optimize the dimension selection for BKZ, and [PV21] further accelerated the algorithm. We use the same method in [PV21] to reduce the evaluation cost.

input: σ, rr, $d_{\text{start}} = 50$, $P_{\text{success}} = 0.8$;
output: d_{svp};
for $d_{\text{svp}} \leftarrow d_{\text{start}}$ **to** d **do**
  $p \leftarrow \Pr\left[y \leftarrow \sigma^2\chi^2_{d_{\text{svp}}} \middle| y \leq \left(\text{GH}\left(rr_{[d-d_{\text{svp}}:d]}\right)\right)^2\right]$;
  if $p > P_{\text{success}}$ **then**
    PSC $\leftarrow$ Equality (14) with $\beta = d_{\text{svp}}$;
    return d_{svp}, PSC;

Algorithm 10: PumpEstimationInLWE

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:

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

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

When solving LWE instead of SVP_γ, we use Algorithm 10 instead of Algorithm 4. In addition, we can combine pnj-BKZ simulator and *Pump* estimator to give a probabilistic simulator for solving uSVP_γ based on our optimization strategy as Algorithm 8 and 9, which has been introduced in Section 4.

5.2 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 [ADPS16] and has been proved the correctness in [AGVW17], 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

$$\min_{\beta\in\mathbb{N}} \left\{ T_{\text{bkz}}(\beta) : \sigma\sqrt{\beta} \leq \delta\left(\beta\right)^{2\beta-d-1} \cdot q^{\frac{d-n-1}{d}} \right\}. \tag{16}$$

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}. Thus, it obtains the inequality (17) to simulate the above scenario and hope to find the minimal cost of (β, d_{svp}):

$$\min_{\beta,d_{\text{svp}}\in\mathbb{N}} \left\{ 2\cdot T_{\text{bkz}}(\beta) + \text{PSC}(d_{\text{svp}}) : \|\pi_{d-d_{\text{svp}}}(\mathbf{v})\| \leq \text{GH}(\mathcal{L}_{\pi[d-d_{\text{svp}}]}) \right\}, \tag{17}$$

$$\text{i.e.} \quad \min_{\beta,d_{\text{svp}}\in\mathbb{N}} \left\{ 2d\cdot 2^{c\cdot(\beta-f(\beta))} + 2^{c\cdot(d_{\text{svp}}-f(d_{\text{svp}}))} : \sigma\cdot\sqrt{d_{\text{svp}}} \leq \delta_\beta^{d_{\text{svp}}-d} \cdot \left(\frac{\Gamma(\frac{d_{\text{svp}}}{2}+1)}{\sqrt{\pi}}\right)^{\frac{1}{d_{\text{svp}}}} \cdot q^{\frac{d-n-1}{d}} \right\}, \tag{18}$$

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 10) to simulate the norm of the target vector. Thus the inequality becomes

$$\min_{\beta,d_{\text{svp}}\in\mathbb{N}} \left\{ T_{\text{pnjBKZs}}\left(\mathbf{B}\right) + \text{PSC}\left(d_{\text{svp}}\right) : \Pr\left[y \leftarrow \sigma^2\chi^2_{d_{\text{svp}}} \middle| y \leq \left(\text{GH}\left(\mathcal{L}_{\pi[d-d_{\text{svp}}:d]}\right)\right)^2 \right] \geq P_{\text{success}} \right\}, \tag{19}$$

$$\begin{aligned} \text{i.e.} \quad \min_{\beta,\text{d}_{\text{svp}}\in\mathbb{N}} \{ & T_{\text{pnjBKZs}}\left(\mathbf{B}\right) + \text{PSC}\left(d_{\text{svp}}\right) : \\ & \Pr\left[y \leftarrow \sigma^2\chi^2_{d_{\text{svp}}} \middle| y \leq \delta^{d_{\text{svp}}-d} \cdot \left(\frac{\Gamma(\frac{d_{\text{svp}}}{2}+1)}{\sqrt{\pi}}\right)^{\frac{1}{d_{\text{svp}}}} \cdot q^{\frac{d-n-1}{d}} \right] \geq P_{\text{success}} \}, \end{aligned} \tag{20}$$

where δ is the basis quality after pnj-BKZs. $T_{\text{pnjBKZs}}\left(\mathbf{B}\right)$ 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 (20). Furthermore, the concrete process is as the Algorithm 11.

input: n, q, α, m_{all}, β_{bound}, $d_{\text{bound}}^{(\text{svp})}$, 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 (18);
$m_{\max}, m_{\min} \leftarrow \max\{m$ **satisfies** equation (18)$\}, \min\{m$ **satisfies** equation (18)$\}$;
while $\tau \neq 0$ **do**
  Construct $\mathbf{B}$ by $\left(\mathbf{A}^{m_0\times n}, \mathbf{b}^{m_0\times 1}, q\right)$ as the equality (3);
  $S_{\min}$, $T_{\min} \leftarrow$ `EnumBS`$(rr(\mathbf{B}), m_0 + 1, \sigma^2 m_0 + 1, J)$;
  $m_1 \leftarrow m_0$;
  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 $\left(\mathbf{A}^{m\times n}, \mathbf{b}^{m\times 1}, q\right)$ as the equality (3);
      S, $T_{\text{total}} \leftarrow$ `EnumBS`$(rr(\mathbf{B}), d, M, J)$;
      if $T_{\min} < T_{\text{total}}$ **then**
        $S_{\min}$, $T_{\min}$, $m_1 \leftarrow S$, T_{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 11: Our LWE Samples Selection Algorithm

Using the optimization strategy for LWE instance number selection, we can solve challenges faster than G6K default strategy. See the table 6.

Table 6. 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%

6 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 $(n, \alpha) \in \{(40, 0.035), (90, 0.005), (50, 0.025), (55, 0.020)\}$ in TU Darmstadt LWE challenge website. (We solved the LWE instance $(n, \alpha) = (80, 0.005)$ earlier, but we only use an improved heuristic blocksize strategy, not the optimized strategy presented in this paper.)

6.1 Simulation Results and Comparison

We use a workstation with Intel Xeon 5128 16c 32@2.3GHz, 1.48T RAM and NVIDIA GTX 3090 * 2 to construct our simulating algorithms, and also for running tests and experiments.

The comparison between simulating results for different modes are given below:

Table 7. Simulated result of Mode comparison.

(n,α)	Default G6K (h)	BKZ-only (h)	pro-pnj-BKZ (h)
(95,0.005)	45720	3786242	6662
(100,0.005)	921308	91646709	150395
(40,0.040)	77350	276700	3144
(45,0.035)	676715	5463845	48242

As Table 7 shown, for the LWE instance $(40, 0.040)$, we have an increase of about 24.6 times compared to the default pnj-BKZ in G6K. Moreover, for the LWE instance $(100, 0.005)$ we have an increase of about 611 times compared to a BKZ-only algorithm.

6.2 Actual Wall Time for Solving LWE Challenges

We have solved the LWE instances $(n, \alpha) \in \{(40, 0.035), (50, 0.025), (55, 0.020)\}$ in TU Darmstadt LWE challenge website by a service with AMD EPYC^{TM} 7002 Series 128@2.6GHz, NVIDIA RTX 3090 * 8, 1.5T RAM (denoted as Machine A), and solved $(n, \alpha) = (90,0.005)$ by a service with AMD EPYC^{TM} 7002 Series 64@2.6GHz, a100 *4, 512 GB RAM (denoted as Machine B). Since the original solving strategy used in solving $(n, \alpha) = (40, 0.035)$ is generated by BSSA instead of EnumBS, we rerun the solving algorithm using a strategy generated by EnumBS on our test machine in Section 6.1 (denoted as Machine C).

We have listed the the actual running time and RAM cost in solving the above LWE challenges in Table 8. The unit of running time in Table 8 is hour and the unit of RAM in it is GB.

We can see that our simulated time is close to the wall time we used in solving LWE challenges from Table 8. Since machine A is faster than machine C, and machine B is slower than machine C, the wall time for solving $(50, 0.025), (55, 0.020)$ (resp. $(90, 0.005)$) is also smaller (resp. larger) then the simulated time. For $(40, 0.035)$, it takes more time since the length of LWE error term is relatively large hence failed to be solved with $P_{\text{success}} = 0.8$.

Table 8. Actual running time, RAM cost and simulation cost.

(n,α)	Machine	Walltime (h)	RAM (GB)	Simulated Time (h) ‡‡
(40,0.035)	C	79	384	57
(50,0.025)	A	592	184	868
(55,0.020)	A	611	890	835
(90,0.005)	B	370	332	278

‡‡ Using the time cost model in Section 3.3 based on machine C to simulate the time cost of pro-pnj-BKZ.

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) and two time cost models (pnj-BKZ time cost model and Pump time cost model). Experimental results show that our simulators can accurately predict the behavior of pnj-BKZ even if jump>1 and the behavior of Pump respectively. Besides, both our time cost models fit well with the walltime of experiments. We propose two-step mode and prove that this model is indeed better than default mode in G6K and BKZ only mode. We design two new blocksize strategy selection algorithms: BSSA and EnumBS, and demonstrate the optimality of the EnumBS strategy. Meanwhile, applying the blocksize strategy generated from EnumBS to solve the LWE Challenge results in at most 24.6 times (611 times respectively) improvement compared to default G6K mode (BKZ only mode respectively) and help us to solve the TU Darmstadt LWE challenges $(n, \alpha) \in \{(40, 0.035),(90, 0.005),(50, 0.025),(55, 0.020)\}$. What's more, we also propose a new LWE samples selection strategy which saves at most 5% time cost in our new algorithm compared to the original one in G6K.

7.2 Future Work

Time Cost Model When constructing the time cost model of Pump in section 3.3, we found that the time cost of each Pump increases linearly with the increase of index of Pump (index $< d - f$) in each tour of pnj-BKZ. Meanwhile, the increasing slope of this Pump cost with increasing index seems to be proportional to the blocksize of pnj-BKZ. It is also unclear how to accurately describe the relationship between the growth slope and the blocksize of pnj-BKZ. We don't know why these phenomenona occur, and we plan to find out the reason in future.

Dual Attack During the writing of this paper, we noticed that compared with primal attack, the dual attack with FFT and modulus exchange technology seems to have less computational complexity in solving the LWE problem. Using this dual attack, one can efficiently determine whether a part of the private key vector guessed are correct or not, and finally recover partial component of private key vector to decrease the dimension of LWE problem. But there is currently no implementation of this efficient dual attack, so we plan to implement this dual attack and combined it with our optimized lattice reduce strategy to solve higher dimensional LWE challenges.

Code Improvement We hope to improve the interaction between CPU and GPU, try to optimize the code of "gpu_sieve" implemented in C++ and CUDA code in G6K-GPU. Also, we will try to solve the saturation warning occurred in G6K/G6K-GPU.

Security Estimation of LWE We will re-estimate the security of LWE-based cryptosystems. We have a more accurate simulator for the efficient lattice reduction algorithm pnj-BKZ algorithm ($jump \geq 1$) in section 3.2 and give an optimized lattice reduction strategies in section 2.7 which can help we to give a more accurate security hardness estimation of these main LWE-based schemes.

Blocksizes Strategy Selection Algorithm Although the EnumBS could obtain the optimal blocksize strategy, its theoretical complexity is exponential, though it is more efficient than a theoretical guess in a practical test. In future work, we hope to improve the blocksize strategy selection algorithm in solving efficiency.

References

ABD+20. Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. Kyber(Round 3). page 42, 2020.

ABLR21. Martin R. Albrecht, Shi Bai, Jianwei Li, and Joe Rowell. Lattice Reduction with Approximate Enumeration Oracles. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology – CRYPTO 2021*, Lecture Notes in Computer Science, pages 732–759, Cham, 2021. Springer International Publishing.

ADH+19. Martin R. Albrecht, Léo Ducas, Gottfried Herold, Elena Kirshanova, Eamonn W. Postlethwaite, and Marc Stevens. The General Sieve Kernel and New Records in Lattice Reduction. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2019*, pages 717–746, Cham, 2019. Springer International Publishing.

ADPS16. Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. Post-quantum Key Exchange—A New Hope. pages 327–343, 2016.

AFG14. Martin R. Albrecht, Robert Fitzpatrick, and Florian Göpfert. On the Efficacy of Solving LWE by Reduction to Unique-SVP. In Hyang-Sook Lee and Dong-Guk Han, editors, *Information Security and Cryptology – ICISC 2013*, pages 293–310, Cham, 2014. Springer International Publishing.

AGVW17. Martin R. Albrecht, Florian Göpfert, Fernando Virdia, and Thomas Wunderer. Revisiting the Expected Cost of Solving uSVP and Applications to LWE. In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology – ASIACRYPT 2017*, pages 297–322, Cham, 2017. Springer International Publishing.

APS15. Martin R. Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of Learning with Errors. *Journal of Mathematical Cryptology*, 9(3), January 2015.

AWHT16. Yoshinori Aono, Yuntao Wang, Takuya Hayashi, and Tsuyoshi Takagi. Improved Progressive BKZ Algorithms and Their Precise Cost Estimation by Sharp Simulator. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology – EUROCRYPT 2016*, Lecture Notes in Computer Science, pages 789–819, Berlin, Heidelberg, 2016. Springer.

Bab86. L. Babai. On Lovász' lattice reduction and the nearest lattice point problem. *Combinatorica*, 6(1):1–13, March 1986.

BDGL16. Anja Becker, Léo Ducas, Nicolas Gama, and Thijs Laarhoven. New directions in nearest neighbor searching with applications to lattice sieving. In *Proceedings of the twenty-seventh annual ACM-SIAM symposium on Discrete algorithms*, SODA '16, pages 10–24, USA, January 2016. Society for Industrial and Applied Mathematics.

BG14. Shi Bai and Steven D. Galbraith. An Improved Compression Technique for Signatures Based on Learning with Errors. In Josh Benaloh, editor, *Topics in Cryptology – CT-RSA 2014*, pages 28–47, Cham, 2014. Springer International Publishing.

BGJ15. Anja Becker, Nicolas Gama, and Antoine Joux. Speeding up lattice sieving without increasing the memory, using sub-quadratic nearest neighbor search, 2015.

BL16. Anja Becker and Thijs Laarhoven. Efficient (Ideal) Lattice Sieving Using Cross-Polytope LSH. In David Pointcheval, Abderrahmane Nitaj, and Tajjeeddine Rachidi, editors, *Progress in Cryptology – AFRICACRYPT 2016*, pages 3–23, Cham, 2016. Springer International Publishing.

BSW18. Shi Bai, Damien Stehlé, and Weiqiang Wen. Measuring, Simulating and Exploiting the Head Concavity Phenomenon in BKZ. In Thomas Peyrin and Steven Galbraith, editors, *Advances in Cryptology – ASIACRYPT 2018*, Lecture Notes in Computer Science, pages 369–404, Cham, 2018. Springer International Publishing.

Che13. Phong-Quang Chen, Yuanmi; Nguyen. *Réduction de réseau et sécurité concrète du chiffrement complètement homomorphe*. PhD Thesis, 2013.

CN11. Yuanmi Chen and Phong Q. Nguyen. BKZ 2.0: Better Lattice Security Estimates. In Dong Hoon Lee and Xiaoyun Wang, editors, *Advances in Cryptology – ASIACRYPT 2011*, Lecture Notes in Computer Science, pages 1–20, Berlin, Heidelberg, 2011. Springer.

DEKL+20. Léo Ducas, Tancrède Lepoint Eike Kiltz, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. *Dilithium(Round 3)*. NIST PQC probject, 2020.

DSDGR20. Dana Dachman-Soled, Léo Ducas, Huijing Gong, and Mélissa 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*, pages 329–358, Berlin, Heidelberg, August 2020. Springer-Verlag.

DSvW21. Léo Ducas, Marc Stevens, and Wessel van Woerden. Advanced Lattice Sieving on GPUs, with Tensor Cores. In Anne Canteaut and François-Xavier Standaert, editors, *Advances in Cryptology – EUROCRYPT 2021*, Lecture Notes in Computer Science, pages 249–279, Cham, 2021. Springer International Publishing.

Duc18. Léo Ducas. Shortest Vector from Lattice Sieving: A Few Dimensions for Free. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018*, pages 125–145, Cham, 2018. Springer International Publishing.

FBB+15. Robert Fitzpatrick, Christian Bischof, Johannes Buchmann, Özgür Dagdelen, Florian Göpfert, Artur Mariano, and Bo-Yin Yang. Tuning GaussSieve for Speed. In Diego F. Aranha and Alfred Menezes, editors, *Progress in Cryptology - LATINCRYPT 2014*, volume 8895, pages 288–305. Springer International Publishing, Cham, 2015. Series Title: Lecture Notes in Computer Science.

GNR10. Nicolas Gama, Phong Q. Nguyen, and Oded Regev. Lattice Enumeration Using Extreme Pruning. In David Hutchison, Takeo Kanade, Josef Kittler, Jon M. Kleinberg, Friedemann Mattern, John C. Mitchell, Moni Naor, Oscar Nierstrasz, C. Pandu Rangan, Bernhard Steffen, Madhu Sudan, Demetri Terzopoulos, Doug Tygar, Moshe Y. Vardi, Gerhard Weikum, and Henri Gilbert, editors, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110, pages 257–278. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010. Series Title: Lecture Notes in Computer Science.

HK17. Gottfried Herold and Elena Kirshanova. Improved Algorithms for the Approximate k-List Problem in Euclidean Norm. In Serge Fehr, editor, *Public-Key Cryptography – PKC 2017*, Lecture Notes in Computer Science, pages 16–40, Berlin, Heidelberg, 2017. Springer.

HKL18. Gottfried Herold, Elena Kirshanova, and Thijs Laarhoven. Speed-Ups and Time–Memory Trade-Offs for Tuple Lattice Sieving. In *Public-Key Cryptography – PKC 2018*, pages 407–436. Springer, Cham, March 2018.

HPS11. Guillaume Hanrot, Xavier Pujol, and Damien Stehlé. Analyzing Blockwise Lattice Algorithms Using Dynamical Systems. In Phillip Rogaway, editor, *Advances in Cryptology – CRYPTO 2011*, Lecture Notes in Computer Science, pages 447–464, Berlin, Heidelberg, 2011. Springer.

Kan83. Ravi Kannan. Improved algorithms for integer programming and related lattice problems. In *Proceedings of the fifteenth annual ACM symposium on Theory of computing*, STOC '83, pages 193–206, New York, NY, USA, December 1983. Association for Computing Machinery.

Laa15. Thijs Laarhoven. Sieving for Shortest Vectors in Lattices Using Angular Locality-Sensitive Hashing. In Rosario Gennaro and Matthew Robshaw, editors, *Advances in Cryptology – CRYPTO 2015*, Lecture Notes in Computer Science, pages 3–22, Berlin, Heidelberg, 2015. Springer.

LLL82. A. K. Lenstra, H. W. Lenstra, and L. Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261(4):515–534, December 1982.

LM09. Vadim Lyubashevsky and Daniele Micciancio. On Bounded Distance Decoding, Unique Shortest Vectors, and the Minimum Distance Problem. In Shai Halevi, editor, *Advances in Cryptology - CRYPTO 2009*, volume 5677, pages 577–594. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009. Series Title: Lecture Notes in Computer Science.

LM18. Thijs Laarhoven and Artur Mariano. Progressive Lattice Sieving. In Tanja Lange and Rainer Steinwandt, editors, *Post-Quantum Cryptography*, pages 292–311, Cham, 2018. Springer International Publishing.

LPR10. Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On Ideal Lattices and Learning with Errors over Rings. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, Lecture Notes in Computer Science, pages 1–23, Berlin, Heidelberg, 2010. Springer.

MV10. Daniele Micciancio and Panagiotis Voulgaris. Faster exponential time algorithms for the shortest vector problem. In *Proceedings of the 2010 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, Proceedings, pages 1468–1480. Society for Industrial and Applied Mathematics, January 2010.

NV08. Phong Q. Nguyen and Thomas Vidick. Sieve algorithms for the shortest vector problem are practical. *Journal of Mathematical Cryptology*, 2(2), January 2008.

Pei16. Chris Peikert. A Decade of Lattice Cryptography. *Found. Trends Theor. Comput. Sci.*, 10(4):283–424, March 2016. Place: Hanover, MA, USA Publisher: Now Publishers Inc.

PV21. Eamonn W. Postlethwaite and Fernando Virdia. On the Success Probability of Solving Unique SVP via BKZ. In Juan A. Garay, editor, *Public-Key Cryptography – PKC 2021*, volume 12710, pages 68–98. Springer International Publishing, Cham, 2021. Series Title: Lecture Notes in Computer Science.

Reg09. Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM*, 56(6):34:1–34:40, September 2009.

Sch91. C. P. Schnorr. Factoring Integers and Computing Discrete Logarithms via Diophantine Approximation. In Donald W. Davies, editor, *Advances in Cryptology — EUROCRYPT '91*, Lecture Notes in Computer Science, pages 281–293, Berlin, Heidelberg, 1991. Springer.

SE91. C. P. Schnorr and M. Euchner. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. In L. Budach, editor, *Fundamentals of Computation Theory*, Lecture Notes in Computer Science, pages 68–85, Berlin, Heidelberg, 1991. Springer.

Xag10. Keita Xagawa. Cryptography with Lattices. page 244, 2010.

A Pump Simulator

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 proves 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 2.5.

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 12: Pump Simulator