






Refined Strategy for Solving LWE in Two-step Mode

Wenwen Xia^{1,2,4}^{**} , Leizhang Wang³^{**} , Geng Wang^{1,2}^{*} , Dawu Gu¹^{*} , and
Baocang Wang³^{*} 

¹ School of Electronic, Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai, 200240, China

{xww_summer, wanggxx, dwgu}@sjtu.edu.cn

² State Key Laboratory of Cryptology, P.O.Box 5159, Beijing 100878, China

³ State Key Laboratory of Integrated Service Networks, Xidian University, Xi'an, 710071, China

⁴ School of Cyber Engineering, Xidian University, Xi'an, 710071, China
lzwang_2@stu.xidian.edu.cn, bcwang@xidian.edu.cn

Abstract. Learning with Errors (LWE) and its variants are widely used in constructing lattice-based cryptographic schemes, including NIST standards Kyber and Dilithium, and a refined estimation of LWE's hardness is crucial for their security. Currently, primal attack is considered the fastest algorithm for solving LWE problem in practice. It reduces LWE to a unique Shortest Vector Problem (uSVP) and combines lattice reduction algorithms with SVP calls such as enumeration or sieving. However, finding the most time-efficient combination strategy for these algorithms remains a challenge. The designers of Kyber highlighted this issue as open problem Q7: "A refined (progressive) lattice reduction strategy and a precise analysis of the gains using reduction preprocessing plus a single SVP call in large dimensions are still missing."

In this paper, we address this problem by presenting a Strategy Search algorithm named PSSearch for solving uSVP and LWE, using progressive BKZ as the lattice reduction and sieving as the SVP call. Compared to the heuristic strategy used in G6K (Albrechet et al., Eurocrypt 2019), the strategy generated by our algorithm has the following advantages: (1) We design a tree search algorithm with pruning named PSSearch to find the minimal time-cost strategy in two-step mode and prove its correctness, showing that the fastest approach in two-step mode for solving uSVP and LWE can be achieved in a reasonable timeframe; (2) We propose the first tight simulation for BKZ that can jump by $J > 1$ blocks, which allows us to choose more flexible jump values to improve reduction efficiency. (3) We propose a refined dimension estimation method for the SVP call. We tested the accuracy of our new simulation algorithm and the efficiency of our new strategy through experiments. Furthermore, we apply the strategies generated by SSearch to solve the TU Darmstadt LWE Challenges with $(n, \alpha) \in \{(80, 0.005), (40, 0.035), (90, 0.005), (50, 0.025),$

* Corresponding author.

** Wenwen Xia and Leizhang Wang are the co-first authors of this work.

(55, 0.020), (40, 0.040)} using the G6K framework, achieving improvements of 7.2 to 23.4 times over the heuristic strategy employed in G6K. By combining the minimal time-cost strategy selection with the refined two-step estimator for LWE (Xia et al., PKC 2024), we re-estimate the hardness of NIST standards Kyber and Dilithium, and determine the influence of the strategy. Specifically, the security levels of NIST standards decrease by 3.4 to 4.6 bits, rather than 2 to 8 bits indicated in the Kyber documentation. It achieves a decrease of 1.1 to 1.3 bits compared to the refined two-step estimation using trivial strategy.

Keywords: Lattice Cryptanalysis · Progressive Reduction Strategy · G6K · PnJBKZ Simulator · Concrete Hardness Estimation.

1 Introduction

Learning with Errors (LWE) [1] plays an important role in lattice-based cryptography, as the security of a large fraction of lattice-based cryptographic schemes [2–5] relies on the hardness of LWE or its variants [1, 6–8], including NIST post-quantum standards Kyber and Dilithium [8, 9]. So to provide concrete security levels of these schemes, a tight estimation on the hardness of LWE is necessary. This means that we need to identify the most efficient algorithm for solving LWE, along with its precise complexity.

Among the many different methods for solving LWE, primal attack [10] is considered the most efficient in practice currently. Primal attack uses Kannan’s embedding technique [11] to transform LWE into the unique Shortest Vector Problem (uSVP) on a specific lattice, which contains an extraordinary short vector, known as the unique shortest vector. If we assume that, after certain level of reduction on the lattice, the projection of the unique shortest vector becomes the shortest vector on a projected sub-lattice, we can recover it by first finding the shortest vector on the sub-lattice using an SVP oracle and then using Babai’s lifting [12] to lift it onto the full-dimensional lattice. This approach was first suggested in [13], verified by [14], and has already been used in many LWE estimators, such as [15].

The choice of lattice reduction and SVP solving algorithms is important in LWE primal attack. In the literature, cryptanalysts often use BKZ [16] as the lattice reduction algorithm, which has a tunable parameter β called blocksize to balance between time cost and reduction effectiveness, and use either enumeration or lattice sieving as the SVP oracle. Although the efficiency could be further improved by finding better algorithms for lattice reduction or SVP solving, designing these fundamental algorithms seems to be extremely hard. However, even if we fix these fundamental algorithms, there is still room for improvement if we choose a better solving strategy. Here, “strategies” refers to the different ways of calling the fundamental algorithms with various parameters.

All LWE solving algorithms and LWE estimators use certain strategies, although some of them are presented implicitly. In BKZ 2.0 [17] and ADPS16 [18], BKZ with a fixed blocksize is called multiple times (referred to as tours) until

a short vector is found, while the SVP call is not called outside BKZ. This is later improved by a progressive strategy, where the blocksize is increased by each BKZ tour. Aono et al. presented the Improved Progressive BKZ (ProBKZ) [19] with an explicit strategy selection algorithm to output the sequence of blocksizes for BKZ tours. They also added an individual SVP call after BKZ, but this SVP call is made on the full-dimensional lattice basis by enumeration. All of these algorithms use enumeration as the SVP call, which is later been outperformed by lattice sieving in higher dimensions. In the LWE estimator designed by Albrecht et al. [15], sieving is used as the SVP call, and they presented two different modes for estimating the complexity of LWE primal attack, one mode involves progressive BKZ with blocksize added by 1 for each tour, while the other is a two-step mode that combines progressive BKZ with a final sieving call, where the dimension is chosen to balance the cost between the two steps. The concept of “two-step” is illustrated in Fig. 1. In 2019, Albrecht et al. designed a lattice solving framework called G6K [20], which implements various sieving algorithms [18, 21–25] and uses individual techniques [26, 27] to accelerate lattice sieving. They also implemented an LWE solving algorithm in G6K, where each BKZ tour is followed by a conditional sieve and it will be triggered heuristically. Besides, there is no guarantee that the sieving will yield a solution for LWE because the dimension estimation method in ADPS16 [18] has a non-negligible failure probability for solving LWE.



Fig. 1: Two-step Mode

We can see that all these strategies are heuristic, with no guarantees of optimality. In the Kyber document [8], the designers raised an open problem Q7: “A refined (progressive) lattice reduction strategy and a precise analysis of the gains using reduction preprocessing plus a single SVP call in large dimensions are still missing.” The designers expect that, by optimizing the strategy for BKZ blocksizes and final SVP dimension, the security estimation for Kyber might be lowered by 2 to 8 bits.

In this paper, we solve this open problem by presenting an minimal time-cost strategy for uSVP and LWE in primal attack. First, we note that, all possible solving strategies form a strategy space with finite elements, thus the minimal time-cost strategy can be found by searching over the strategy space. However, there are three main issues to be solved: (1) Give a formal definition for the strategy space, such that the strategy space contains all reasonable strategies that succeed with high probability. (2) Since the strategy space is exponentially large, we must find a way to reduce the complexity of the strategy searching algorithm, so that the minimal time-cost strategy can be found in reasonable time. (3) For the search algorithm to function effectively, we need to provide a precise time cost estimation for each strategy in the strategy space.

1.1 Our Contribution

We list our contribution in this work as follows:

1. We define the strategy space in the two-step mode, which contains all the two-step strategy with a high success probability.

2. We design a tree search algorithm on the strategy space with pruning, named Pruning Strategy Search (PSSearch). We prove that PSSearch can output the minimal time-cost two-step strategy for solving LWE in the strategy space, so the PSSearch can be used to accelerate LWE solver, as well as present tight estimation for LWE-based cryptographic schemes.

3. We implement PSSearch using the most efficient lattice reduction and SVP algorithm up to date, which are PnJBKZ and Pump in G6K. More concretely, we design a simulator for PnJBKZ which can simulate the behavior for $\text{jump} > 1$, to support searching more flexible and efficient reduction strategy. We also give a refined dimension estimation method for the SVP call considering the distribution of the LWE noise vector.

Particularly, with the minimal time-cost two-step strategy for solving LWE generated by PSSearch, we successfully cracked six previously unsolved TU Damstadt LWE Challenges¹, which is 23.4 times faster than the previously most efficient LWE solver, G6K [28]. See Figs. 2, 5 and Table 4. In Reduction Step, the time cost needed to achieve the same basis quality is faster than the trivial reduction strategy, by a factor of 4 to 36.4 times. See Fig. 6. The executable code of TwoStepSolver based on PSSearch is publicly available on GitHub².

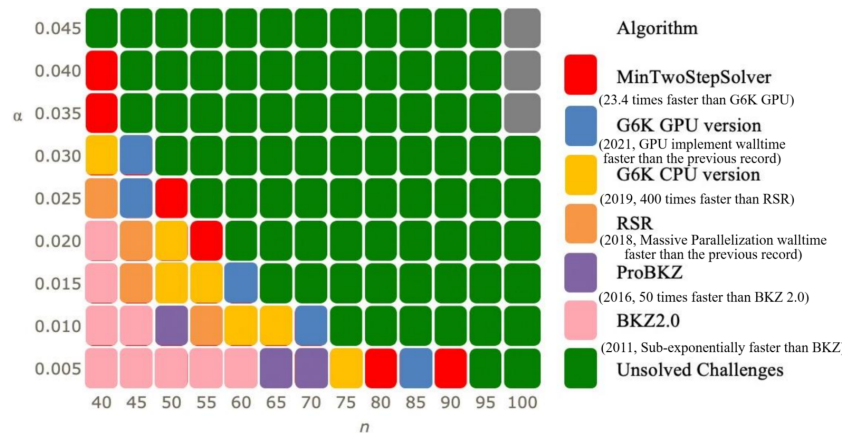


Fig. 2: LWE Challenges and the Algorithms to Solve it

Our new hardness estimation for NIST standards Kyber and Dilithium show that when using the time cost minimal strategy, the security levels is decreased by 3.4 to 4.6 bits rather than 2 to 8 bits indicated in the Kyber documentation, which confirms the Q7 in the document of Kyber. Compared to the trivial

¹ https://www.latticechallenge.org/lwe_challenge/challenge.php

² <https://github.com/Summwer/pro-pnj-bkz>

strategy used in the “Refined Two-Step Mode LWE Estimator” [29], the security levels decrease by 1.1 to 1.3 bits. See Table 5.

1.2 Related Works

Lattice Solving Algorithms. BKZ [16] is the most popular lattice reduction algorithm typically used for solving LWE. BKZ combines the LLL [30] and an SVP algorithm to balance the time cost and the success probability using blocksize β . Many cryptanalysts improved the BKZ algorithm, e.g. the extreme pruning [31] to speed up enumeration, BKZ 2.0 [17] based on [31], approximate enumeration oracle [32], and progressive reduction parameters optimization in BKZ such as *Improved Progressive BKZ* (ProBKZ) [19]. [14] showed that the unique shortest vector is recovered by first finding its projection in a projected sublattice then lifting it to the full lattice and verifying the BKZ successful condition of solving LWE in [33]. In 2019, Albrecht et al. [20] designed the *General Sieve Kernel* (G6K) and implemented the progressive sieve [27] named Pump, which can selectively call the Gauss sieve [21, 22], NV sieve [34], k -list sieve [23, 24] or BGJ1 sieve [25]. Progressive sieve and dimension-for-free (d4f) technique [26] are used in Pump for acceleration, where Pump is a progressive sieve and insertion algorithm that sieves on the projected sublattice and can insert more than one vector into the lattice basis. Ducas et al. [28] improved G6K using GPU (named G6K-GPU-Tensor) and implemented the fastest sieving algorithm BDGL16 [18] in both G6K and G6K-GPU-Tensor. G6K provides an algorithm to solve LWE, which will conditionally call Pump to find short vectors on the projected sublattice and lift them into the full lattice basis after running several tours of PnJBKZ. PnJBKZ uses Pump as its SVP oracle so that even if it calls the SVP oracle with a jump value, it still can ensure the skipped basis vector is reduced by Pump. It solves TU Darmstadt LWE Challenges 400 times faster than the previous records.¹

Lattice Reduction Simulators. The first BKZ simulator is proposed in [17], which uses the Gaussian heuristic to predict BKZ- β , where we refer to the BKZ 2.0 simulator. In 2016, while proposing Progressive BKZ [19], Aono et al. introduced a new BKZ simulator for predict a fully BKZ- β reduced basis. In 2017, Yu and Ducas [35] conducted extensive experiments to evaluate the practical behavior of BKZ. They provided a detailed study of the distribution of Gram-Schmidt vector lengths for BKZ-reduced bases and more accurately quantified the “head concavity” phenomenon based on their observations. In 2018, Bai, Stehle, and Wen [36] proposed an even more accurate BKZ simulator by considering the distribution of short vectors in random lattices. The simulator developed by Bai et al. [36] can predict the “head concavity” phenomenon in the Gram-Schmidt norm curves after BKZ- β reduction with high accuracy. However, since the simulator proposed by Bai et al. [36] accounts for the distribution of random lattice vectors, it is a randomized algorithm, meaning its predictions are stochastic

¹ https://github.com/WvanWoerden/G6K-GPU-Tensor/blob/main/lwe_challenge.py

rather than deterministic. The prediction results will only converge after running BKZ- β with the same fixed blocksize β for a sufficiently large number of reduction tours. In practice, to achieve higher reduction efficiency, it is often impractical to run a large number of tours with the same blocksize β . In many heuristic progressive reduction strategies [17,20,28,37], run only one tour of BKZ reduction for each blocksize β . Additionally, the randomized simulator cannot accurately estimate the concrete security strength of cryptographic schemes if the evaluation strategy involves only one tour of BKZ reduction for each progressive blocksize β and the estimation results do not converge. Therefore, we do not base our construction of the PnJBKZ simulator on the aforementioned BKZ 2.0 simulator variants.

Roadmap. The paper is organized as Fig. 3. Sec. 2 presents the notations and preliminaries. We first formalize the definitions of the strategy space for solving LWE and give a Simplified Strategy Search (SSearch) method in Section 3, where the SSearch can be determined under a given reduction simulator named `ReductionSim` for lattice reduction algorithm \mathcal{R} , a dimension estimation method named `SVPDimEst` of the SVP call \mathcal{C} and a time cost model. Next, we give a pruning version of SSearch (PSSearch) in Sec. 4 and prove its correctness of outputting the minimal time cost strategy for solving LWE. In Sec. 5, we propose a PnJBKZ simulator for instantiating PSSearch with the lattice reduction algorithm PnJBKZ. Finally, we give the experiments in Sec. 6 as follows, (1) Give the accuracy verification of PnJBKZ simulator; (2) Compare the time cost of the LWE solver implemented in G6K GPU version with our two-step LWE solver using strategy generated by PSSearch and the practical time cost model proposed in Sec. 2.6; (3) Present the optimized strategy for solving the LWE Challenge; (4) Verify the simulation accuracy of MinTwoStepSolver; (5) Give the detail about new LWE records; (6) Re-estimate NIST schemes Kyber and Dilithium base on PSSearch and gate count model [38].

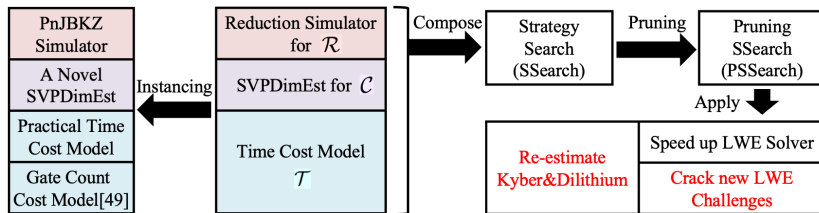


Fig. 3: Roadmap

2 Preliminaries

We denote vectors by lower-case bold letters, e.g. $\mathbf{a}, \mathbf{b}, \mathbf{c}, \dots$, and matrices by upper-case bold letters, e.g. $\mathbf{A}, \mathbf{B}, \mathbf{C}, \dots$. For a matrix $\mathbf{B} = (\mathbf{b}_0, \dots, \mathbf{b}_{d-1})$, we write \mathbf{b}_i as its $i + 1$ -th column vector. The Euclidean norm of a vector $\mathbf{v} \in \mathbb{R}^m$ is denoted by $\|\mathbf{v}\|$. We write $\langle \cdot, \cdot \rangle$ for inner products and \cdot for matrix-vector

products. By abuse of notation we consider vectors to be row resp. column vectors depending on context, such that $\mathbf{v} \cdot \mathbf{A}$ and $\mathbf{A} \cdot \mathbf{v}$ are both meaningful. We write $\mathbf{0}_{m \times d}$ as the $m \times d$ all zero matrix. If the dimensions are clear from context, we may omit the subscripts. $|\mathbf{B}|$ is denoted as the absolute value of the determinant of matrix \mathbf{B} . $A := \llbracket a_0, \dots, a_{n-1} \rrbracket$ is denoted as an ordered list with size $\sharp A = n$ and $A[i] = a_i$, while $\llbracket a_0, \dots, a_{n-1} \rrbracket \cup \llbracket a_n \rrbracket = \llbracket a_0, \dots, a_{n-1}, a_n \rrbracket$. We define $[a, b] := \{a, a+1, \dots, b-1\}$ and $[d] = [0, d]$.

2.1 Lattices

Let \mathcal{L} be a d -dimensional lattice in \mathbb{R}^m and its basis be $\mathbf{B} \in \mathbb{R}^{m \times d}$, we denote $\mathbf{B}_{[i,j]} := (\mathbf{b}_i, \dots, \mathbf{b}_{j-1})$. We can also denote the lattice generated by basis \mathbf{B} as $\mathcal{L}(\mathbf{B})$. Let $\mathbf{B}^* = (\mathbf{b}_0^*, \dots, \mathbf{b}_{d-1}^*)$ be the Gram-Schmidt orthogonalization of \mathbf{B} , in which $\mathbf{b}_i^* = \mathbf{b}_i - \sum_{j=0}^{i-1} \mu_{i,j} \mathbf{b}_j^*$, $\mu_{i,j} = \langle \mathbf{b}_i, \mathbf{b}_j^* \rangle / \|\mathbf{b}_j^*\|^2$, $\forall i \in [d]$. Denote by l_i the logarithm of Gram-Schmidt norm, i.e. $l_i = \ln(\|\mathbf{b}_i^*\|)$, for $i \in \{0, \dots, d-1\}$. Let $\text{rr}(\mathbf{B}) = (l_0, \dots, l_{d-1})$, abbreviate to rr , $\text{rr}_{[i,j]} = (l_i, \dots, l_{j-1})$. For $i \in [d]$, we denote by π_i the orthogonal projection over $(\mathbf{b}_0, \dots, \mathbf{b}_{i-1})^\perp$. For $0 \leq j \leq k \leq d-1$, we denote by $\mathbf{B}_{\pi[j,k]}$ the local projected block $(\pi_j(\mathbf{b}_j), \pi_j(\mathbf{b}_{j+1}), \dots, \pi_j(\mathbf{b}_{k-1}))$, and by $\mathcal{L}_{\pi[j,k]}$ the lattice spanned by $\mathbf{B}_{\pi[j,k]}$.

The volume of a lattice $\text{Vol}(\mathcal{L})$ is defined as $\text{Vol}(\mathcal{L}) = |\mathbf{B}| = \prod_{i=0}^{d-1} \|\mathbf{b}_i^*\|$. We write $\lambda_i(\mathcal{L})$ for Minkowski's successive minima, the Gaussian Heuristic predicts $\lambda_1(\mathcal{L}) \approx \text{GH}(\mathbf{B}) = \sqrt{\frac{d}{2\pi e}} |\mathbf{B}|^{1/d}$.

2.2 Lattice Reduction

Definition 1 (Size-reduced). *The Gram-Schmidt coefficients of a d -dimension lattice basis \mathbf{B} set as $\mu_{i,j}$ for any $1 \leq j < i \leq d$. Then the basis \mathbf{B} is size-reduced if following holds: For $1 \leq j < i \leq d$: $|\mu_{i,j}| \leq 1/2$.*

Definition 2 (Hermite-Korkine-Zolotarev and Block-Korkine-Zolotarev reductions [39]). *The basis \mathbf{B} of a d -dimensional lattice \mathcal{L} is HKZ reduced if \mathbf{B} is size-reduced and $\|\mathbf{b}_i^*\| = \lambda_1(\mathcal{L}_{\pi[i,d]})$, for all $i < d$. A d -dimensional \mathcal{L} is BKZ- β reduced if \mathbf{B} is size-reduced and $\|\mathbf{b}_i^*\| = \lambda_1(\mathcal{L}_{\pi[i, \min\{i+\beta, d\}]})$, for all $i < d$.*

In this paper, when we refer to the property of HKZ reduction with respect to a blocksize β , we mean that the HKZ reduced property holds for a β -sized block, i.e. $\mathbf{B}_{\pi[j, j+\beta]}$.

Definition 3 (Root Hermite Factor). *For a basis \mathbf{B} of d -dimensional lattice, the root Hermite factor is defined as $\delta(\mathbf{B}) = (\|\mathbf{b}_0\| / |\mathbf{B}|^{1/d})^{1/d}$. For larger block-size of BKZ, it follows the asymptotic formula $\delta(\beta)^{2(\beta-1)} = \frac{\beta}{2\pi e} (\beta\pi)^{1/\beta}$ [40]. $\delta(\mathbf{B})$ can be used to measure current lattice basis quality of the lattice basis \mathbf{B} . A better lattice basis quality of \mathbf{B} corresponds to a smaller $\delta(\mathbf{B})$.*

Heuristic 1 (Geometric Series Assumption [20]) *Let \mathbf{B} be a lattice basis after the lattice reduction, then $\|\mathbf{b}_i^*\| \approx \alpha \cdot \|\mathbf{b}_{i-1}^*\|$, $0 < \alpha < 1$.*

Combine the GSA with root-Hermite factor (Definition 3) and $\text{Vol}(\mathcal{L}) = \prod_{i=0}^{d-1} \|\mathbf{b}_i^*\|$, it infers that $\alpha = \delta^{-\frac{2d}{d-1}} \approx \delta^{-2}$. Let s be the slope value of the logarithm of GS norms l_i for $\forall i \in \{1, \dots, d\}$, $s \approx \ln \alpha$ and $\delta \approx e^{-\frac{s}{2}}$.

Even if GSA does not strictly hold, we can still apply least squares fitting on $\ln \|\mathbf{b}_i^*\|$ to estimate the slope s . In earlier works such as [20], s is a measurement for basis quality: the basis quality becomes better when s is closer to 0.

Heuristic 2 (Sandpile Model Assumption (SMA) [41]) *For any HKZ reduced basis $(b_i)_{i \leq \beta}$, $l_i = \frac{1}{2} \ln \gamma_{\beta-i+1} + \frac{1}{\beta-i+1} \sum_{j=i}^{\beta} l_j$ for all $i \leq \beta$ with $(l_i = \log \|\mathbf{b}_i^*\|)_{i \leq \beta}$. Here γ_i is the i -dimension Hermite's constant which equals to $\lambda_1(L)/(\det L)^{1/\dim(L)}$. We use $\sqrt{\frac{\dim(L)}{2\pi e}}$ to approximate this $\gamma_{\dim(L)}$.*

2.3 Sieving Algorithms and Progressive Sieve

The first practical NV sieving algorithm uses a database of $N_0 = O(2^{0.2075d})$ vectors and runs in time $N_0^2 = O(2^{0.415d})$ by repeatedly checking all pairs $\mathbf{v} \pm \mathbf{w}$ [34]. To find the shortest vector, N_0 is the minimal number of vectors to ensure saturating the ball of radius $\text{GH}(\mathcal{L}) \sqrt{4/3}$ by short vector. In a line of works [18, 25, 42, 43] the time complexity was gradually decreased to $O(2^{0.292d})$ by nearest neighbor searching techniques. The progressive sieve [27] can save the cost of the classical sieve, which was later implemented in [20, 28] through a right-to-left operation. A progressive sieve first calls a sieve on a small dimensional projected lattice, then uses Babai's nearest plane algorithm [12] to lift the vectors to a higher dimensional projected lattice. It repeats such a step until the short vectors are lifted onto the full dimensional lattice.

2.4 Technologies in G6K

G6K [20] is an abstract machine for running sieve and reduction algorithms, which is built on generalizing and extending the previous sieve algorithms. G6K-GPU-Tensor [28] as a state-of-art SVP solver improves the efficiency of G6K by GPU implementations and holds many records in TU Darmstadt SVP Challenges which is at least 400 times faster than the previous records.

Dimension for Free (d4f) Technique. D4f technology [26] can bring sub-exponential time speedup and memory decrease for sieving algorithms. [26] has given two theoretical d4f estimations for solving β -dimension SVP as $\text{d4f}(\beta) = \beta \ln(4/3) / \ln(\beta/2\pi)$ and $\text{d4f}(\beta) = \beta \ln(4/3) / \ln(\beta/2\pi e)$, while in the implementation of G6K [20], it gives a more relaxed value and we called it "optimistic d4f": $\beta < 40$, $\text{d4f}_{\text{op}}(\beta) = 0$; $40 \leq \beta \leq 75$, $\text{d4f}_{\text{op}}(\beta) = \lfloor \beta - 40/2 \rfloor$; $\beta > 75$, $\text{d4f}_{\text{op}}(\beta) = \lfloor 11.5 + 0.075\beta \rfloor$. However, the d4f estimations in G6K are only related to β , which are sometimes inaccurate. [44] proposed a refined d4f value estimation function based on the quality of current lattice basis, and proved its correctness under GSA, which illustrates that $\text{d4f}_\delta = \ln_\delta \sqrt{4/3} \approx \ln(4/3) / (-\text{slope})$. Here the *slope* is the slope value of the logarithm of Gram-Schmidt norms. More detail about $\text{d4f}_{\text{slope}}(s)$ can see the Eq. (5) in [44].

Pump in G6K. Albrecht et al. proposed a sieving style algorithm Pump in [20], which combines *Progressive Sieve* [27] with d4f technique [26] and a novel insertion trick. There are four input parameters for Pump algorithm: lattice basis \mathbf{B} , left insertion bound κ , insertion upper bound d_{svp} and d4f value $\text{d4f}(d_{\text{svp}})$. Here $\kappa + d_{\text{svp}} = d$ and the upper bound of sieve dimension is $d_{\text{svp}} - \text{d4f}(d_{\text{svp}})$. It's worth emphasizing that the Pump will insert up to $d_{\text{svp}} - \text{d4f}(d_{\text{svp}})$ short vectors into the basis index from κ to d by the short vector that has the shortest norm in the short vector set obtained by each sieve on the projected sublattice $\mathcal{L}_{\pi[d-d_{\text{svp}}+i,d]}$ for i from κ to d . Thus, it performs a partial HKZ reduction and outputs a nearly HKZ-reduced context as the paper of G6K [20] mentioned.

PnJBKZ in G6K. PnJBKZ (Pump and Jump BKZ) is a BKZ-type reduction algorithm that uses Pump as its SVP call. By calling Pump instead of earlier SVP oracles, PnJBKZ can insert more than one vector into the lattice basis when processing each block, thus PnJBKZ can perform lattice reduction with an adjustable jump no less than 1. Specifically, running a PnJBKZ with blocksize β and $\text{jump} = J$ means that after executing SVP call on a certain $\mathbf{B}_{[i,i+\beta]}$, the next SVP call will be executed on the $\mathbf{B}_{[i+J,i+\beta+J]}$ rather than $\mathbf{B}_{[i+1,i+\beta+1]}$.

2.5 Primal Attack and SVP Dimension Estimation in Search Step

In the search step of the two-step method for solving LWE, an individual SVP oracle is called after lattice reduction to find the target solution for LWE. Meanwhile, given a lattice basis \mathbf{B} that has been reduced by using lattice reduction algorithm, one can estimate the dimension of the SVP needed to be solved in the search step to ultimately solve LWE.

For a standard LWE instance $(\mathbf{A}, \mathbf{b}) \in \mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^m$ which error vector \mathbf{e} has standard deviation σ , we suppose that each element in \mathbf{e} follows a discrete Gaussian Distribution $\mathcal{D}_{0,\sigma}$ and $m > n$. The *primal attack* [45] transforms LWE into a uSVP by constructing a special embedding lattice basis \mathbf{B} using Kannan's embedding technique [11], where $\mathbf{B} = \begin{pmatrix} \bar{\mathbf{A}} & \mathbf{b} \\ \mathbf{0} & 1 \end{pmatrix}$. Here $\bar{\mathbf{A}}$ is a basis for the q -ary lattice spanned by the columns of \mathbf{A} . Then $\mathbf{t} = (\mathbf{e}, \pm 1)$ is a unique shortest vector in \mathbf{B} and is also the target vector of uSVP. By considering the norm of target vector \mathbf{t} of LWE by its expected value, one can find minimum $d_{\text{svp}} \in [1, d]$ s.t $\sigma \sqrt{d_{\text{svp}}} < \text{GH}(\mathbf{B}_{\pi[d-d_{\text{svp}},d]})$. According to the success conditions for solving LWE [18], as verified by [10], the minimum dimension of SVP that needs to be solved in search step to solve LWE is d_{svp} . Most efficient LWE solvers [20, 28] use this estimation in their implements.

In this paper, we will demonstrate that this estimation is not accurate and will provide a more precise estimation in Section 4.3. We use $\text{SVPDimEst}(\text{rr}, \sigma)$ to denote the dimension estimation method for estimating the dimension of SVP needed to be solved in search step to solve LWE by inputting the Gram-Schmidt norms rr of its lattice basis \mathbf{B} generated by the primal attack.

2.6 SVP Time Cost Model and Reduction Time Cost Model

Accurate time cost models for sieving and enumeration algorithms are crucial in our strategy search algorithm. If we use enumeration as the SVP solving algorithm and sub-algorithm in lattice reduction algorithm, we can apply the enumeration time cost model proposed in [46, 47]. However, if we use sieving as the SVP solving algorithm and sub-algorithm in the lattice reduction process, the time cost model becomes more complicated. In this paper, we mainly focus on the time cost of the progressive sieve algorithm Pump, as it is the most efficient in practice for solving SVP at current.

Let $T_{\text{Pump}}(\beta)$ be the time cost of a β -dimensional progressive sieve Pump (with pump-down active), we can add the costs of all sieving subroutine to get the theoretical time cost $T_{\text{Pump}}(\beta) = 2 \sum_{j=\beta_0}^{\beta} T_{\text{sieve}}(j) \leq 2^{c \cdot (\beta+1) + o(\beta-\beta_0+1)} / (1-2^c) \approx 2^{c \cdot \beta + c_1}$, where β_0 is the dimension of the initial sieving in Pump (β_0 set as 30 in G6K, 50 in G6K-GPU-Tensor), $T_{\text{sieve}}(j)$ is the cost of a j -dimensional sieve, c and c_1 are the coefficients of time cost related to the sieve algorithm.

For practical time cost, we noticed that Pump in G6K (or G6K-GPU-Tensor) implementation requires extra time cost as $O(\beta)$ times of memory cost to generate the SimHash value, which is used to find the nearest neighbor of each vector. Thus, Pump with theoretical $O(2^{c\beta})$ -time cost and $O(2^{c_2\beta})$ -space cost actually requires $O(2^{c\beta} + \beta \cdot 2^{c_2\beta})$ time. Set $c = 0.367$ and $c_2 = 0.2075$ according to Fig. 7 in [28], we can construct a practical Pump time cost model as $T_{\text{Pump}}(\beta) = a_1 \cdot 2^{c\beta + c_1} + a_2 \cdot 2^{c_2\beta + c_3}$ and obtain its coefficients (as Fig. 26 shown) through the curve fitting method. More details for our practical Pump cost model are shown in the Appendix G.

The time complexity model for lattice basis reduction algorithms like BKZ is represented as $T_{\text{BKZ}}(\beta) = (d-\beta)T_{\text{SVP}}(\beta)$; thus, the time complexity for PnJBKZ can be expressed as $T_{\text{BKZ}}(\beta, J) = (d-\beta)/J \cdot T_{\text{Pump}}(\beta)$. However, experiments have shown that this differs significantly from the actual time complexity model (in practice, the time cost of the Pump increases with the index). Consequently, we provide a practical PnJBKZ time complexity model in Appendix G.

3 Strategy Space and the Strategy Search Algorithm

3.1 Strategy Space

Before introducing the strategy space, we should first clarify the definition of the lattice reduction algorithm. Almost all the lattice reduction algorithms are designed as the combinations of several SVP calls and aim to improve the basis quality by replacing each lattice vector \mathbf{b}_i with a shorter vector, for $i \in [d]$. Thus, we give a general definition of the lattice reduction algorithm as follows,

Definition 4 (Generic Reduction ($\mathcal{R}(\mathbf{B}, \xi)$ or $\mathcal{R}-\xi$)). Reduce the lattice basis \mathbf{B} with a blocksize list $\xi = \llbracket (\kappa_0, \beta_0), \dots, (\kappa_{n-1}, \beta_{n-1}) \rrbracket$, where $\kappa_i, n \in [d-1]$, and $\beta_i \in [1, d - \kappa_i + 1]$. For $i \in [n]$, let $\beta' = \min\{\beta_i, d - \kappa_i\}$, then iteratively call the following two operations:

1. Find the shortest vector \mathbf{v} by running an β' -dimensional SVP call on the projected sublattice basis $\mathbf{B}_{[\kappa_i, \kappa_i + \beta'_i]}$.
2. Insert \mathbf{v} into the position κ_i of the lattice \mathbf{B} by calling an LLL reduction on $(\mathbf{v}, \mathbf{b}_{\kappa_i}, \dots, \mathbf{b}_{\kappa_i + \beta'_i - 1})$ and removing the zero vector after the LLL.

We list the initialization \mathcal{R} - ξ by most commonly used lattice reduction algorithms in practice in Table 1. Each specific reduction algorithm corresponds to specific parameters and can be described by defining the form of ξ .

Table 1: The form of ξ in Generic Reduction initialization

Reduction Algorithm	Reduction Parameter	Each κ_i and β_i in $\xi = \llbracket (\kappa_0, \beta_0), \dots, (\kappa_{n-1}, \beta_{n-1}) \rrbracket$
BKZ [16]	β	$n = d - 2, \kappa_i = i, \beta_i = \min\{\beta, d - i\}, i \in [d - 1]$
PnJBKZ [20]	(β, J)	$n = d - 2, \kappa_i = i, \beta_i = \min\{\beta - (i \bmod J), d - i\}, i \in [d - 1]$
HKZ [11, 48]	-	$n = d - 2, \kappa_i = i, \beta_i = d - i, i \in [d - 1]$

Lattice Basis Quality. There are many different ways to describe the quality of a lattice basis. In the context of BKZ reduction, as GSA is satisfied, one may use the slope of $\ln(\|\mathbf{b}_0^*\|), \dots, \ln(\|\mathbf{b}_{d-1}^*\|)$ to define the quality of a BKZ reduced basis. However, if we consider the lattice basis quality after a tour of PnJBKZ or other reduction algorithms, we can no longer assume GSA. Here we give a definition of a more generic version of lattice basis quality, which can cover the generic reduction in definition 4.

Definition 5 (Generic Basis Quality). For a lattice basis \mathbf{B} , let $(|\mathbf{B}_{[0,1]}| = \|\mathbf{b}_0^*\|, |\mathbf{B}_{[0,2]}| = \|\mathbf{b}_0^*\| \cdot \|\mathbf{b}_1^*\|, \dots, |\mathbf{B}_{[0,d]}| = \prod_{i=0}^{d-1} \|\mathbf{b}_i^*\|)$ be the lattice basis quality of \mathbf{B} .

Moreover, for two different bases \mathbf{C} and \mathbf{D} of a same lattice, we say \mathbf{D} has the same or better lattice basis quality than \mathbf{C} , if $|\mathbf{C}_{[0,k+1]}| \geq |\mathbf{D}_{[0,k+1]}|$ for all $k \in [d]$, written as $\mathbf{D} \geq_{\mathcal{Q}} \mathbf{C}$ or $\text{rr}(\mathbf{D}) \geq_{\mathcal{Q}} \text{rr}(\mathbf{C})$.

If we limit $n = d - 2$ and $\kappa_i = i$ for ξ in the generic reduction \mathcal{R} - ξ , then the Gram-Schmidt norms of the lattice basis after a lattice reduction \mathcal{R} - ξ can be simulated by Gaussian Heuristic, and then it has the following property:

Property 1 (Property of the Generic Reduction) Assume Gaussian Heuristic holds. For a generic reduction tour \mathcal{R} - ξ , where $\xi = \llbracket (0, \beta_0), \dots, (d-2, \beta_{d-2}) \rrbracket$, then the following 2 properties hold:

1. Let $\mathbf{B}' = \mathcal{R}\text{-}\xi(\mathbf{B})$ be the reduced basis after a tour of \mathcal{R} - ξ , $\mathbf{B}' \geq_{\mathcal{Q}} \mathbf{B}$.
2. Given two lattice bases \mathbf{C} and \mathbf{D} , and \mathbf{C}' (resp. \mathbf{D}') is the lattice basis after a tour reduction of \mathcal{R} - ξ on \mathbf{C} (resp. \mathbf{D}). If $\mathbf{D} \geq_{\mathcal{Q}} \mathbf{C}$, then $\mathbf{D}' \geq_{\mathcal{Q}} \mathbf{C}'$.

Now we prove that Property 1 is preserved when we call the same generic reduction \mathcal{R} - ξ on two different lattice bases \mathbf{C} and \mathbf{D} .

Proof. Item 1. (Proof by induction.) Suppose the Gram-Schmidt norms of \mathbf{B} (resp. \mathbf{B}') are (l_0, \dots, l_{d-1}) (resp. (l'_0, \dots, l'_{d-1})). For each i , the Gram-Schmidt norm after \mathcal{R} - ξ can be simulated as $\text{GH}(\mathbf{B}_{\pi[i, i+\beta']})$, where $\beta' = \min\{\beta_i, d-i\}$.

If $k=0$, $|\mathbf{B}'_{\pi[0,1]}| = l'_0 \approx \min\{\text{GH}(\mathbf{B}_{\pi[0,\beta_0]}), l_0\} \leq l_0$ obviously. Assume $|\mathbf{B}'_{\pi[0,n]}| \leq |\mathbf{B}_{\pi[0,n]}|$ holds while $k = n$. When $k = n+1$, under the SMA we have, $l'_n = \min\{l_n, \sqrt{\frac{\beta'}{2\pi e}} \cdot \left(\frac{|\mathbf{B}_{\pi[0,n+\beta']}|}{|\mathbf{B}'_{\pi[0,n]}|}\right)^{1/\beta'}\}$. We depart it into two case: (1) $l'_n = l_n$, then $|\mathbf{B}'_{\pi[0,n+1]}| = |\mathbf{B}'_{\pi[0,n]}| \cdot l_n \leq |\mathbf{B}_{\pi[0,n+1]}|$; (2) $l'_n = \sqrt{\frac{\beta'}{2\pi e}} \cdot \left(\frac{|\mathbf{B}_{\pi[0,n+\beta']}|}{|\mathbf{B}'_{\pi[0,n]}|}\right)^{1/\beta'} \leq l_n$, then

$$\begin{aligned} |\mathbf{B}'_{\pi[0,n+1]}| &= |\mathbf{B}'_{\pi[0,n]}| \cdot \sqrt{\frac{\beta'}{2\pi e}} \cdot \left(\frac{|\mathbf{B}_{\pi[0,n+\beta']}|}{|\mathbf{B}'_{\pi[0,n]}|}\right)^{1/\beta'} = |\mathbf{B}'_{\pi[0,n]}|^{(\beta'-1)/\beta'} \cdot \sqrt{\frac{\beta'}{2\pi e}} |\mathbf{B}_{\pi[0,n+\beta']}|^{1/\beta'} \\ &\leq |\mathbf{B}_{\pi[0,n]}|^{(\beta'-1)/\beta'} \cdot \sqrt{\frac{\beta'}{2\pi e}} |\mathbf{B}_{\pi[0,n+\beta']}|^{1/\beta'} = |\mathbf{B}_{\pi[0,n]}| \cdot \sqrt{\frac{\beta'}{2\pi e}} \cdot \left(\frac{|\mathbf{B}_{\pi[0,n+\beta']}|}{|\mathbf{B}_{\pi[0,n]}|}\right)^{1/\beta'} \\ &= |\mathbf{B}_{\pi[0,n]}| \cdot \sqrt{\frac{\beta'}{2\pi e}} \cdot |\mathbf{B}_{\pi[n,n+\beta']}|^{1/\beta'} \leq |\mathbf{B}_{\pi[0,n+1]}|, \end{aligned} \tag{1}$$

where $\beta' = \min\{\beta_n, d-n\}$. Thus, $\mathbf{B}' \geq_{\mathcal{Q}} \mathbf{B}$.

Item 2. (Proof by induction.) Suppose the Gram-Schmidt norms of \mathbf{C} and \mathbf{D} are $\mathbf{x} = (x_0, \dots, x_{d-1})$ and $\mathbf{y} = (y_0, \dots, y_{d-1})$. $|\mathbf{C}_{\pi[0,k+1]}| \geq |\mathbf{D}_{\pi[0,k+1]}|$ yields $\prod_{i=0}^k x_i \geq \prod_{i=0}^k y_i$, for all $k \in [d]$. Suppose the output Gram-Schmidt norms of lattice basis \mathbf{C} and \mathbf{D} after a tour of Reduction \mathcal{R} - ξ are $\mathbf{x}' = (x'_0, \dots, x'_{d-1})$ and $\mathbf{y}' = (y'_0, \dots, y'_{d-1})$. If $k = 0$,

$$|\mathbf{C}'_{\pi[0,1]}| = x'_0 = \sqrt{\frac{\beta_0}{2\pi e}} \left(\prod_{i=0}^{\beta_0-1} x_i\right)^{\frac{1}{\beta_0}} \geq \sqrt{\frac{\beta_0}{2\pi e}} \left(\prod_{i=0}^{\beta_0-1} y_i\right)^{\frac{1}{\beta_0}} = y'_0 = |\mathbf{D}'_{\pi[0,1]}|$$

Assume $|\mathbf{C}'_{\pi[0,n]}| \geq |\mathbf{D}'_{\pi[0,n]}|$ holds while $k=n$. When $k=n+1$, under the SMA,

$$\begin{aligned} |\mathbf{C}'_{\pi[0,n+1]}| &= |\mathbf{C}'_{\pi[0,n]}| \cdot x'_n = |\mathbf{C}'_{\pi[0,n]}| \cdot \sqrt{\frac{\beta'}{2\pi e}} \left(\frac{|\mathbf{C}_{\pi[0,n+\beta']}|}{|\mathbf{C}'_{\pi[0,n]}|}\right)^{1/\beta'} \\ &= |\mathbf{C}'_{\pi[0,n]}|^{(\beta'-1)/\beta'} \sqrt{\frac{\beta'}{2\pi e}} |\mathbf{C}_{\pi[0,n+\beta']}|^{1/\beta'} \geq |\mathbf{D}'_{\pi[0,n]}|^{(\beta'-1)/\beta'} \sqrt{\frac{\beta'}{2\pi e}} |\mathbf{D}_{\pi[0,n+\beta']}|^{1/\beta'} \\ &= |\mathbf{D}'_{\pi[0,n]}| \cdot \sqrt{\frac{\beta'}{2\pi e}} \left(\frac{|\mathbf{D}_{\pi[0,n+\beta']}|}{|\mathbf{D}'_{\pi[0,n]}|}\right)^{1/\beta'} = |\mathbf{D}'_{\pi[0,n+1]}|, \end{aligned} \tag{2}$$

where $\beta' = \min\{\beta_n, d-n\}$. Thus, $\mathbf{D}' \geq_{\mathcal{Q}} \mathbf{C}'$. \square

Then, we formally define the strategy space in two-step mode for solving LWE.

Definition 6 (Strategy Space). *Given a lattice basis \mathbf{B} of an uSVP instance. Suppose we have a lattice reduction algorithm \mathcal{R} with an adjustable parameter ξ , an SVP Oracle \mathcal{C} , then let $\mathcal{S}(\mathcal{R}, \mathcal{C})$ be the corresponding Strategy Space, each two-step strategy $(\mathbf{S} = \llbracket \xi_0, \dots, \xi_i, \dots \rrbracket, d_{\text{svp}}) \in \mathcal{S}$ can find the unique shortest vector of the lattice $\mathcal{L}(\mathbf{B})$ by running the following two steps:*

1. *Reduction Step:* Iteratively call the reduction method \mathcal{R} with parameter ξ_i on lattice basis \mathbf{B} using the reduction strategy \mathcal{S} , where each \mathcal{R} - ξ can improve the basis quality. Output a lattice basis \mathbf{B}' after running all the reductions with parameter $\xi \in \mathcal{S}$.
2. *Searching Step:* Find the unique shortest vector by running a d_{sVP} -dimensional SVP call on $\mathbf{B}'_{[d-d_{\text{sVP}}, d]}$ to find the projection of target vector and lift it to the full dimensional lattice.

BKZ is the most common lattice reduction algorithm and PnJBKZ (Sec. 2.4) is the most efficient lattice reduction algorithm used in practice at current. We can choose BKZ or PnJBKZ as the reduction algorithm \mathcal{R} , and lattice enumeration or (progressive) lattice sieve as the SVP Oracle used in the searching step.

Progressive Reduction. In 2020, Li and Nguyen [49] presented the first rigorous dynamic analysis of BKZ and in 2024, Wang [50] presented the first dynamic analysis of PnJBKZ. According to the conclusions of [49–51], the lattice basis reduced by the lattice reduction algorithms like BKZ [16], Slide reduction [37] and PnJBKZ [20] with a fixed reduction parameter, will converge to a specific reduced lattice basis after a polynomial number of reduction iterations. Therefore, for a well-reduced lattice basis, using BKZ or PnJBKZ as the lattice reduction algorithm \mathcal{R} , further improving the basis quality by continuing to apply the lattice reduction algorithm \mathcal{R} with the same or even weaker reduction parameters is futile, and we do not consider this a useful strategy. To achieve better reduction quality, it is necessary to use different reduction parameters that offer a stronger reduction effect, which is usually called a progressive reduction strategy.

A valid reduction strategy to improve the lattice basis quality is finite, making strategy space \mathcal{S} finite, as shown in Theorem 1 and proven in Appendix C.1.

Theorem 1. *If a lattice basis \mathbf{B} reduced by repeatedly calling fixed \mathcal{R} - ξ converges to a fully-reduced basis after a finite number of calls, then \mathcal{S} is a finite set.*

3.2 Strategy Search Algorithm

Before introducing the strategy search algorithm, we should introduce the time cost model for each strategy in solving LWE, as our goal is to find the strategy with the smallest time cost. Given a time cost model $\mathcal{T} = (T_{\mathcal{R}}, T_{\mathcal{C}})$, denote the time cost of entire reduction step as $T_{\mathcal{R}_s}(\mathcal{S}) = \sum_{\xi \in \mathcal{S}} T_{\mathcal{R}}(\xi)$ and the time cost of search step as $T_{\mathcal{C}}(d_{\text{sVP}})$. Then the total time cost of strategy $(\mathcal{S}, d_{\text{sVP}})$ for solving LWE is $T_{\text{total}}(\mathcal{S}, d_{\text{sVP}}) = T_{\mathcal{R}}(\mathcal{S}) + T_{\mathcal{C}}(d_{\text{sVP}})$.

According to Theorem 1, the Strategy Space \mathcal{S} for solving an uSVP instance is finite. There exists a strategy $(\mathcal{S}_{\min}, d_{\text{sVP}}^{(\min)})$ s.t. $T_{\text{total}}(\mathcal{S}_{\min}, d_{\text{sVP}}^{(\min)}) = \min\{T_{\text{total}}(\mathcal{S}, d_{\text{sVP}}) : (\mathcal{S}, d_{\text{sVP}}) \in \mathcal{S}\}$. Then, we formalize the Q7 in Kyber [8] as a combinatorial optimization problem:

Definition 7 (The Problem of Searching a Refined Progressive BKZ Reduction Strategy for Solving LWE). *Given an LWE instance (\mathbf{A}, \mathbf{b}) , transform it into a uSVP instance using a primal attack. Consider an arbitrary*

lattice reduction algorithm \mathcal{R} , an SVP call \mathcal{C} , and their corresponding time cost model \mathcal{T} . The objective is to identify a refined two-step strategy $(\mathbf{S}, d_{\text{svp}}) \in \mathcal{S}(\mathcal{R}, \mathcal{C})$ that minimizes the total time cost for solving the LWE instance.

Let TwoStepSolver (Alg. 1) be a two-step strategy algorithm for solving LWE, which employs lattice reduction algorithm \mathcal{R} and the SVP call \mathcal{C} to solve LWE according to the inputting two-step strategy $(\mathbf{S}, d_{\text{svp}})$. This approach involves first applying a series of reductions, and at a good timing utilizing the SVP call to search for the target vector.

In this paper, we propose an algorithm named *Strategy Search*(SSearch) to identify a strategy $(\mathbf{S}_{\min}, d_{\text{svp}}^{(\min)}) \in \mathcal{S}$ that minimizes time cost for solving an arbitrary LWE instance under the given reduction algorithm \mathcal{R} and SVP call \mathcal{C} .

However, since the time cost of the reduction algorithm is exponential with respect to reduction parameter ξ , it is impractical to find the optimized reduction strategy by actually running all possible strategies. To accurately simulate the reduction effort of each strategy, we introduce the concept of the polynomial-time simulator of \mathcal{R} and a dimension estimation method for the SVP calls, denoted as `ReductionSim` and `SVPDimEst`. `ReductionSim` can be initialized as one of the simulators: the BKZ simulator [17] or the PnJBKZ simulator proposed in Sec.5, depending on the reduction algorithm \mathcal{R} used, such as BKZ or PnJBKZ. PnJBKZ simulator in Sec.5 is the first polynomial-time simulator for PnJBKZ. Given Gram-Schmidt norms rr_0 and a reduction strategy \mathbf{S} , `ReductionSim`(rr_0, \mathbf{S}) predicts how the Gram-Schmidt norms changes after iteratively calling \mathcal{R} - ξ reduction for $\xi \in \mathbf{S}$ with input rr_0 . If \mathbf{S} contains a single reduction strategy ξ , we can also re-write `ReductionSim`(rr_0, \mathbf{S}) as `ReductionSim`(rr_0, ξ).

input : LWE instance $(\mathbf{A}, \mathbf{b}) \in \mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^m$, a lattice reduction algorithm \mathcal{R} ,
an SVP call \mathcal{C} , a two-step strategy $(\mathbf{S}, d_{\text{svp}}) \in \mathcal{S}(\mathcal{R}, \mathcal{C})$;
output: The unique shortest vector \mathbf{t} ;

1 **Function** TwoStepSolver($(\mathbf{A}, \mathbf{b}), \mathcal{R}, \mathcal{C}, (\mathbf{S}, d_{\text{svp}})$):
2 Construct lattice basis \mathbf{B} using primal attack to solve LWE instance
 (\mathbf{A}, \mathbf{b}) ;
3 $\mathbf{B} = \text{LLL}(\mathbf{B})$;
4 **for** $\xi \in \mathbf{S}$ **do**
5 $\mathbf{B} \leftarrow \mathcal{R}(\mathbf{B}, \xi)$;
6 $\pi_{d-d_{\text{svp}}}(\mathbf{t}) \leftarrow$ run an SVP call \mathcal{C} on $\mathbf{B}_{\pi[d-d_{\text{svp}}:d]}$;
7 $\mathbf{t} \in \mathcal{L} \leftarrow$ Lift $\pi_{d-d_{\text{svp}}}(\mathbf{t})$ by Babai's Nearest Plane Algorithm [12];
8 **return** \mathbf{t} ;

Algorithm 1: Two-step Solver

Condition 1 Given a lattice reduction algorithm \mathcal{R} and a SVP call \mathcal{C} , we have

1. A reduction simulator named `ReductionSim` can predict the reduction effort of \mathcal{R} accurately in polynomial time cost;
2. A dimension estimation method denoted as `SVPDimEst` (e.g. Sec. 2.5) can estimate the dimension for the SVP call \mathcal{C} in the search step;
3. The time cost model \mathcal{T} for \mathcal{R} and \mathcal{C} ;

If Condition 1 is satisfied, SSearch can find the minimal time cost strategy $(S_{\min}, d_{\text{svp}}^{(\min)})$ s.t. $T_{\text{total}}(S_{\min}, d_{\text{svp}}^{(\min)}) = \min\{T_{\text{total}}(S, d_{\text{svp}}) : (S, d_{\text{svp}}) \in \mathcal{S}\}$.

Then, we give a simplified version of SSearch as Alg. 2. It aims to (1) enumerate all the possible lattice reduction strategy and estimate the dimension of the SVP call; (2) Compute the total time cost of each LWE solving strategy and output the the lowest time cost solving strategy $(S_{\min}, d_{\text{svp}}^{(\min)})$.

Since \mathcal{S} is finite according to Theorem 1, and the time cost of each strategy with (S, d_{svp}) is a determined pair, if we enumerate all the possible reduction strategy S , we can find a strategy with minimal time cost.

Corollary 1. *If Condition 1 is satisfied, Simplified SSearch outputs a strategy with minimal time cost for solving a given LWE instance.*

```

input : LWE instance  $(\mathbf{A}, \mathbf{b}, \sigma) \in \mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^m \times \mathbb{R}^+$ , a simulator
          ReductionSim for  $\mathcal{R}$ , an SVP dimension estimation method
          SVPDimEst for  $\mathcal{C}$ , a time cost model  $\mathcal{T}$ ;
output:  $(S_{\min}, d_{\text{svp}}^{(\min)})$ ;
1 Function SimplifiedSSearch( $(\mathbf{A}, \mathbf{b}, \sigma)$ , ReductionSim, SVPDimEst,  $\mathcal{T}$ ):
2    $rr_0 \leftarrow$  The Gram-Schmidt norms of a lattice basis  $\mathbf{B}$  from LWE instance
    $(\mathbf{A}, \mathbf{b})$  by primal attack;
3    $(T_{\mathcal{R}}, T_{\mathcal{C}}) \leftarrow \mathcal{T}$ ;  $d_{\text{svp}} \leftarrow$  SVPDimEst( $rr_0, \sigma$ );
4    $T_{\min} \leftarrow T_{\mathcal{C}}(d_{\text{svp}})$ ;  $(S_{\min}, d_{\text{svp}}^{(\min)}) \leftarrow (\llbracket \rrbracket, d_{\text{svp}})$ ;
5   for each possible non-empty reduction strategy  $S$  do
6      $rr \leftarrow$  ReductionSim( $rr_0, S$ );  $d_{\text{svp}} \leftarrow$  SVPDimEst( $rr, \sigma$ );
7      $T_{\text{total}}(S, d_{\text{svp}}) \leftarrow T_{\mathcal{R}_s}(S) + T_{\mathcal{C}}(d_{\text{svp}})$ ;
8     if  $T_{\text{total}} < T_{\min}$  then
9        $(S_{\min}, d_{\text{svp}}^{(\min)}) \leftarrow (S, d_{\text{svp}})$ ;  $T_{\min} \leftarrow T_{\text{total}}$ ;
10  return  $(S_{\min}, d_{\text{svp}}^{(\min)})$ ;

```

Algorithm 2: Simplified SSearch

When TwoStepSolver employs the minimal time-cost two-step strategy $(S_{\min}, d_{\text{svp}}^{(\min)})$ determined by an SSearch algorithm, then we name this algorithm MinTwoStepSolver, we demonstrate its high efficiency in solving LWE in Sec.6.

4 Pruning SSearch

Although the Simplified SSearch algorithm can output the minimal time-cost strategy, the strategy space is vast, making it challenging to find the minimal time-cost strategy quickly. To solve this problem, we propose the Pruning SSearch method (PSSearch, Alg. 3) to decrease the complexity of searching entire strategy space by pruning strategies that will not be or develop into the final solution during the search. In Sec. 4.2, we will give a formal proof for the correctness of our algorithm.

4.1 Main Algorithm

In PSSearch, we search for the strategy with the minimal time cost by traversing the strategy tree in a breadth-first manner. The strategy tree is defined as follows,

Definition 8 (Strategy Tree \mathcal{W}). *A Strategy Tree \mathcal{W} consists of*

1. *The root node of \mathcal{W} is $\text{root} = (\llbracket \rrbracket, d_{\text{svp}}^{(0)})$, where $d_{\text{svp}}^{(0)}$ is calculated by *SVPDimEst*.*
2. *The i -th level node is in the form $(S_i = \llbracket \xi_1, \dots, \xi_i \rrbracket, d_{\text{svp}}^{(i)})$, which consists of a reduction strategy with length i and $d_{\text{svp}}^{(i)} = \text{SVPDimEst}(\text{ReductionSim}(\text{rr}_0, S_i), \sigma)$.*
3. *For any i -th level strategy (S_i, d_{svp}) , its child node is $(S_{i+1} = S_i \cup \llbracket \xi_{i+1} \rrbracket, d_{\text{svp}}^{(i+1)})$, where ξ_{i+1} is any reduction parameter that can further improve the basis quality.*

At the beginning, we have an empty strategy list BS , denote $\#\text{BS}$ as the size of BS . Each element in BS is a pair (S, d_{svp}) . Let rr_0 be the sequence of Gram-Schmidt norms of an lattice basis \mathbf{B} . We first add a strategy $(S_0 = \llbracket \rrbracket, d_{\text{svp}}^{(0)})$ with no reduction, namely the root node of \mathcal{W} , denote as root . $d_{\text{svp}}^{(0)} = \text{SVPDimEst}(\text{rr}_0, \sigma)$. We calculate the time cost of this strategy. $T_{\text{total}}(S_0, d_{\text{svp}}^{(0)}) = T_{\mathcal{R}_s}(S_0) + T_C(d_{\text{svp}}^{(0)}) = 0 + T_C(d_{\text{svp}}^{(0)}) = T_C(d_{\text{svp}}^{(0)})$.

Then, we will repeatedly visit the strategies in BS and determine whether to add the child strategy node into BS according to a specific pruning rule until there are no child nodes left for searching. For instance, we will firstly consider all the possible sub-strategies of length 1, which are the child nodes of root , specifically $(\llbracket \xi \rrbracket, d_{\text{svp}})$ for different ξ . After searching all the child nodes of the nodes in BS , output the minimal time-cost strategy in BS .

Notably, the strategies in the list BS are organized such that the time cost of Reduction step $T_{\mathcal{R}_s}(S)$ increases while the time cost of Searching step $T_C(d_{\text{svp}})$ decreases. The pruning strategy is not related to the total time cost, as there may exists a descendant node with smaller total time cost. Instead, we compare both the time cost and the basis quality after the reduction step between nodes. If a node has higher time cost and worse basis quality than another, it cannot be grown into a strategy with lowest total time cost, thus it can be discarded. In other words, for each new strategy (S', d'_{svp}) waiting to be added, we will compare it with each current strategy in $(S, d_{\text{svp}}) \in \text{BS}$ and apply the following rules to either save it or delete it.

Rule 1 (Pruning strategies in \mathcal{W}) *Pruning strategies in \mathcal{W} consists of*

1. *If it exists a $(S, d_{\text{svp}}) \in \text{BS}$ that has both lower reduction time cost and better basis quality than which of (S', d'_{svp}) , i.e. $T_{\mathcal{R}_s}(S) < T_{\mathcal{R}_s}(S')$ and $\text{ReductionSim}(\text{rr}_0, S) \geq_Q \text{ReductionSim}(\text{rr}_0, S')$, then we will not add (S', d'_{svp}) .*
2. *Otherwise, we will add (S', d'_{svp}) and remove all strategies in BS that have both higher reduction time cost and worse basis quality than (S', d'_{svp}) .*

4.2 Correctness Proof of Pruning SSearch

To ensure that PSSearch (Alg. 3) can still find the strategy with the shortest time cost in the pruned state, we provide Theorem 2 to prove its correctness. Before giving the proof of Theorem 2, we will introduce a necessary lemma to support it. For a given SVPDimEst as description of Sec. 2.5, we can use Property 1 to prove that each lattice basis maintains the property of T_C , i.e. Lemma 1.

Theorem 2. *Given the strategy space \mathcal{S} as Def. 6 and SVPDimEst as described in Sec. 2.5, if Condition 1 is satisfied, the $T_C(x)$ is a monotonically increasing function regarding x and the Gram-Schmidt norms after a reduction \mathcal{R} can be predicted by Gaussian Heuristic, then the algorithm 3, PSSearch returns the reduction strategy in \mathcal{S} with minimum time cost to solve a given LWE instance.*

```

input : LWE instance  $(\mathbf{A}, \mathbf{b}, \sigma) \in \mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^m \times \mathbb{R}^+$ , a simulator
          ReductionSim for  $\mathcal{R}$ , an SVP dimension estimation method for  $\mathcal{C}$ , a
          time cost model  $\mathcal{T}$ ;
output:  $(\mathbf{S}_{\min}, d_{\text{svp}}^{(\min)})$ ;
1 Function PSSearch( $(\mathbf{A}, \mathbf{b}, \sigma)$ , ReductionSim, SVPDimEst,  $\mathcal{T}$ ):
2    $rr_0 \leftarrow$  The Gram-Schmidt norms of a lattice basis  $\mathbf{B}$  from LWE instance
   ( $\mathbf{A}, \mathbf{b}$ ) by primal attack;
3    $(T_{\mathcal{R}}, T_{\mathcal{C}}) \leftarrow \mathcal{T}$ ;  $k \leftarrow 0$ ;  $d_{\text{svp}} \leftarrow$  SVPDimEst( $rr_0, \sigma$ );  $\text{BS} \leftarrow \{(\mathbf{B}, d_{\text{svp}})\}$ ;
4   while  $k < \#\text{BS}$  do
5      $(\mathbf{S}, d_{\text{svp}}) \leftarrow \text{BS}[k]$ ;  $\xi \leftarrow$  the last element of  $\mathbf{S}$ ;
6     for  $\forall \xi'$  s.t. ReductionSim( $rr_0, \mathbf{S} \cup \{\xi'\}$ )  $\geq_{\mathcal{Q}}$  ReductionSim( $rr_0, \mathbf{S}$ ) do
7        $\mathbf{S}' \leftarrow \mathbf{S} \cup \{\xi'\}$ ;  $rr' \leftarrow$  ReductionSim( $rr_0, \mathbf{S}'$ );
8        $d'_{\text{svp}} \leftarrow$  SVPDimEst( $rr', \sigma$ );
9       if  $\exists (\mathbf{S}, d_{\text{svp}}) \in \text{BS}$  s.t.  $T_{\mathcal{R}_s}(\mathbf{S}') \geq T_{\mathcal{R}_s}(\mathbf{S})$  and
        ReductionSim( $rr_0, \mathbf{S}'$ )  $\leq_{\mathcal{Q}}$  ReductionSim( $rr_0, \mathbf{S}$ ) then
10        | continue;
11        else
12        |  $\text{BS} \leftarrow \text{BS} \cup \{(\mathbf{S}', d'_{\text{svp}})\}$ ;
13        | for  $\forall (\mathbf{S}, d_{\text{svp}}) \in \text{BS}$  s.t.  $T_{\mathcal{R}_s}(\mathbf{S}') \leq$ 
          |  $T_{\mathcal{R}_s}(\mathbf{S})$  and ReductionSim( $rr_0, \mathbf{S}'$ )  $\geq_{\mathcal{Q}}$  ReductionSim( $rr_0, \mathbf{S}$ )
          | do
14        | | Remove  $(\mathbf{S}, d_{\text{svp}})$  from BS;
15        |  $k \leftarrow k + 1$ ;
16   Find the strategy  $(\mathbf{S}_{\min}, d_{\text{svp}}^{(\min)}) \in \text{BS}$  s.t.
         $T_{\text{total}}(\mathbf{S}_{\min}, d_{\text{svp}}^{(\min)}) = \min_{(\mathbf{S}_{\min}, d_{\text{svp}}^{(\min)}) \in \text{BS}} T_{\text{total}}(\mathbf{S}, d_{\text{svp}})$ ;
17   return  $(\mathbf{S}_{\min}, d_{\text{svp}}^{(\min)})$ ;

```

Algorithm 3: Pruning SSearch

Lemma 1. *Suppose GH and SMA holds. Given an SVPDimEst described as Sec. 2.5 and two arbitrary basis $\mathbf{D} \neq \mathbf{C}$ for the same d -dimensional lattice generated from an LWE instance with standard deviation σ by primal attack. Assume $\mathbf{D} \geq_{\mathcal{Q}} \mathbf{C}$. Let \mathbf{C}' (resp. \mathbf{D}') be the Gram-Schmidt norms of lattice basis after calling a tour of \mathcal{R} - ξ on \mathbf{C} (resp. \mathbf{D}), then SVPDimEst($rr(\mathbf{C}'), \sigma$) \geq SVPDimEst($rr(\mathbf{D}'), \sigma$).*

The proof of Lemma 1 is similar to those in proving Property 1, so we omit them. For details, see the Appendix C.

Then we can give the proof of Theorem 2 as follows,

Proof. (Proof of Theorem 2.) Let rr_0 be the input Gram-Schmidt norms of a random lattice basis, Suppose that the strategy in \mathcal{S} with minimum time cost is $(S = \llbracket \xi_0, \dots, \xi_{z-1} \rrbracket, d_{\text{svp}})$. We write the sub-strategy $\llbracket \xi_0, \dots, \xi_i \rrbracket$ of S as S_i and let $d_{\text{svp}}^{(i)} := \text{SVPDimEst}(\text{ReductionSim}(rr_0, S_i), \sigma)$ for $0 \leq i \leq z$.

$T_{\mathcal{C}}(d_{\text{svp}}^{(i-1)}) > T_{\mathcal{C}}(d_{\text{svp}}^{(i)})$ for all $i \leq z$. Otherwise, removing ξ_i from S_i can get a strategy solving an LWE instance in less time.

From the description of PSSearch, either (1) S is inside the final strategy set BS, or (2) there is a sub-strategy S_i of S having been removed from BS (then S won't appear in BS anymore). Since S has a minimum time cost among all strategies in BS, S must be the final output strategy and meets the first case. Now we show that the second case cannot occur.

If $(S_i, d_{\text{svp}}^{(i)})$ is removed from BS, then there must be another strategy (S', d'_{svp}) such that $\text{ReductionSim}(rr_0, S') \geq_{\mathcal{Q}} \text{ReductionSim}(rr_0, S_i)$, and the reduction time cost $T_{\mathcal{R}_s}(S_i) \geq T_{\mathcal{R}_s}(S')$. Considering a new strategy $S^* := S' \cup \llbracket \xi_{i+1}, \dots, \xi_{z-1} \rrbracket$, it infers that $T_{\mathcal{R}_s}(S) \geq T_{\mathcal{R}_s}(S^*)$. Let $d_{\text{svp}}^* = \text{SVPDimEst}(\text{ReductionSim}(rr_0, S^*), \sigma)$, from Lemma 1 and the increasing property of $T_{\mathcal{C}}$, we can get that $T_{\mathcal{C}}(d_{\text{svp}}) \geq T_{\mathcal{C}}(d_{\text{svp}}^*)$. Since $T_{\mathcal{R}_s}(S_i) \geq T_{\mathcal{R}_s}(S^*)$, it contradicts the expectation that S is the reduction strategy of minimum time-cost strategy. \square

4.3 A Novel Dimension Estimation Method for the SVP Call

We introduce the classical `SVPDimEst` method in Sec. 2.5. However, estimating the projected norm of target vector as $\|\pi_{d-d_{\text{svp}}}(\mathbf{t})\| = \sigma\sqrt{d_{\text{svp}}}$ has a failure probability in solving LWE in last SVP call, as \mathbf{t} follows a certain distribution (usually discrete Gaussian), so there is a possibility that $\|\pi_{d-d_{\text{svp}}}(\mathbf{t})\| > \text{GH}(\mathbf{B}_{\pi[d-d_{\text{svp}}]}) \geq \sigma\sqrt{d_{\text{svp}}}$. To solve this problem, we propose `SVPDimEst` (Alg. 5) for estimating the sieving dimension to solve general LWE with arbitrary target vector distributions. It aims to predict the dimension of last SVP call more accurately and improve the solving success probability. Its detailed description is in Appendix B and it remains the same property stated in Lemma 1 as Lemma 2.

5 The Design of PnJBKZ Simulator

Before we use PSSearch to actually solve an LWE instance, we should first instantiate it with a practical lattice reduction algorithm. PnJBKZ [20] is the most efficient lattice reduction algorithm in practice. However, before we can generate reduction strategies for PnJBKZ using PSSearch, we must first give an accurate simulator for PnJBKZ, especially when $\text{jump} > 1$.

The rigorous dynamic analysis proposed in [49, 50] analyzes the reduction effect, which converges only after running numerous tours of a fixed block-size BKZ- β or PnJBKZ- (β, J) reduction. However, in practice, we require a polynomial-time PnJBKZ- (β, J) simulator to predict the practical reduction effects of PnJBKZ- (β, J) with a more flexible number of tours. This is because the

number of tours in a fixed blocksize BKZ- β reduction is often relatively small during most progressive reduction processes [17, 19, 20, 37].

Meanwhile, the classical BKZ simulator cannot be used to predict PnJBKZ when $\text{jump} > 1$ is employed. Fig. 4 shows that a PnJBKZ reduction strategy with $\text{jump} > 1$ can significantly improve the reduction efficiency. Specifically, the $\text{jump}=9$ progressive reduction strategy is four times faster than the $\text{jump}=1$ strategy in achieving a same slope of -0.019 . To generate optimized strategies for PnJBKZ, choosing the appropriate PnJBKZ reduction parameters (β, J) is crucial. Therefore, developing a polynomial-time accurate PnJBKZ simulator that can reliably predict the reduction effort of PnJBKZ is necessary.

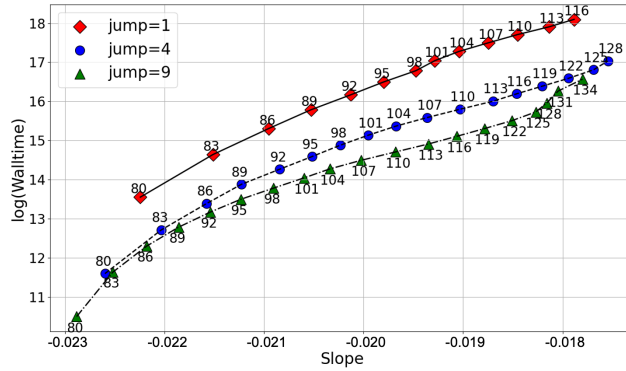


Fig. 4: Efficiency Speedup in Reduction Step by Jump strategy. Test on a 252-dimensional lattice basis. The walltime and slope are averaged over 5 instances for each test. Each instance ran on machine C with 2 GPUs, and 32 threads. The points are labeled by β .

5.1 The PnJBKZ Simulator Construction.

Before presenting the detailed construction of the PnJBKZ simulator, we first briefly review the main idea of the BKZ 2.0 simulator proposed in [17], which uses the Gaussian heuristic to predict BKZ- β .

Let $l'_i = \ln(\|\mathbf{b}_i^*\|)$ and $l''_i = \ln(\|\mathbf{b}_i^{**}\|)$ denote the Gram-Schmidt vectors reduced by one tour of BKZ- β and PnJBKZ(β, J), respectively, for $i \in [d]$. Denote the BKZ simulator proposed in [17] as BKZSim. Our PnJBKZ simulator will be referred to as PnJBKZSim, which simulates the change in lattice basis \mathbf{B} after calling a PnJBKZ algorithm to reduction.

Let $\mathcal{L}(\mathbf{B}^{(i)})$ represent the lattice basis after the first i blocks' reduction and let $\mathcal{L}(\mathbf{B}^{(0)})$ denote the initial lattice basis. For $\forall i \in [d]$, define $l_i = \ln \|\mathbf{b}_i^*\|$ and l'_i be the logarithm of $\|\mathbf{b}_i^*\|$ after one tour of reduction using BKZ- β . The BKZ simulator proposed in [17] first will calculate $\text{Sim}(l'_0) = \ln \left(\text{GH}(\mathcal{L}(\mathbf{B}_{\pi[0,\beta]}^{(0)})) \right) \approx \frac{1}{2} \ln(\beta/2\pi e) + \frac{1}{\beta} \ln \left(\text{Vol}(\mathcal{L}(\mathbf{B}_{\pi[0,\beta]}^{(0)})) \right)$ under GH. It then calculates $\text{Sim}(l'_1) :=$

$$\ln \left(\text{GH}(\mathcal{L}(\mathbf{B}_{\pi[1,\beta+1]}^{(1)})) \right) \approx \frac{1}{2} \ln(\beta/2\pi e) + \frac{1}{\beta} \left(\ln \text{Vol}(\mathcal{L}(\mathbf{B}_{\pi[0,\beta+1]}^{(0)})) - \text{Sim}(l'_0) \right),$$

using GH and the information of $\text{Sim}(l'_0)$. Since the insertion of new \mathbf{b}_0 alters the lengths of the Gram-Schmidt (GS) vectors, i.e. for $\forall i \in \{1, \dots, d-1\}$, l_i changes to some unknown values. In particular, $\text{Vol}(\mathcal{L}(\mathbf{B}_{\pi[0,\beta]}^{(1)})) = \text{Vol}(\mathcal{L}(\mathbf{B}_{\pi[0,\beta]}^{(0)}))$, but l_0 changes to l'_0 after the insert of new \mathbf{b}_0 . So when $\text{Vol}(\mathcal{L}(\mathbf{B}_{\pi[1,\beta+1]}^{(0)})) = \prod_{i=1}^{\beta} \|\mathbf{b}_i^*\|$, after the insert of new \mathbf{b}_0 it will change to $\text{Vol}(\mathcal{L}(\mathbf{B}_{\pi[1,\beta+1]}^{(1)})) = \left(\prod_{i=0}^{\beta} \|\mathbf{b}_i^*\| \right) / \|\mathbf{b}_0^*\|$. Here $\text{Sim}(l'_0)$ is a simulated approximate value of l'_0 by GH. Iteratively calculating all remaining unknown $\text{Sim}(l'_i)$ by $\text{Sim}(l'_i) = \ln \left(\text{GH}(\mathcal{L}(\mathbf{B}_{\pi[i,i+\beta]}^{(i)})) \right) \approx \frac{1}{2} \ln(\beta/2\pi e) + \frac{1}{\beta} \left(\ln \text{Vol}(\mathcal{L}(\mathbf{B}_{\pi[0,i+\beta]}^{(0)})) - \sum_{j=0}^{i-1} \text{Sim}(l'_j) \right)$, for $\forall i \in [d-\beta]$. For $\forall i \in \{d-\beta, \dots, d-1\}$, $\text{Sim}(l'_i)$, it gradually decrease the blocksize to 2. Then such a simulator can predict the value of each l'_i in \mathbf{B}^{*} which is reduced by one tour of BKZ- β .

However, the BKZ 2.0 simulator [17] and its variants cannot be used directly to simulate the behavior of PnJBKZ if $\text{jump} > 1$. Set $\text{jump} = J$. Let $\mathcal{L}(\mathbf{B}^{(i)})$ represent the lattice basis after the first i -th vector was inserted in i -th position of GS vectors and $\mathcal{L}(\mathbf{B}^{(0)})$ be the initial lattice basis. We observe that when $J > 1$, each time a new vector \mathbf{b}_i^* is inserted into the first position of the block $\mathbf{B}_{\pi[i,k]}^{(i)}$, the norms of the $J-1$ GS vectors $\mathbf{b}_{i+1}^*, \dots, \mathbf{b}_{i+J-1}^*$ will change and remain unknown. These unknown norms prevent the BKZ 2.0 simulator from accurately predicting the norm of the first GS vector in the next block. This ultimately results in the norms of subsequent vectors being impossible to predict through iterative calculations. Our approach leverages the properties of HKZ-reduced lattice bases to estimate these unknown norms between adjacent blocks when $J > 1$.

To solve this problem, we first present an ideal version the Pump algorithm—denoted Pump', which satisfies the property that a projected sublattice basis $\mathbf{B}_{\pi[i,i+\beta]}$ after the reduction of Pump'($\mathbf{B}_{\pi[i,i+\beta]}, i, \beta, f$) strictly satisfies the property of an HKZ-reduced basis, for all $i \in [d-\beta]$, dimension of entire lattice basis \mathbf{B} is d . Then we can construct a PnJBKZ simulator for PnJBKZ which uses Pump'. Let l''_i represent the logarithm of each Gram-Schmidt norm after the reduction of one tour of PnJBKZ(β, J). Under GH, we can simulate each l''_i and the simulation values denoted as $\text{Sim}(l''_i) = \ln \left(\text{GH} \left(\mathcal{L}(\mathbf{B}_{\pi[i,k]}^{(i)}) \right) \right)$. Here $i \in [d-\beta]$ and $k = \min\{i - (i \bmod J) + \beta, d\}$.

The key is how to calculate the volume of $\mathcal{L}(\mathbf{B}_{\pi[i,k]}^{(i)})$. Similar to BKZ, during the PnJBKZ reduction process only the lattice basis matrix of the projected sub-lattice has undergone a unimodular transformation, and the volume of the projection sub-lattice has not changed. Specifically, the volume of $\mathcal{L}(\mathbf{B}_{\pi[0,k]}^{(i)})$ equals that of $\mathcal{L}(\mathbf{B}_{\pi[0,k]}^{(0)})$. Suppose we already know $\text{Sim}(l''_j)$, for $\forall j \in [i]$; then we calculate $\ln \left(\mathcal{L}(\mathbf{B}_{\pi[i,k]}^{(i)}) \right) := \ln \left(\text{Vol} \left(\mathcal{L}(\mathbf{B}_{\pi[0,k]}^{(0)}) \right) \right) - \ln \left(\text{Vol} \left(\mathcal{L}(\mathbf{B}_{\pi[0,i]}^{(i)}) \right) \right) = \sum_{j=0}^{k-1} l_j - \sum_{j=0}^{i-1} \text{Sim}(l''_j)$. Here $i \in [d]$, $k = \min\{i - (i \bmod J) + \beta, d\}$. Under Gaussian Heuristic, for $i \in [d]$, we can iteratively calculate $\text{Sim}(l''_i) :=$

$$\frac{1}{2} \ln \left(\frac{k-i}{2\pi e} \right) + \frac{1}{k-i} \left(\sum_{j=0}^{k-1} l_j - \sum_{j=0}^{i-1} \text{Sim}(l''_j) \right), \quad k = \begin{cases} i - (i \bmod J) + \beta, & i \in [d-\beta] \\ d, & i \in [d] \setminus [d-\beta] \end{cases} \quad (3)$$

In other words, we only need to input the initial Gram-Schmidt norms $l_i = \ln(\|\mathbf{b}_i^*\|)$, where $i \in [d]$ of the lattice basis. Without performing the PnJBKZ reduction, we can simulate l_i'' using Eq. (3), which describes the change in lattice basis after each tour of the PnJBKZ(β, \mathcal{J}) reduction. Here l_i'' are these actual

GS vector norms of lattice base after reducing by one tour of PnJBKZ(β, J). We provide a detailed description of the PnJBKZ simulator as Alg. 4⁵.

input : rr , blocksize $\beta \in \{45, \dots, d\}$, jump J and number of tours t .
output: A prediction for the logarithms of the Gram-Schmidt norms $l''_i = \ln(\|\mathbf{b}_i''\|)$ after t tours PnJBKZ- β reduction with jump is J .

```

1 Function PnJBKZSim( $rr, \beta, J, t$ ):
2   for  $i \leftarrow 0$  to 44 do
3      $r_i \leftarrow$  average  $\ln(\|\mathbf{b}_i^*\|)$  of a HKZ reduced random unit-volume
       45-dimensional lattice;
4   for  $i \leftarrow 45$  to  $\beta$  do
5      $c_i \leftarrow \ln(V_i(1)^{-1/i}) = \ln\left(\frac{\Gamma(i/2+1)^{1/i}}{\pi^{1/2}}\right)$ ;
6   for  $j \leftarrow 0$  to  $t-1$  do
7     flag  $\leftarrow$  true; //flag to store whether  $L_{[k,d]}$  has changed
8     for  $k \leftarrow 0$  to  $d-\beta-1$  do
9        $\beta' \leftarrow \min(\beta, d-k)$ ; //Dimension of local block
10       $h \leftarrow \min(k - (k \bmod J) + \beta - 1, d-1)$ ;
11       $\ln(V) \leftarrow \sum_{i=0}^h l_i - \sum_{i=0}^{k-1} l''_i$ ; //Let  $\sum_{i=0}^{-1} l''_i = 0$ 
12      if flag = True then
13        if  $\ln(V) / (\beta' - (k \bmod J)) + c_{\beta' - (k \bmod J)} < l_k$  then
14           $l''_k \leftarrow \ln(V) / (\beta' - (k \bmod J)) + c_{\beta' - (k \bmod J)}$ ;
15          flag  $\leftarrow$  False;
16        else
17           $l''_k \leftarrow \ln(V) / (\beta' - (k \bmod J)) + c_{\beta' - (k \bmod J)}$ ;
18      for  $k \leftarrow d-\beta$  to  $d-46$  do
19         $\beta' \leftarrow d-k$ ;  $h \leftarrow d-1$ ;  $\ln(V) \leftarrow \sum_{i=0}^h l_i - \sum_{i=0}^{k-1} l''_i$ ;
20        if flag = True then
21          if  $\ln(V) / \beta' + c_{\beta'} < l_k$  then
22             $l''_k \leftarrow \ln(V) / \beta' + c_{\beta'}$ ; flag  $\leftarrow$  false;
23          else
24             $l''_k \leftarrow \ln(V) / \beta' + c_{\beta'}$ ;
25         $\ln(V) \leftarrow \sum_{i=0}^h l_i - \sum_{i=0}^{k-1} l''_i$ ;
26        for  $k \leftarrow d-45$  to  $d-1$  do
27           $l''_k \leftarrow \frac{\ln(V)}{45} + r_{k+45-d}$ ;
28        for  $k \leftarrow 0$  to  $d-1$  do
29           $l_k \leftarrow l''_k$ ;
30   return  $(l_0, \dots, l_{d-1})$ ;

```

Algorithm 4: PnJBKZ Simulator

⁵ Besides, we need to remind that in simulating the norm of GS vectors, when $i \equiv 1(\bmod J)$, the index i of $\text{GH}\left(\mathcal{L}\left(\mathbf{B}_{\pi[i, i+\beta-(i \bmod J)]}^{(i)}\right)\right)$ in our simulator is the same as that of $\text{GH}\left(\mathcal{L}\left(\mathbf{B}_{\pi[i, i+\beta]}^{(i)}\right)\right)$ in the BKZ-2.0 simulator, the calculation form looks same in two simulators. However, the simulated volumes of projected sublattice are different in the two simulators. Because in BKZ 2.0 simulator [17] $\text{Vol}(\mathcal{L}'\left(\mathbf{B}_{\pi[i, i+\beta]}^{(i)}\right)) = \prod_{j=0}^{i+\beta-1} \|\mathbf{b}_j^*\| / \prod_{j=0}^{i-1} \|\mathbf{b}_j^{*'}\|$ and it calculates $\|\mathbf{b}_j^{*'}\|$ by $\|\mathbf{b}_j^{*'}\| := \text{GH}\left(\mathcal{L}'\left(\mathbf{B}_{\pi[j, j+\beta]}^{(i)}\right)\right)$, while in PnJBKZ simulator $\text{Vol}(\mathcal{L}''\left(\mathbf{B}_{\pi[i, i+\beta]}^{(i)}\right)) = \prod_{j=0}^{i+\beta-1} \|\mathbf{b}_j^*\| / \prod_{j=0}^{i-1} \|\mathbf{b}_j^{*''}\|$ and $\|\mathbf{b}_j^{*''}\|$ obtained from Eq. (3). So when $i \equiv 1(\bmod J)$ the calculation of $\text{GH}\left(\mathcal{L}\left(\mathbf{B}_{\pi[i, i+\beta]}^{(i)}\right)\right)$ is different in different simulators.

5.2 Upper bound of Jump in PnJBKZ simulator

In this section, we show that it is reasonable for us to use Pump' to simulate the actual Pump if the Jump value in PnJBKZ is below a specific upper bound.

At first, for predicting the reduction effort of PnJBKZ which uses Pump' as its SVP Oracle, we set the upper bound of jump value J to be β . More specifically, the lattice basis $\mathbf{B}_{[i, i+\beta]}$ reduced by a β -dimensional Pump' is a HKZ reduced lattice basis, i.e. $\|\mathbf{b}_i^*\| = \lambda_1(\mathcal{L}_{\pi[i, k]})$, $i < d$, $k = \min\{i + \beta - (i \bmod J), d\}$. When $J \leq \beta$, PnJBKZ simulator use $\text{Sim}(l_i'') = \ln\left(\text{GH}\left(\mathcal{L}(\mathbf{B}_{\pi[i, k]}^{(i)})\right)\right)$ to predict $\lambda_1(\mathcal{L}_{\pi[i, k]})$ and GH ensures PnJBKZ simulator is accurate.

On the other hand, for $J > \beta$, there will always be $J - \beta > 0$ vectors at the beginning of each block whose norms remain unknown, making PnJBKZ simulator unusable in such circumstances. Furthermore, the verification experimental results presented in Appendix D.1 demonstrate the accuracy of the PnJBKZ simulator for predicting the reduction effect of the PnJBKZ, which uses the ideal version of Pump'. The verification of this ideal situation tests that the basic theory behind our construction of the PnJBKZ simulator is correct.

However, in practice, the D4f technique [26] typically used to accelerate sieving in each Pump used in PnJBKZ. A Pump will only perform $(\beta - \text{d4f}(\beta))$ -dimensional progressive sieving during the Pump-up stage, and can thus perform at most $(\beta - \text{d4f}(\beta))$ embeddings during the Pump-down stage. Consequently, the output basis of each β -dimensional Pump will inevitably include $\text{d4f}(\beta)$ -dimensional GS vectors that do not meet the HKZ-reduced basis properties. Thus the upper bound of Jump value should be smaller than $\beta - \text{d4f}(\beta)$.

Finally when predicting the reduction effort of PnJBKZ used in practice, which uses Heuristic optimistic d4f value $\text{d4f}_{\text{op}}(\beta)$ in each Pump, we set the upper bound of jump J to be $\text{d4f}_{\text{slope}}(s) = \ln(4/3)/(-s)$. Since in Appendix D.2, we will demonstrate that the actual d4f is relate to the current basis quality, rather than relying on a fixed Heuristic optimistic d4f value used in the implementation of G6K. More discussions and numerous experiments to verify the accuracy of PnJBKZ simulator in predicting PnJBKZ uses optimistic d4f can be found in Appendix D.2, D.3 and D.4.

6 Experiments and Application to LWE

Since PnJBKZ (resp. Pump) is recognized as the most efficient lattice reduction algorithm (resp. SVP call) currently available, we utilize PnJBKZ (resp. Pump) for TwoStepSolver. This section is dedicated to demonstrating the efficiency of the MinTwoStepSolver in solving LWE instances in practice. We also show its efficiency in solving SVP in Appendix F. Here

$$\text{MinTwoStepSolver} = \text{TwoStepSolver} + \text{PSSearch} + \mathcal{T}$$

is an algorithm that utilizes TwoStepSolver in conjunction with a strategy generated by PSSearch, all under a specific time cost model \mathcal{T} , to effectively solve

LWE instances. Sec. 6.1 presents verification experiments for the PnJBKZ simulator, which serves as a fundamental component of PSSearch. In Sec. 6.2, we apply the MinTwoStepSolver to solve LWE instances and compare it with G6K-GPU-Tensor. Sec. 6.3 introduces the optimized solving strategies generated by PSSearch which are utilized in Sec. 6.2 for solving the TU Darmstadt LWE Challenges. Additionally, we provide a simulation accuracy test in Sec. 6.4. Furthermore, Sec. 6.5 highlights new records achieved in solving the TU Darmstadt LWE Challenges. Finally, Sec. 6.6 offers a refined security estimation of LWE in NIST schemes based on PSSearch.

6.1 Verification experiments of PnJBKZ Simulator

Reduction Prediction for Practical PnJBKZ. To ensure that the PnJBKZ simulator can accurately predict the reduction effort of the PnJBKZ when using the optimistic d4f values, we employ a more refined d4f value estimation proposed in [44] to adjust the optimistic d4f settings during the simulation of PnJBKZ reduction. For more details, see Appendix D.2.

The experimental results in Appendix D.3 show that, for various reduction parameters such as blocksize, jump, and tours, most ratios $l_i''/\text{Sim}(l_i'')$ fall within the range [0.95, 1.05]. Ratios outside this range are still within [0.90, 1.10]. See Fig. 10,11. In comparison, when CN11 simulator simulating classic BKZ, most ratios also fall within [0.95, 1.05], though some ratios at the extremes exceed this range, with the largest ratio reaching 1.15 and the smallest falling below 0.85. This indicates that the prediction accuracy of our PnJBKZ simulator for PnJBKZ reductions using d4f technology is at least as good as that of the classic BKZ simulator [17] for predicting BKZ reductions. For more details on predicting PnJBKZ with d4f technology, see Appendix D.2 and D.3.

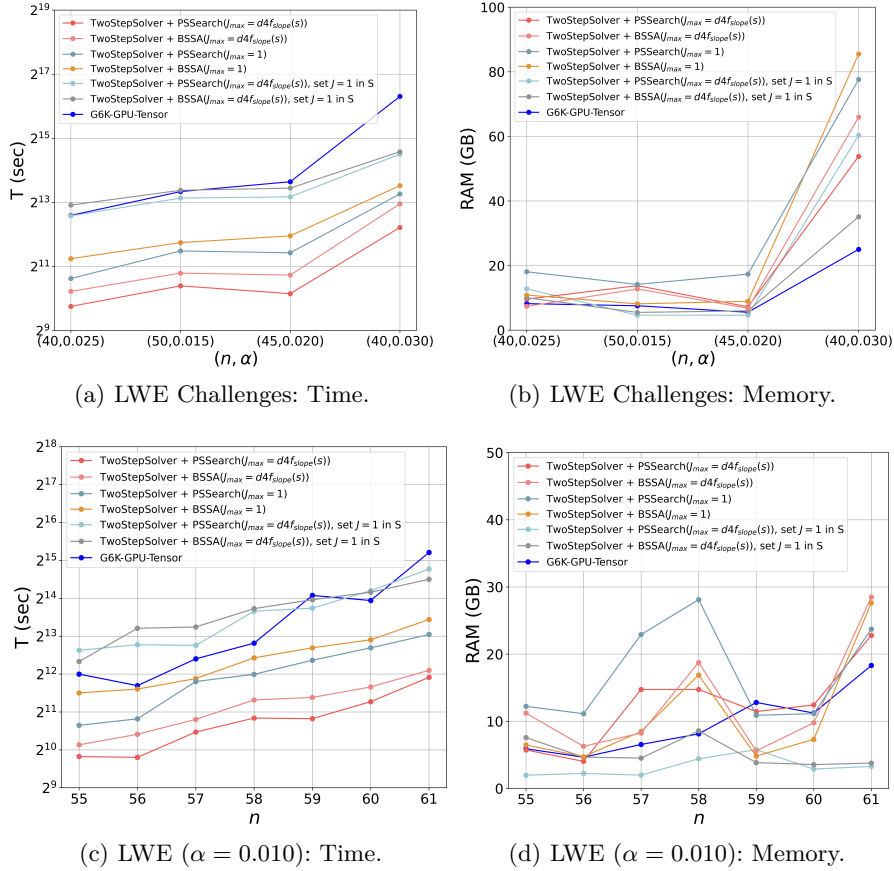
More importantly, the main purpose of constructing the polynomial-time PnJBKZ simulator is to find a more efficient reduction strategy for solving LWE in practice. The key feature is that, given a reduction strategy that uses Jump, the PnJBKZ simulator can accurately predict the basis quality during reduction. Our experiments, detailed in Table 3 and Appendix K, demonstrate that for the LWE challenge lattice basis with various reduction parameters (e.g. different blocksizes and jump sizes), the predicted slope values from PnJBKZ simulator closely match those obtained from actual reductions. Therefore, the PnJBKZ simulator we have constructed is sufficiently effective for its intended purpose. More results of verification experiments with different reduction parameters on various lattice bases can be found in Appendix D.3 and D.4 and link ³.

6.2 Efficiency of MinTwoStepSolver for solving LWE

From the result of Fig. 5 and Table 4, we can see that using the strategy selected by PSSearch (BSSA) significantly decreased the walltime cost by about 7.2~23.4

³ <https://github.com/Summwer/pro-pnj-bkz/tree/merge-enumbs-and-practical-cost-model/simulator-test>

(5.2~10.2) times compared to that of the default LWE solving strategy in G6K when all LWE solvers use the same float type “dd” to calculate. One can refer to the log files of Fig. 5 in the folders `lwechal-test` and `lwe-instance-test`. It can also be reproduced by running the test code `implement_lwechal_forall.sh` and `implement_lwe_instance_forall.sh` in source code². Besides Fig. 6 indicates that in the reduction step, to achieve the same or better lattice basis quality, the optimized reduction strategy found by using the PSSearch can be 4 to 36.7 times faster than the trivial progressive reduction strategy.



§ The experiment used “dd” float type and `pump/down=True`, under identical benchmark (machine C). “default G6K” refers to `lwe_challenge.py` implemented in `g6k`. `TwoStepSolver + PSSearch(·)` (resp. `TwoStepSolver + BSSA(·)`) represents the cost of running `TwoStepSolver` with strategies from `PSSearch` (resp. `BSSA`). J_{\max} denotes the maximum jump value in the strategy. “Set $J=1$ in S ” means generating a strategy S and setting the jump=1.

Fig. 5: Comparison of Different LWE-solving Algorithms under same benchmark. §

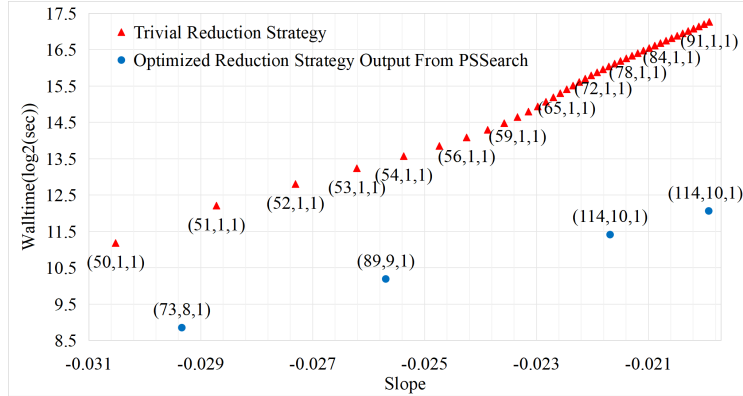


Fig. 6: Speedup in Reduction Step. Test on LWE challenge ($n=60, \alpha = 0.010$) 222-dimensional lattice basis. The points are labeled by (β, J, tours) .

All experimental results, except those in Table 4, were obtained using 32 threads and 2 GPUs on a workstation with an Intel Xeon 5128 (16 cores, 32 threads)@2.3 GHz, 1.48 TB of RAM, and two NVIDIA RTX 3090 GPUs, referred to as machine C. More details about LWE solving efficiency comparison experiments, see Appendix E.1 and Appendix H for more details about BSSA.

6.3 Optimized strategy generated by MinTwoStepSolver

Table 2: Blocksize and Jump strategy generated by PSSearch (threads = 10).

(n, α)	Strategy (β, jump)	SSearchGen/s
(40,0.025)	[(77, 8), (81, 10), (102, 11), (102, 11)]	17.544
(40,0.030)	[(56, 8), (80, 10), (81, 10), (102, 11), (114, 11), (119, 11)]	72.042
(45,0.020)	[(70, 8), (80, 10), (102, 11), (102, 11), (103, 11)]	32.604
(50,0.015)	[(56, 8), (66, 9), (80, 10), (81, 10), (102, 11), (102, 11)]	52.558

We use PSSearch (Alg. 3) with the practical cost model mentioned in Appendix G and tested on machine C to select the blocksize and jump strategy for solving some instances of TU Darmstadt LWE Challenges, we list the selected strategies in Table 2. Besides, Table 2 shows that the time cost of generating the reduction strategy by PSSearch is acceptable. Also, we upload the open source code for blocksize and jump strategy generation on any LWE instances in folder “strategy_gen” from source code². We solved the TU Darmstadt LWE Challenge instances with $(n, \alpha) \in \{(40, 0.035), (40, 0.040), (50, 0.025), (55, 0.020), (90, 0.005)\}$ successfully by the selected strategies in Appendix J.

6.4 Simulated Accuracy of MinTwoStepSolver for LWE

To show the accuracy of reduction estimation and time cost model, we compare the predicted quality of lattice basis and walltime with that of actual experiments

in each middle node in the reduction step. Table 3 and Appendix K illustrate that both the quality of the actual lattice basis and the actual walltime of each tour of PnJBKZ(β, J) are close to our prediction.⁵

(β, J)	Simulation		Practical		(n, α)	Machine	CPU threads	T (h)	RAM (GB)
	Slope	$\log_2(T)$	Slope	$\log_2(T)$					
					(80,0.005)	C	32	2.78	7.3
(56, 8)	-0.0285	6.0	-0.0277	6.2	(40,0.035)	C	32	50.4	326
(80, 10)	-0.0250	6.3	-0.0245	6.5	(40,0.035) ^a	C	32	1180	283
(81, 10)	-0.0232	6.3	-0.0231	6.6	(50,0.025)	A	128	592	184
(102, 11)	-0.0210	7.5	-0.0212	7.8	(55,0.020)	A	128	611	890
(114, 11)	-0.0196	9.1	-0.0198	9.2	(90,0.005)	B	64	370	332
(119, 11)	-0.0190	10.0	-0.0191	10.1	(40,0.040)	A	128	683	1120

Table 3: Quality and $\log_2(T)$ during reduction of LWE $(n, \alpha) = (40, 0.030)$.

Table 4: Actual running time, RAM cost.

^a Use G6K-GPU [28].

6.5 New LWE Records

Based MinTwoStepSolver, we have solved six LWE instances in TU Darmstadt LWE Challenge website¹. See Fig. 2 and Appendix E.2 for more details.

6.6 Security Estimation for NIST schemes

We re-estimate the hardness of LWE-based NIST schemes [52] to estimate the influence of optimized solving strategy found by PSSearch. Under the RAM model, the estimated security bit of LWE in NIST schemes [52] is reduced by 3.4~4.6 bit compared to the estimation generated by Leaky-LWE-Estimator⁶ in [53] under gate-count model which adopts the improved list-decoding technique proposed in [38]. See Table 5. Our new concrete hardness estimation of LWE⁷ answers Q7 in Section 5.3 of Kyber [8] and narrows the security estimation error interval. For more details, please refer to Appendix E.3 and [29].

6.7 The Acceleration Effect of the Optimized Strategy

In practice, leveraging the advantages of multi-threading, c of the practical sieve time cost model with $\beta \leq 124$ is lower than the theoretical value. Since $r = \frac{T_{\text{PnJBKZ}}(\beta, 1)}{T_{\text{PnJBKZ}}(\beta + \Delta\beta, J)} = \frac{J}{1 - \frac{\Delta\beta}{d-\beta} \cdot 2^{c \cdot \Delta\beta}} > 1$ will become larger if c becomes smaller, the strategy generated from PSSearch accelerates the efficiency in solving LWE Challenge significantly. See Figs. 5, 6 and Table 4. In Fig. 7, (1) The black and blue lines show when the LWE hardness parameters (dimension n , error rate α) become sufficiently large, the acceleration effect of multi-threading will gradually

⁵ The data in Table 3 is extracted from a test in Fig. 5 for comparing the quality and walltime between our simulations and actual experiments. For more results please see Appendix K.

⁶ <https://github.com/lucas/leaky-LWE-Estimator>

⁷ <https://github.com/Summwer/lwe-estimator-with-PnJBKZ.git>

diminish; (2) The blue lines also indicates that as α increases, the LWE instance approaches an SVP instance, the acceleration effect of the optimized solving strategy decreases, since most of the time cost is concentrated in sieving of Search Step. Thus, the acceleration effect of the optimized strategy in practice looks weaker while the LWE parameters increase. However, even if the advantages of multi-threading disappear, the optimized strategy still provides acceleration in solving LWE. In an asymptotic sense (theoretical time cost model), there is still an improvement of 3.4 to 4.6 bits for the NIST standard schemes and there are three groups of security parameters (highlighted in Table 5) do not meet the design standards set by NIST. Compared to the trivial two-step solving strategy proposed [29], the security levels decreased by 1.1 to 1.3 bits. See Table 5.

Table 5: Refined Security Estimation results for NIST schemes.[‡]

	$\log_2 G / \log_2(\text{gates})$				$\log_2 B / \log_2(\text{bit})$				$\Delta \log_2 G$
	NIST Required [54]	Previous	Two-step		Previous	Two-step			
			S_0	S_{op}		S_0	S_{op}		
Kyber512	143	146	142.6	141.4	94.0	99.1	98.1	3.4	4.6
Kyber768	207	208.9	205.5	204.3	138.7	144.0	143.2	3.4	4.6
Kyber1024	272	281.1	277.7	276.5	189.8	195.4	194.3	3.3	4.4
Dilithium-II	146	152.9	150.8	149.5	98.0	104.3	103.3	2.1	3.4
Dilithium-III	207	210.2	207.9	206.7	138.8	145.3	144.3	2.3	3.5
Dilithium-V	272	279.2	277.0	275.7	187.52	194.1	193.0	2.2	3.5

[‡] “Previous” is the security estimation in the statement of Kyber and Dilithium. “ $S_0 = \{(\beta_i = i + 2, J_i = 1) \mid i = 1, \dots, \beta\}$ ” is a trivial progressive BKZ+Pump in Two-step mode to estimate security as [29] stated. “ S_{op} ” is a progressive PnJBKZ+Pump with the optimized strategy selected by PSSearch in Two-step mode to estimate security. $\Delta \log_2 G$ is the difference between “Previous” and “Two-step” under the RAM model in strategy S_0 and S_{op} in the logarithm of gate count with base 2. The gate count of all estimations in this Table uses the same improved list-decoding technique proposed by MATZOV [38].

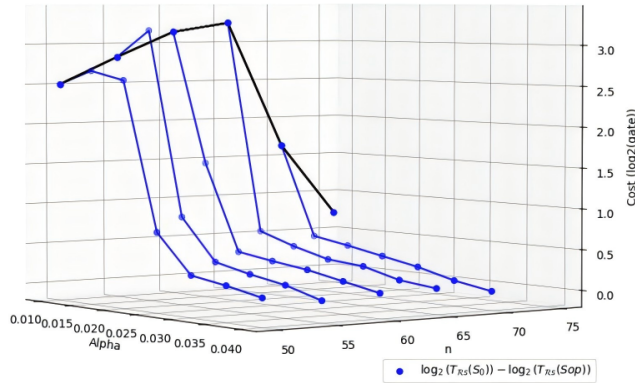


Fig. 7: Acceleration effect of the optimized strategy. LWE challenge lattice basis (n, α) .

References

1. O. Regev, “On lattices, learning with errors, random linear codes, and cryptography,” *Journal of the ACM*, vol. 56, pp. 34:1–34:40, Sept. 2009.
2. L. Ducas, V. Lyubashevsky, and T. Prest, “Efficient identity-based encryption over ntru lattices,” in *Advances in Cryptology – ASIACRYPT 2014* (P. Sarkar and T. Iwata, eds.), (Berlin, Heidelberg), pp. 22–41, Springer Berlin Heidelberg, 2014.
3. X. Boyen, “Attribute-based functional encryption on lattices,” in *Theory of Cryptography* (A. Sahai, ed.), (Berlin, Heidelberg), pp. 122–142, Springer Berlin Heidelberg, 2013.
4. J. M. B. Mera, A. Karmakar, T. Marc, and A. Soleimanian, “Efficient lattice-based inner-product functional encryption,” in *Public-Key Cryptography – PKC 2022* (G. Hanaoka, J. Shikata, and Y. Watanabe, eds.), (Cham), pp. 163–193, Springer International Publishing, 2022.
5. J. H. Cheon, A. Kim, M. Kim, and Y. Song, “Homomorphic encryption for arithmetic of approximate numbers,” in *Advances in Cryptology – ASIACRYPT 2017* (T. Takagi and T. Peyrin, eds.), (Cham), pp. 409–437, Springer International Publishing, 2017.
6. V. Lyubashevsky, C. Peikert, and O. Regev, “On Ideal Lattices and Learning with Errors over Rings,” in *Advances in Cryptology – EUROCRYPT 2010* (H. Gilbert, ed.), (Berlin, Heidelberg), pp. 1–23, Springer, 2010.
7. S. Bai and S. D. Galbraith, “An Improved Compression Technique for Signatures Based on Learning with Errors,” in *Topics in Cryptology – CT-RSA 2014* (J. Benaloh, ed.), (Cham), pp. 28–47, Springer International Publishing, 2014.
8. R. Avanzi, J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, P. Schwabe, G. Seiler, and D. Stehlé, *Kyber*. NIST PQC project, 2021.
9. Bai, Shi, L. Ducas, E. Kiltz, Lepoint, Tancrede, V. Lyubashevsky, P. Schwabe, G. Seiler, and D. Stehlé, *Dilithium*. NIST PQC project, 2021.
10. M. R. Albrecht, F. Göpfert, F. Virdia, and T. Wunderer, “Revisiting the expected cost of solving usvp and applications to lwe,” in *Advances in Cryptology – ASIACRYPT 2017* (T. Takagi and T. Peyrin, eds.), (Cham), pp. 297–322, Springer International Publishing, 2017.
11. R. Kannan, “Improved algorithms for integer programming and related lattice problems,” in *Proceedings of the fifteenth annual ACM symposium on Theory of computing*, STOC ’83, (New York, NY, USA), pp. 193–206, Association for Computing Machinery, Dec. 1983.
12. L. Babai, “On Lovász’ lattice reduction and the nearest lattice point problem,” *Combinatorica*, vol. 6, pp. 1–13, Mar. 1986.
13. M. Liu and P. Q. Nguyen, “Solving BDD by Enumeration: An Update,” in *Topics in Cryptology – CT-RSA 2013* (E. Dawson, ed.), Lecture Notes in Computer Science, (Berlin, Heidelberg), pp. 293–309, Springer, 2013.
14. M. R. Albrecht, F. Göpfert, F. Virdia, and T. Wunderer, “Revisiting the Expected Cost of Solving uSVP and Applications to LWE,” in *Advances in Cryptology – ASIACRYPT 2017* (T. Takagi and T. Peyrin, eds.), vol. 10624, (Cham), pp. 297–322, Springer International Publishing, 2017.
15. M. R. Albrecht, C. Yun, and H. Hunt, “lattice-estimator.” <https://github.com/malb/lattice-estimator>.
16. C. P. Schnorr and M. Euchner, “Lattice basis reduction: Improved practical algorithms and solving subset sum problems,” in *Fundamentals of Computation Theory* (L. Budach, ed.), (Berlin, Heidelberg), pp. 68–85, Springer, 1991.

17. Y. Chen and P. Q. Nguyen, “BKZ 2.0: Better Lattice Security Estimates,” in *Advances in Cryptology – ASIACRYPT 2011* (D. H. Lee and X. Wang, eds.), (Berlin, Heidelberg), pp. 1–20, Springer, 2011.
18. A. Becker, L. Ducas, N. Gama, and T. Laarhoven, “New directions in nearest neighbor searching with applications to lattice sieving,” in *Proceedings of the twenty-seventh annual ACM-SIAM symposium on Discrete algorithms, SODA ’16*, (USA), pp. 10–24, Society for Industrial and Applied Mathematics, Jan. 2016.
19. Y. Aono, Y. Wang, T. Hayashi, and T. Takagi, “Improved Progressive BKZ Algorithms and Their Precise Cost Estimation by Sharp Simulator,” in *Advances in Cryptology – EUROCRYPT 2016* (M. Fischlin and J.-S. Coron, eds.), (Berlin, Heidelberg), pp. 789–819, Springer Berlin Heidelberg, 2016.
20. M. R. Albrecht, L. Ducas, G. Herold, E. Kirshanova, E. W. Postlethwaite, and M. Stevens, “The General Sieve Kernel and New Records in Lattice Reduction,” in *Advances in Cryptology – EUROCRYPT 2019* (Y. Ishai and V. Rijmen, eds.), (Cham), pp. 717–746, Springer International Publishing, 2019.
21. D. Micciancio and P. Voulgaris, “Faster exponential time algorithms for the shortest vector problem,” in *Proceedings of the 2010 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, Proceedings, pp. 1468–1480, Society for Industrial and Applied Mathematics, Jan. 2010.
22. R. Fitzpatrick, C. Bischof, J. Buchmann, Ö. Dagdelen, F. Göpfert, A. Mariano, and B.-Y. Yang, “Tuning gauss sieve for speed,” in *Progress in Cryptology – LATIN-CRYPT 2014* (D. F. Aranha and A. Menezes, eds.), (Cham), pp. 288–305, Springer International Publishing, 2015.
23. G. Herold and E. Kirshanova, “Improved Algorithms for the Approximate k-List Problem in Euclidean Norm,” in *Public-Key Cryptography – PKC 2017* (S. Fehr, ed.), (Berlin, Heidelberg), pp. 16–40, Springer, 2017.
24. G. Herold, E. Kirshanova, and T. Laarhoven, “Speed-Ups and Time–Memory Trade-Offs for Tuple Lattice Sieving,” in *Public-Key Cryptography – PKC 2018*, pp. 407–436, Springer, Cham, Mar. 2018.
25. A. Becker, N. Gama, and A. Joux, “Speeding-up lattice sieving without increasing the memory, using sub-quadratic nearest neighbor search.” <https://eprint.iacr.org/2015/522>, 2015.
26. L. Ducas, “Shortest Vector from Lattice Sieving: A Few Dimensions for Free,” in *Advances in Cryptology – EUROCRYPT 2018* (J. B. Nielsen and V. Rijmen, eds.), (Cham), pp. 125–145, Springer International Publishing, 2018.
27. T. Laarhoven and A. Mariano, “Progressive Lattice Sieving,” in *Post-Quantum Cryptography* (T. Lange and R. Steinwandt, eds.), (Cham), pp. 292–311, Springer International Publishing, 2018.
28. L. Ducas, M. Stevens, and W. van Woerden, “Advanced Lattice Sieving on GPUs, with Tensor Cores,” in *Advances in Cryptology – EUROCRYPT 2021* (A. Canteaut and F.-X. Standaert, eds.), (Cham), pp. 249–279, Springer International Publishing, 2021.
29. W. Xia, L. Wang, G. Wang, D. Gu, and B. Wang, “A Refined Hardness Estimation of LWE in Two-Step Mode,” in *Public-Key Cryptography – PKC 2024* (Q. Tang and V. Teague, eds.), (Cham), pp. 3–35, Springer Nature Switzerland, 2024.
30. A. K. Lenstra, H. W. Lenstra, and L. Lovász, “Factoring polynomials with rational coefficients,” *Mathematische Annalen*, vol. 261, pp. 515–534, Dec. 1982.
31. N. Gama, P. Q. Nguyen, and O. Regev, “Lattice Enumeration Using Extreme Pruning,” in *Advances in Cryptology – EUROCRYPT 2010* (H. Gilbert, ed.), vol. 6110, pp. 257–278, Berlin, Heidelberg: Springer Berlin Heidelberg, 2010.

32. M. R. Albrecht, S. Bai, J. Li, and J. Rowell, “Lattice Reduction with Approximate Enumeration Oracles,” in *Advances in Cryptology – CRYPTO 2021* (T. Malkin and C. Peikert, eds.), (Cham), pp. 732–759, Springer International Publishing, 2021.
33. E. Alkim, L. Ducas, T. Pöppelmann, and P. Schwabe, “Post-quantum key Exchange—A new hope,” in *25th USENIX Security Symposium (USENIX Security 16)*, (Austin, TX), pp. 327–343, USENIX Association, Aug. 2016.
34. P. Q. Nguyen and T. Vidick, “Sieve algorithms for the shortest vector problem are practical,” *Journal of Mathematical Cryptology*, vol. 2, Jan. 2008.
35. Y. Yu and L. Ducas, “Second order statistical behavior of lll and bkz ,” in *Selected Areas in Cryptography – SAC 2017* (C. Adams and J. Camenisch, eds.), (Cham), pp. 3–22, Springer International Publishing, 2018.
36. S. Bai, D. Stehlé, and W. Wen, “Measuring, Simulating and Exploiting the Head Concavity Phenomenon in BKZ,” in *Advances in Cryptology – ASIACRYPT 2018* (T. Peyrin and S. Galbraith, eds.), (Cham), pp. 369–404, Springer International Publishing, 2018.
37. N. Gama and P. Q. Nguyen, “Predicting lattice reduction,” in *Advances in Cryptology – EUROCRYPT 2008* (N. Smart, ed.), (Berlin, Heidelberg), pp. 31–51, Springer Berlin Heidelberg, 2008.
38. MATZOV, “Report on the Security of LWE: Improved Dual Lattice Attack.” <https://doi.org/10.5281/zenodo.6412487>, Apr. 2022.
39. P. Q. Nguyen, “Hermite’s Constant and Lattice Algorithms,” in *The LLL Algorithm* (P. Q. Nguyen and B. Vallée, eds.), pp. 19–69, Berlin, Heidelberg: Springer Berlin Heidelberg, 2009. Series Title: Information Security and Cryptography.
40. Y. Chen, *Réduction de réseau et sécurité concrète du chiffrement complètement homomorphe*. PhD Thesis, 2013.
41. G. Hanrot, X. Pujol, and D. Stehlé, “Analyzing blockwise lattice algorithms using dynamical systems,” in *Advances in Cryptology – CRYPTO 2011* (P. Rogaway, ed.), (Berlin, Heidelberg), pp. 447–464, Springer Berlin Heidelberg, 2011.
42. T. Laarhoven, “Sieving for Shortest Vectors in Lattices Using Angular Locality-Sensitive Hashing,” in *Advances in Cryptology – CRYPTO 2015* (R. Gennaro and M. Robshaw, eds.), (Berlin, Heidelberg), pp. 3–22, Springer, 2015.
43. A. Becker and T. Laarhoven, “Efficient (Ideal) Lattice Sieving Using Cross-Polytope LSH,” in *Progress in Cryptology – AFRICACRYPT 2016* (D. Pointcheval, A. Nitaj, and T. Rachidi, eds.), (Cham), pp. 3–23, Springer International Publishing, 2016.
44. L. Wang, Y. Wang, and B. Wang, “A trade-off svp-solving strategy based on a sharper pnj-bkz simulator,” in *Proceedings of the 2023 ACM Asia Conference on Computer and Communications Security, ASIA CCS ’23*, (New York, NY, USA), p. 664–677, Association for Computing Machinery, 2023.
45. M. R. Albrecht, R. Fitzpatrick, and F. Göpfert, “On the efficacy of solving lwe by reduction to unique-svp,” in *Information Security and Cryptology – ICISC 2013* (H.-S. Lee and D.-G. Han, eds.), (Cham), pp. 293–310, Springer International Publishing, 2014.
46. M. R. Albrecht, S. Bai, P.-A. Fouque, P. Kirchner, D. Stehlé, and W. Wen, “Faster enumeration-based lattice reduction: Root hermite factor $k^{1/(2k)}$ time $k^{k/8+o(k)}$,” in *Advances in Cryptology – CRYPTO 2020* (D. Micciancio and T. Ristenpart, eds.), (Cham), pp. 186–212, Springer International Publishing, 2020.
47. M. R. Albrecht, S. Bai, J. Li, and J. Rowell, “Lattice reduction with approximate enumeration oracles,” in *Advances in Cryptology – CRYPTO 2021* (T. Malkin and C. Peikert, eds.), (Cham), pp. 732–759, Springer International Publishing, 2021.

48. P. Q. Nguyen and B. Valle, *The LLL Algorithm: Survey and Applications*. Springer Publishing Company, Incorporated, 1st ed., 2009.
49. J. Li and P. Q. Nguyen, “A complete analysis of the bkz lattice reduction algorithm.” <https://eprint.iacr.org/2020/1237>, 2020.
50. L. Wang, “Analyzing pump and jump bkz algorithm using dynamical systems,” in *Post-Quantum Cryptography* (M.-J. Saarinen and D. Smith-Tone, eds.), (Cham), pp. 406–432, Springer Nature Switzerland, 2024.
51. J. Li and M. Walter, “Improving convergence and practicality of slide-type reductions,” *Information and Computation*, vol. 291, p. 105012, 2023.
52. I. T. L. C. S. R. CENTER, “Post-quantum cryptography pqc selected algorithms 2022.” <https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022>, 2022.
53. D. Dachman-Soled, L. Ducas, H. Gong, and M. Rossi, “LWE with Side Information: Attacks and Concrete Security Estimation,” in *Advances in Cryptology – CRYPTO 2020: 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17–21, 2020, Proceedings, Part II*, (Berlin, Heidelberg), pp. 329–358, Springer-Verlag, Aug. 2020.
54. “National Institute of Standards and Technology, Submission requirements and evaluation criteria for the post-quantum cryptography standardization process.” <https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/call-for-proposals-final-dec-2016.pdf>, 2016. [Online].
55. A. Leon-Garcia, *Probability, statistics, and random processes for electrical engineering*. Upper Saddle River, NJ: Pearson/Prentice Hall, 3. ed ed., 2008.
56. E. W. Postlethwaite and F. Virdia, “On the Success Probability of Solving Unique SVP via BKZ,” in *Public-Key Cryptography – PKC 2021* (J. A. Garay, ed.), vol. 12710, (Cham), pp. 68–98, Springer International Publishing, 2021.
57. V. Lyubashevsky and D. Micciancio, “On Bounded Distance Decoding, Unique Shortest Vectors, and the Minimum Distance Problem,” in *Advances in Cryptology - CRYPTO 2009* (S. Halevi, ed.), vol. 5677, pp. 577–594, Berlin, Heidelberg: Springer Berlin Heidelberg, 2009.
58. M. R. Albrecht, R. Player, and S. Scott, “On the concrete hardness of Learning with Errors,” *Journal of Mathematical Cryptology*, vol. 9, Jan. 2015.
59. C. Peikert, “A Decade of Lattice Cryptography,” *Found. Trends Theor. Comput. Sci.*, vol. 10, pp. 283–424, Mar. 2016. Place: Hanover, MA, USA Publisher: Now Publishers Inc.
60. K. Xagawa, “Cryptography with Lattices.” <https://xagawa.net/pdf/2010Thesis.pdf>, 2010.
61. M. R. Albrecht, V. Gheorghiu, E. W. Postlethwaite, and J. M. Schanck, “Estimating quantum speedups for lattice sieves,” in *Advances in Cryptology – ASIACRYPT 2020* (S. Moriai and H. Wang, eds.), (Cham), pp. 583–613, Springer International Publishing, 2020.
62. M. R. Albrecht, F. Göpfert, F. Virdia, and T. Wunderer, “Revisiting the Expected Cost of Solving uSVP and Applications to LWE,” in *Advances in Cryptology – ASIACRYPT 2017* (T. Takagi and T. Peyrin, eds.), (Cham), pp. 297–322, Springer International Publishing, 2017.

A Basic Definitions

Definition 9 (The Gaussian Distribution [55]). Let $\sigma, \mu \in \mathbb{R}$ be the standard deviation and the mean value respectively, a continuous Gaussian Distribution denoted as $N(\mu, \sigma^2)$. Its probabilistic density function $\rho_{N(\mu, \sigma^2)} = e^{-\frac{(x-\mu)^2}{2\sigma^2}} / \sigma\sqrt{2\pi}$.

Definition 10 (The discrete Gaussian Distribution [56]). Let $\sigma, \mu \in \mathbb{R}$ be the standard deviation and the mean value respectively, a discrete Gaussian Distribution denoted as $D_{\mu, \sigma}$. If $\mu = 0$, then denote $D_\sigma = D_{0, \sigma}$. Its probability mass function is $f_{D(\mu, \sigma)} : \mathbb{Z} \rightarrow [0, 1], x \rightarrow f_{N(\mu, \sigma)}(x) / f_{N(\mu, \sigma^2)}(\mathbb{Z})$, where $f_{N(\mu, \sigma)}(\mathbb{Z}) = \sum_{x \in \mathbb{Z}} f_{N(\mu, \sigma)}(x)$.

Definition 11 (Chi-Squared Distribution [55]). Given n random variables $X_i \sim N(0, 1)$, the random variables $X_0^2 + \dots + X_{n-1}^2$ follows a chi-squared distribution χ_n^2 over \mathbb{R}^* of mean n and variance $2n$ with probabilistic density function $\rho_{\chi_n^2}(x) = x^{\frac{n}{2}-1} e^{-\frac{x}{2}} / 2^{\frac{n}{2}} \Gamma(n/2)$. Given n random variables $Y_i \sim N(0, \sigma^2)$, the random variables $Y_0^2 + \dots + Y_{n-1}^2$ follows a scaled chi-squared distribution $\sigma^2 \cdot \chi_n^2$ over \mathbb{R}^* of mean $n\sigma^2$ and variance $2n\sigma^2$.

A.1 Lattice Hard Problems

Definition 12 (unique Shortest Vector Problem(uSVP $_\gamma$) [57]). Given an arbitrary basis \mathbf{B} on lattice $\mathcal{L} = \mathcal{L}(\mathbf{B})$, \mathcal{L} satisfies the condition $\gamma\lambda_1(\mathbf{B}) < \lambda_2(\mathbf{B})$ ($\gamma > 1$, $\lambda_2(\mathbf{B})$ is norm of the second shortest vector which is linearly independent to the shortest vector), find the shortest non-zero vector \mathbf{v} s.t. $\|\mathbf{v}\| = \lambda_1(\mathbf{B})$.

Definition 13. (LWE $_{m,n,q,D_\sigma}$ Distribution [58–60]) Given some samples $m \in \mathbb{Z}$, a secret vector dimension $n \in \mathbb{Z}$, 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 Learning with Errors (LWE) distribution Ψ is constructed by the pair $(\mathbf{A}, \mathbf{b} = \mathbf{A}\mathbf{s} + \mathbf{e}) \in (\mathbb{Z}_q^{m \times n}, \mathbb{Z}_q^m)$ sampled as above.

Definition 14 (Search LWE $_{m,n,q,D_\sigma}$ problem [58–60]). Given a pair (\mathbf{A}, \mathbf{b}) sampled from LWE $_{m,n,q,D_\sigma}$ distribution Ψ compute the pair (\mathbf{s}, \mathbf{e}) .

B Detailed Description of our Novel SVPDimEst

The noise vector $\mathbf{e} \in \mathbb{Z}_q^m$ of LWE follow a discrete Gaussian distribution $D_{0, \sigma}$ with standard deviation σ . Then, the probability distribution of squared norm of target vector in β -dimensional sub-lattice can be described as $\sigma^2 \cdot \chi_\beta^2$.

The idea, which considers the norm of the projected target vector on the sub-lattice as a random variable rather than an expected value, was first proposed in [53] for estimating the blocksize of each BKZ. We modify it to SVP

call dimension by replacing the simulating Gram-Schmidt norms of BKZ with Gaussian Heuristic value of the projected sub-lattice.

```

input :  $rr, \sigma$ ;
output:  $d_{\text{svp}}$ ;
1 Function  $\text{SVPDimEst}(rr, \sigma)$ :
2   for  $d_{\text{svp}} \leftarrow d_{\text{start}}$  to  $d$  do
3      $P_{\text{suc}}(d_{\text{svp}}) \leftarrow \Pr \left[ x \leftarrow \sigma^2 \cdot \chi_{d_{\text{svp}}}^2 : x \leq (\text{GH}(rr_{[d-d_{\text{svp}}:d]}))^2 \right]$ ;
4     if  $P_{\text{suc}}(d_{\text{svp}}) \geq 0.999$  then
5       return  $d_{\text{svp}}$ ;

```

Algorithm 5: Dimension Estimation for the SVP call on solving LWE.

In Alg. 5, set $T_{\mathcal{C}}(d_{\text{svp}})$ as the estimated expected cost of the final SVP call. Considering the progressive SVP call, we calculate failure and success probabilities. The success probability of a β -dimensional progressive SVP call denoted as $P_{\text{suc}}(\beta)$ computed by line 3 in Alg. 5. The event E_{β} means finding the target vector precisely at β -dimensional during a progressive SVP call with success probability $\Pr(E_{\beta})=P_{\text{suc}}(\beta) - P_{\text{suc}}(\beta - 1)$ and let $\Pr(E_{\beta_0-1})=0$. The expected time cost of E_{β} is $\sum_{i=\beta_0}^{\beta} T_{\text{SVP}}(i) \cdot \Pr(E_{\beta})$. Iterating β from β_0 to d_{svp} , then

$$T_{\mathcal{C}}(d_{\text{svp}}) = \sum_{\beta=\beta_0}^{d_{\text{svp}}} \left[\sum_{i=\beta_0}^{\beta} T_{\text{SVP}}(i) \cdot (P_{\text{suc}}(\beta) - P_{\text{suc}}(\beta - 1)) \right] = \sum_{\beta=\beta_0}^{d_{\text{svp}}} T_{\text{SVP}}(\beta) \cdot P_{\text{suc}}(\beta), \quad (4)$$

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

Fig. 8 shows that even if $\sigma \sqrt{d_{\text{svp}}} \leq \text{GH}(\mathbf{B}_{\pi[d-d_{\text{svp}}]}), \|\pi_{d-d_{\text{svp}}}(\mathbf{t})\|$ is possibly larger than $\text{GH}(\mathbf{B}_{\pi[d-d_{\text{svp}}]})$, i.e. estimating the upper bound of Pump by the expected value is over-optimistic. The red line shows that the norm of projected vector of our estimated dimension value satisfies the condition $\|\pi_{d-d_{\text{svp}}}(\mathbf{t})\| \leq \text{GH}(\mathbf{B}_{\pi[d-d_{\text{svp}}]})$ by testing 100 trials of LWE instances. Because $\|\mathbf{e}\|^2$ is a randomly positive variable following chi-squared distribution rather than a fixed value. It is more reasonable to consider a high success probability (≥ 0.999) for recovering the target vector with a new estimated dimension in Alg. 5 to solve LWE problem.

Lemma 2 illustrates that considering the chi-squared distribution to estimate the dimension of SVP (SVPDimEst) needed for recovering the target vector of LWE, a better lattice basis quality allows for recovering the target vector of LWE by solving a smaller dimension for the SVP call.

Lemma 2. *Suppose GH and SMA holds. Given an SVPDimEst described as Alg. 5 and two arbitrary basis $\mathbf{C} \neq \mathbf{D}$ for the same d -dimensional lattice generated from an LWE instance with standard deviation σ by primal attack. Assume $\mathbf{D} \geq_{\mathcal{Q}} \mathbf{C}$ and $T_{\text{SVP}}(d_{\text{svp}})$ is a monotonically increasing function regarding d_{svp} . Let \mathbf{C}' (resp. \mathbf{D}') be the lattice basis after calling a tour of $\mathcal{R}\text{-}\xi$ on \mathbf{C} (resp. \mathbf{D}), then $T_{\mathcal{C}}(\text{SVPDimEst}(rr(\mathbf{C}'), \sigma)) \geq T_{\mathcal{C}}(\text{SVPDimEst}(rr(\mathbf{D}'), \sigma))$.*

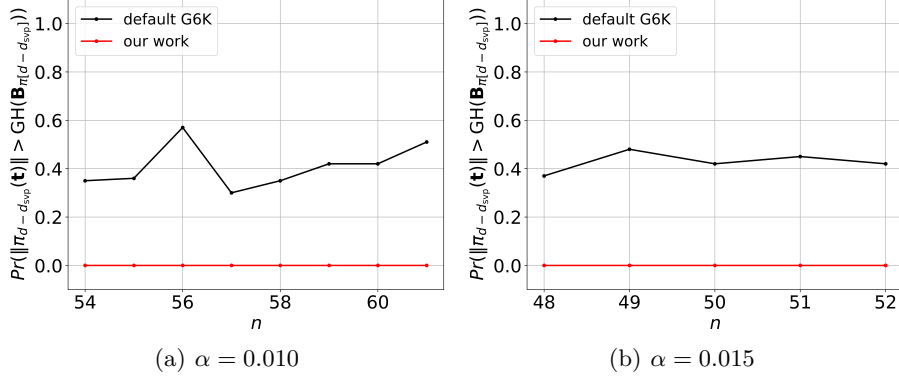


Fig. 8: The failure probability of the estimated dimension for last SVP call. For each (n, α) , 100 randomly LWE instances are generated. The black line shows `SVPDimEst` introduced in Sec. 2.5 has a non-negligible probability s.t. $\|\pi_{d-d_{\text{SVP}}}(\mathbf{t})\| > \text{GH}(\mathbf{B}_{\pi[d-d_{\text{SVP}},d]}) \geq \sigma \sqrt{d_{\text{SVP}}}$. The red line shows that using the estimated dimension computed by Alg. 5 in our work, the condition $\|\pi_{d-d_{\text{SVP}}}(\mathbf{t})\| \leq \text{GH}(\mathbf{B}_{\pi[d-d_{\text{SVP}},d]})$ can always be satisfied.

Proof. From Property 1 and the condition $|\mathbf{C}_{\pi[0,k]}| \geq |\mathbf{D}_{\pi[0,k]}|$, it infers that $|\mathbf{C}'_{\pi[0,k]}| \geq |\mathbf{D}'_{\pi[0,k]}|$, $\forall k \in [1, d]$. Since $|\mathbf{C}| = |\mathbf{D}|$, it yields that $\text{GH}(\mathbf{C}'_{\pi[k,d]}) \leq \text{GH}(\mathbf{D}'_{\pi[k,d]})$, $\forall k \in [d]$.

Considering that the norm of the LWE target vector projection follows a β -dimensional chi-squared distribution with deviation σ as Alg. 5 shown and from the definition of chi-squared distribution, $P(Y) := \Pr[x \leftarrow \sigma^2 \cdot \chi_{d_{\text{SVP}}}^2 : x \leq Y]$ is an increasing function relying on Y . So it infers that $\forall \beta \in [1, d]$, $P(\text{GH}(\mathbf{C}'_{[d-\beta,d]})^2) \leq P(\text{GH}(\mathbf{D}'_{[d-\beta,d]})^2)$ and there is a minimum d_{SVP} s.t. $P(\text{GH}(\mathbf{C}'_{[d-d_{\text{SVP}},d]})^2) \leq 0.999 \leq P(\text{GH}(\mathbf{D}'_{[d-d_{\text{SVP}},d]})^2)$. Since $T_{\text{SVP}}(d_{\text{SVP}})$ is monotonically increasing regarding d_{SVP} and from Eq. (4), $T_{\mathcal{C}}(\text{SVPDimEst}(rr(\mathbf{C}), \sigma)) \geq T_{\mathcal{C}}(\text{SVPDimEst}(rr(\mathbf{D}), \sigma))$. \square

We can still prove the correctness of `PPSearch` by an adaptive Theorem 2 (denoted as Theorem 3) by replacing the `SVPDimEst` method in Sec. 2.5 with Alg. 5 and prove the Theorem 3 by replacing the Lemma 1 with Lemma 2.

Theorem 3. *Given the strategy space \mathcal{S} as Def. 6 defined and `SVPDimEst` as Alg. 5, if Condition 1 is satisfied and $T_{\text{SVP}}(d_{\text{SVP}})$ is a monotonically increasing function regarding d_{SVP} , then Alg. 3, `PSSearch` returns the reduction strategy in \mathcal{S} with minimum time cost to solve LWE instance.*

C Proofs of Theorems and Lemmas

C.1 Finite Strategy Space

Theorem 1. *If a lattice basis \mathbf{B} reduced by repeatedly calling \mathcal{R} with a fixed parameter ξ converges to a fully-reduced basis after a finite number of calls, then \mathcal{S} is a finite set.*

Proof. Repeatedly calling the same reduction \mathcal{R} with parameter ξ on a lattice basis \mathbf{B} yields a converge after a finite number of calls. We call such converged lattice basis ξ -reduced basis. If the ξ -reduced basis can output the target vector \mathbf{v} , then $d_{\text{svp}} = 0$; Otherwise, it exists a minimum value d_{svp} with $d_{\text{svp}} \in [1, d + 1]$ such that the target vector \mathbf{v} can be found under the ξ -reduced basis. Because if $d_{\text{svp}} = d$, then \mathbf{v} must be found through a d -dimensional SVP call. Since the selection range of each ξ is finite, then each \mathcal{S} is finite. Thus, \mathcal{S} is finite. \square

C.2 Order Preservation for SVPDimEst

Lemma 1. *Suppose Gaussian Heuristic and Sandpile Model Assumption holds. Given an SVPDimEst described as Sec. 2.5 and two arbitrary basis $\mathbf{D} \neq \mathbf{C}$ for the same d -dimensional lattice generated from an LWE instance with standard deviation σ by primal attack. Assume $\mathbf{D} \geq_{\mathcal{Q}} \mathbf{C}$. Let \mathbf{C}' (resp. \mathbf{D}') be the Gram-Schmidt norms of lattice basis after calling a tour of \mathcal{R} - ξ on \mathbf{C} (resp. \mathbf{D}), then $\text{SVPDimEst}(\text{rr}(\mathbf{C}'), \sigma) \geq \text{SVPDimEst}(\text{rr}(\mathbf{D}'), \sigma)$.*

Proof. From Property 1 and the condition $\mathbf{D} \geq_{\mathcal{Q}} \mathbf{C}$, it infers that $\mathbf{D}' \geq_{\mathcal{Q}} \mathbf{C}'$. Since $|\mathbf{C}| = |\mathbf{D}|$, it yields that $\text{GH}(\mathbf{C}'_{\pi[k,d]}) \leq \text{GH}(\mathbf{D}'_{\pi[k,d]})$, $\forall k \in [d]$. From the description of SVPDimEst in Sec. 2.5, $d_{\text{svp}} := \text{SVPDimEst}(\mathbf{D})$ is the minimal value s.t. $\sigma\sqrt{d_{\text{svp}}} \leq \text{GH}(\mathbf{D}_{\pi[d-d_{\text{svp}},d]})$. Then there are only the following two case occurring, (1) $\text{GH}(\mathbf{C}'_{\pi[d-d_{\text{svp}},d]}) < \sigma\sqrt{d_{\text{svp}}} \leq \text{GH}(\mathbf{D}'_{\pi[d-d_{\text{svp}},d]})$, we have $\text{SVPDimEst}(\text{rr}(\mathbf{C}'), \sigma) > \text{SVPDimEst}(\text{rr}(\mathbf{D}'), \sigma)$; (2) $\sigma\sqrt{d_{\text{svp}}} \leq \text{GH}(\mathbf{C}'_{\pi[d-d_{\text{svp}},d]}) \leq \text{GH}(\mathbf{D}'_{\pi[d-d_{\text{svp}},d]})$, we have $\text{SVPDimEst}(\text{rr}(\mathbf{C}'), \sigma) = \text{SVPDimEst}(\text{rr}(\mathbf{D}'), \sigma)$. Thus we get $\text{SVPDimEst}(\text{rr}(\mathbf{C}'), \sigma) \geq \text{SVPDimEst}(\text{rr}(\mathbf{D}'), \sigma)$. \square

D PnJBKZ Simulator Accuracy Verification Experiments

D.1 Ideal version PnJBKZ simulation

In an ideal scenario for PnJBKZ, each Pump invoked by the simulator is capable of outputting an HKZ-reduced basis. Verifying this ideal case tests whether the fundamental theory behind our construction of the PnJBKZ simulator is correct. Specifically, we demonstrate that when the jump parameter J of PnJBKZ is smaller than the blocksize β , the PnJBKZ simulator, constructed using the properties of HKZ-reduced bases and Gaussian heuristics, is reasonable and can accurately predict the actual reduction effects of PnJBKZ.

To practically ensure that each Pump outputs an HKZ-reduced basis, it is necessary to forego the d4f technique. Without this adjustment, a Pump will only perform $\beta - \text{d4f}(\beta)$ dimensional progressive sieving during the Pump-up stage, and can thus perform at most $\beta - \text{d4f}(\beta)$ embeddings during the Pump-down stage. Consequently, the output basis of each β -dimensional Pump will inevitably include $\text{d4f}(\beta)$ -dimensional GS vectors that do not meet the HKZ-reduced basis properties. Thus, in this subsection [D.1](#), each β -dimensional Pump used by PnJBKZ performs complete β -dimensional progressive sieving during the Pump-up stage and activates sieving during the Pump-down stage. Specifically, it executes full sievings on the corresponding projection sublattices and embeds the shortest vectors found through re-sieving during the Pump-down stage.

We calculate $\text{Sim}(l_i'')$ based strictly on the properties of the HKZ-reduced basis. Similarly to classical BKZ simulators [\[17, 36\]](#), which assess the accuracy of BKZ simulators, we use the ratio $l_i''/\text{Sim}(l_i'')$ for $i \in [0, d - 1]$ in each tour of the PnJBKZ reduction as a criterion for measuring the accuracy of the PnJBKZ simulator. See [Fig. 9](#), which shows the prediction accuracy of the PnJBKZ simulator under different jump values. Here, l_i'' represents the average logarithm of the Gram-Schmidt vector lengths obtained from 20 independent reduction experiments with the same parameters (β, J, tours) , and $\text{Sim}(l_i'')$ is the simulated logarithm of these Gram-Schmidt vector lengths, calculated using [Eq. \(3\)](#).

When using the classic Chen-Nguyen simulator to predict the actual reduction effect of BKZ, which employs pruned enumeration as the SVP oracle, most ratios fall within the range $[0.95, 1.05]$. Our simulation results are similar to those of the classic Chen-Nguyen simulator, with the remaining ratios falling within the range of $[0.85, 1.15]$. For details, compare [Fig. 1](#) in [Section 4.3](#) of the BKZ 2.0 paper [\[17\]](#) with [Fig. 9](#) in our paper. Therefore, the error in predicting the reduction effect of the ideal PnJBKZ using our PnJBKZ simulator is not larger than that of the classic Chen-Nguyen simulator [\[17\]](#). The overall prediction results of the PnJBKZ simulator are also shown in [Fig. 9](#).

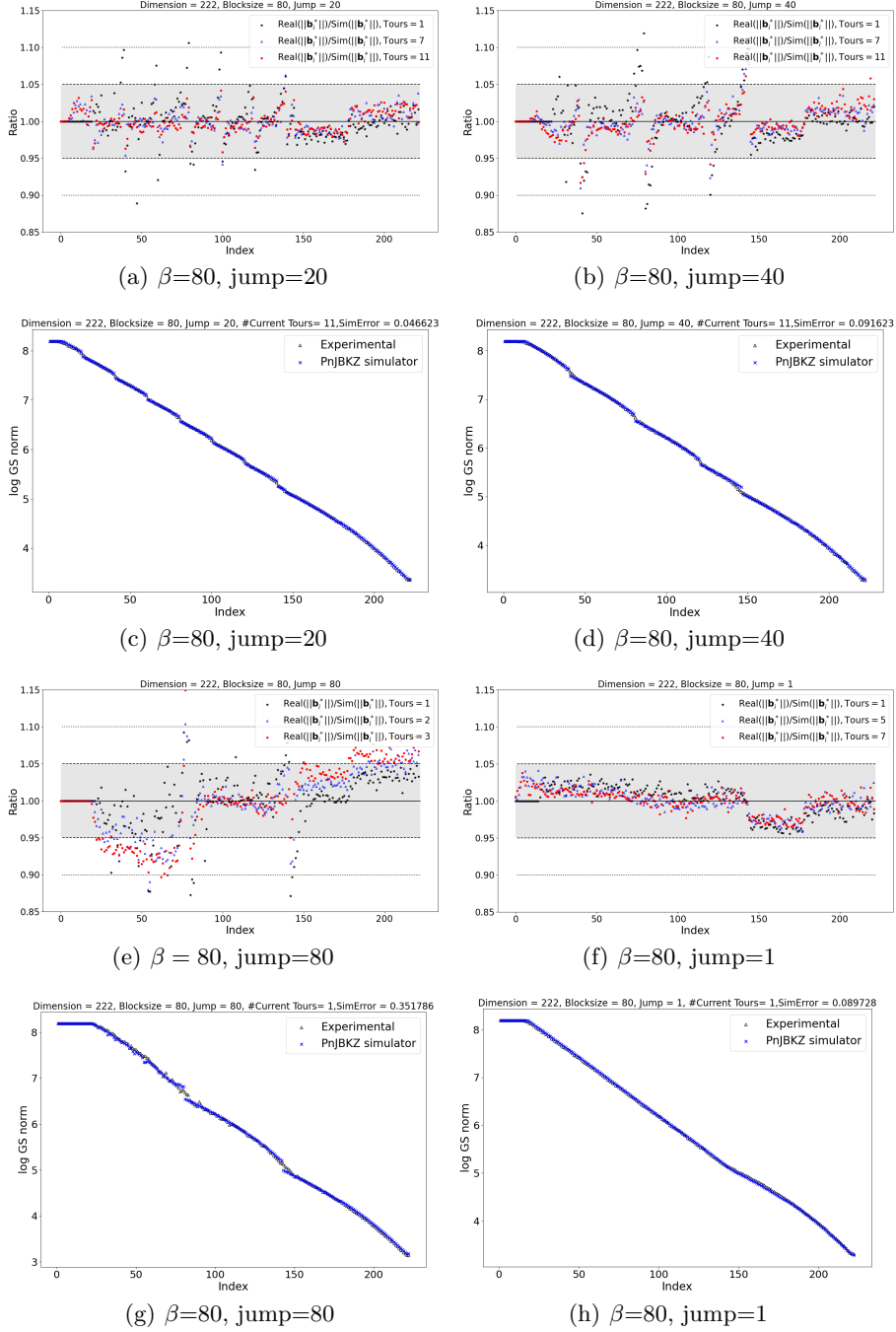


Fig. 9: Ratio $l'_i/\text{Sim}(l'_i)$ and corresponding overall prediction effect of PnJBKZ simulator. Run PnJBKZ(β, J) reduction on a 222-dimension LWE lattice basis ($n=60, \alpha=0.010$) and record the ratio values. We test 20 times for each reduction parameter.

D.2 Predicating PnJBKZ using d4f technology

The ideal PnJBKZ simulation discussed in Section 5.1 did not consider the influence of using dimension for free (d4f) technology. In practice, the implementation of PnJBKZ used in [20] and [28] default use d4f technology to reduce the running time and memory requirements of sieving. Additionally, the PnJBKZ algorithm is often accelerated by heuristically increasing the d4f value. For example, in the G6K implementation [20], the authors use a more optimistic d4f value compared to the theoretical one. Specifically, in [26] has given two theoretical d4f estimations for solving β -dimension SVP as $d4f(\beta) = \beta \ln(4/3) / \ln(\beta/2\pi)$ and $d4f(\beta) = \beta \ln(4/3) / \ln(\beta/2\pi e)$, while in the implementation of G6K [20], it gives a more relaxed value and we called it “optimistic d4f”: $d4f(\beta)$: when $\beta < 40$, $d4f(\beta) = 0$; when $40 \leq \beta \leq 75$, $d4f(\beta) = \lfloor \beta - 40/2 \rfloor$; when $\beta > 75$, $d4f(\beta) = \lfloor 11.5 + 0.075\beta \rfloor$.

After using d4f to accelerate PnJBKZ, each Pump in the algorithm will perform only $(\beta - d4f)$ -dimensional progressive sieving during the Pump-up stage and can thus perform at most $(\beta - d4f)$ embeddings during the Pump-down stage. Consequently, the output basis of each β -dimensional Pump will inevitably include $d4f$ -dimensional GS vectors that do not satisfy the HKZ-reduced basis properties.

However, the PnJBKZ(β, J) with the optimistic d4f setting is quite efficient in practice. To ensure that the PnJBKZ simulator accurately predicts the Gram-Schmidt (GS) values of a PnJBKZ-reduced lattice basis when utilizing d4f technology, including optimistic d4f values, we employ a refined d4f value estimation proposed in [44]. Additionally, we adjust our simulation strategy to enhance the accuracy of the simulator in predicting the behavior of PnJBKZ reductions that employ d4f technology and optimistic d4f settings. In this Appendix D.2 and following verification experiments parts: Appendix D.3, and D.4, all PnJBKZ reduction experiments use the “optimistic d4f” settings applied in the G6K implementation [20].

Wang et al. in [44] proposed an more refined d4f value estimation function based the quality of current lattice basis. Specifically, firstly, we revisit the estimation of the upper bound of the d4f value under the pessimistic condition in [26]:

$$\text{GH}(L) \leq \text{GH}(L_{[f,d]}) \sqrt{4/3} \quad (5)$$

The above inequality can be rewritten as $f \cdot \ln(\delta) \leq \ln(4/3) + \ln(1 - f/d)$ using GSA. Based on $\ln(\delta) = \Theta((\ln\beta)/\beta) = \Theta((\ln d)/d)$, [26] gives an asymptotic value of d4f under pessimistic condition as:

$$f \approx \frac{d \ln(4/3)}{\ln(d/2\pi)} \quad (6)$$

The optimistic condition in [26]:

$$\text{GH}(L) \sqrt{\frac{d-f}{d}} \leq \text{GH}(L_{[f,d]}) \sqrt{\frac{4}{3}}. \quad (7)$$

Using the GSA and optimistic asymptotic value of d4f can be rewritten as:

$$f \approx \frac{d \ln(4/3)}{\ln(d/2\pi e)} \quad (8)$$

The theoretical derivations above are based on the assumption that the current d -dimensional lattice basis is fully-BKZ- $d/2$ reduced. However, during the process of actual lattice reduction, the reduction quality of the lattice basis is gradually improved rather than remaining fixed at a specific level of reduction quality, especially in the initial stage, when it is far away from being fully-BKZ- $d/2$ reduced. The effect of lattice reduction quality improvement on d4f is not considered by Eq.(7) and Eq.(8).

So these two different maximum values of d4f analyzed in [26] only depends on the dimension of the lattice and they are not accurate estimations but asymptotic upper bounds. We try to give a refined estimation of the max value of d4f based on the GSA coefficient.

Since under the GH $\lambda_1(L_{[f+1,d]}) = \text{GH}(L_{[f+1,d]}) \approx \sqrt{\frac{d-f}{2\pi e}} \cdot \left(\prod_{i=f+1}^d \|b_i^*\|\right)^{\frac{1}{d-f}}$, by using GSA, we have:

$$\lambda_1(L_{[f+1,d]}) = \sqrt{\frac{d-f}{2\pi e}} \frac{|\det(L)|^{\frac{1}{d-f}} \cdot \delta^{\frac{f(f-1)}{d-f}}}{\|\mathbf{b}_1^*\|^{\frac{f}{d-f}}}.$$

Since $\|\mathbf{b}_1^*\|$ can also be represented by $\delta^d \cdot |\det(L)|^{\frac{1}{d}}$, we have

$$\lambda_1(L_{[f+1,d]}) = \sqrt{\frac{d-f}{2\pi e}} \frac{|\det(L)|^{\frac{1}{d}}}{\delta^f} \quad (9)$$

Based on the pessimistic condition in [26]: $\text{GH}(L_{[f+1,d]}) \sqrt{4/3} \geq \text{GH}(L)$ and Eq. (9), we can get a conservative d4f estimation based on GSA:

$$f \leq \log_\delta \sqrt{\frac{4(d-f)}{3d}} \quad (10)$$

Based on the optimistic condition in [26]: $\text{GH}(L_{[f+1,d]}) \sqrt{4/3} \geq \pi_f(\text{GH}(L))$ and Eq. (9), we can get an optimistic d4f estimation based on GSA:

$$f \leq \log_\delta \sqrt{\frac{4}{3}} \quad (11)$$

It illustrates that under current reduction quality of lattice basis the max d4f value should be $d4f_\delta = \ln_\delta \sqrt{4/3} \approx \ln(4/3)/(-slope)$. Here the *slope* is the slope value of the logarithm of Gram-Schmidt norms l_i for $\forall i \in \{1, \dots, d\}$.

This leads to a problem: the actual d4f value that can be obtained under a certain reduction quality of lattice basis should theoretically be $d4f_\delta$. If the Jump size is not limited, there is a situation where $d4f_\delta < \text{Jump} < \text{the optimistic heuristic d4f value}$. Because at this time, the first ($\text{Jump} - d4f_\delta$) GS values of each

Pump can no longer be guaranteed to be the shortest vector on the corresponding projection sub-lattice, and finally the prediction of the first $(\text{Jump} - d4f_\delta)$ GS values of each Pump based on the HKZ property is no longer accurate.

To ensure PnJBKZ simulator still can accurately predict the GS values of PnJBKZ reduced lattice basis when using optimistic d4f value during PnJBKZ reduction. We need bound the maximum Jump value during reduction. We conclude this result in Heuristic 3. Later our experiments will show that in practice, when using optimistic d4f value during PnJBKZ reduction Heuristic 3 indeed holds under the appropriate reduction parameter.

Heuristic 3 (Pump outputs HKZ reduced basis) *Given a d -dimensional lattice basis \mathbf{B} with reduction quality that slope equals s . Under reduction parameter $J \leq d4f_{slope}(s) \ll \beta \leq d$. For $\kappa < d - 1$, $\mathbf{B}_{\pi[\kappa, \min\{\kappa+\beta, d\}]}$ reduced by a Pump($\mathbf{B}_{\pi[\kappa, \min\{\kappa+\beta, d\}]}$, $\kappa, \beta, f \leq d4f_{slope}(s)$), the first J vectors in block $\mathbf{B}_{\pi[\kappa, \min\{\kappa+\beta, d\}]}$ have the same Gram-Schmidt norms follow the length profile as a HKZ reduced block, i.e. under Gaussian Heuristic, for $i \in [\kappa, \kappa + J - 1]$, the expected norms of $\|\mathbf{b}_i^*\| \approx \text{GH}(\mathcal{L}(\mathbf{B}_{\pi[i, \min\{\kappa+\beta, d\}]})$.*

Here $d4f_{slope}(s)$ is an upper bound of the d4f value estimation function based on the quality of the current lattice basis proposed in [44]. $d4f_{slope}(s) := \ln_\delta \sqrt{4/3} \approx \ln(4/3)/(-s)$.

As we mentioned above the default d4f function used in the implementation of PnJBKZ (both [20] and [28]) is an optimistic heuristic setting. Such an optimistic d4f setting in the implementation of G6K leads to the actual reduction effort of a PnJBKZ(β, J) with the optimistic d4f setting is more closed to a PnJBKZ(β', J) with the theory d4f estimation value rather than a PnJBKZ(β, J) with the theory d4f estimation value. Here $\beta' \leq \beta$.

Therefore, to more accurately predict the behavior of the PnJBKZ which uses the optimistic d4f function, we give the following simulation strategy. Specifically, using the information of the quality of the current lattice basis, like the slope value s , [44] calculates the refined d4f value estimation by $d4f_{slope}(s) = \ln(4/3)/(-s)$. In this case $J \leq d4f_{slope}(s)$, we calculate $d4fgap(\beta, s) := d4f_{optimistic}(\beta, s) - d4f_{slope}(s)$. If $\beta < 40$, $d4fgap(\beta, s) = 0$; $d4fgap(\beta, s) = \lfloor \beta - 40/2 - d4f_{slope}(s) \rfloor$, if $40 \leq \beta \leq 75$; $d4fgap(\beta, s) = \lfloor 11.5 + 0.075\beta - d4f_{slope}(s) \rfloor$, if $\beta > 75$. Then, we calculate $\beta_{sim} = \beta - d4fgap(\beta, s)$ and replace each blocksize β by β_{sim} as the input of Alg. 4, when using PnJBKZ simulator.

The function $d4fgap(\beta, s)$ aims to calculate the gap between the optimistic d4f setting in the implementation of G6K and the real d4f value under current lattice reduction quality (slope s value). Such a simulation strategy is based on the current lattice reduction quality information (slope value) give a more refined d4f estimation to adjust the over-optimistic d4f used in the default implement of PnJBKZ [20].

Finally, in Appendix D.3 and D.4, we show the verification experiments of Heuristic 3 and the PnJBKZ simulator for predicting PnJBKZ reduction with optimistic d4f settings.

Besides, the verification experiments comparison of PnJBKZ estimations under different d4f estimations can be seen in Section 3.2 of [44]. The comparison results indicate that using the simulation strategy we mention above, can be more accurate in predicting the behavior of PnJBKZ which uses the optimistic d4f function. In this paper, we default use the above simulation strategies to predict the practical reduction effect of the PnJBKZ(β, J) which uses the optimistic d4f setting in practice.

D.3 Verification Experiments of Heuristic 3 and the PnJBKZ Simulator for Predicting PnJBKZ Reduction with Optimistic d4f Settings

In this part, we show that Heuristic 3 is held when the jump parameter J of PnJBKZ is below a specific upper bound. Therefore it is reasonable to use the properties of the HKZ reduction basis to simulate the actual reduction effect of PnJBKZ.

In this part, our experiments tested on the TU Darmstadt LWE Challenge lattice basis with parameter ($n=60, \alpha=0.010$), and before running the PnJBKZ simulator we did small block reduction to remove the influence of q-ary vectors in the LWE Challenge lattice basis. After our pre-processing, we obtain a 222-dimension lattice basis which has a few q-ary vectors in the front of the lattice basis with a slope value equal to -0.0248 (the walltime of such a pre-processing within a few minutes). Then we calculate the ratio $l_i''/\text{Sim}(l_i'')$ for $i \in [0, d-1]$ in each tour of PnJBKZ's reduction, see Fig. 10. Here l_i'' is the average logarithms of these Gram-Schmidt vector lengths obtained from 20 independent reduction experiments that use the same reduction parameter (β, J, tours) do 20 times PnJBKZ reduction respectively, and $\text{Sim}(l_i'')$ is the simulated logarithm of lengths of Gram-Schmidt vector which are calculated by Eq. (3).

We calculate the $\text{Sim}(l_i'')$ strictly according to the property of the HKZ reduction basis and Gaussian Heuristic. Therefore, in addition to being one of the criteria for measuring the accuracy of the PnJBKZ simulator, this ratio $l_i''/\text{Sim}(l_i'')$ can also be used as a criterion for judging whether Heuristic 3 is held. In particular, it can be seen from Fig. 10 that for β from 85 increasing to 100 and **jump** from 1 increasing to 12, which is the minimum theoretical upper bound value under the current quality of lattice basis: $\text{d4f}_{\text{slope}}(s = -0.0248) \approx 11.6 \leq 12$. When **jump** $\leq \lceil \text{d4f}_{\text{slope}}(s = -0.0248) \rceil = 12$, even the **tours** increase to 12, all most of the ratios $l_i''/\text{Sim}(l_i'')$ are within range: $[0.95, 1.05]$ (the rest ratios are also within range $[0.90, 1.10]$). Fig. 10 indicates that based on the refined d4f estimation, our PnJBKZ simulator is accurate in predicting the reduction effort of PnJBKZ and Heuristic 3 is held when $J \leq \text{d4f}_{\text{slope}}(s)$.

In comparison, when simulating classic BKZ, most ratios also fall within $[0.95, 1.05]$, though some ratios at the extremes exceed this range, with the largest ratio reaching 1.15 and the smallest falling below 0.85. This indicates that the prediction accuracy of the PnJBKZ simulator for PnJBKZ reductions using d4f technology is at least as good as that of the classic BKZ simulator [17] for predicting BKZ reductions.

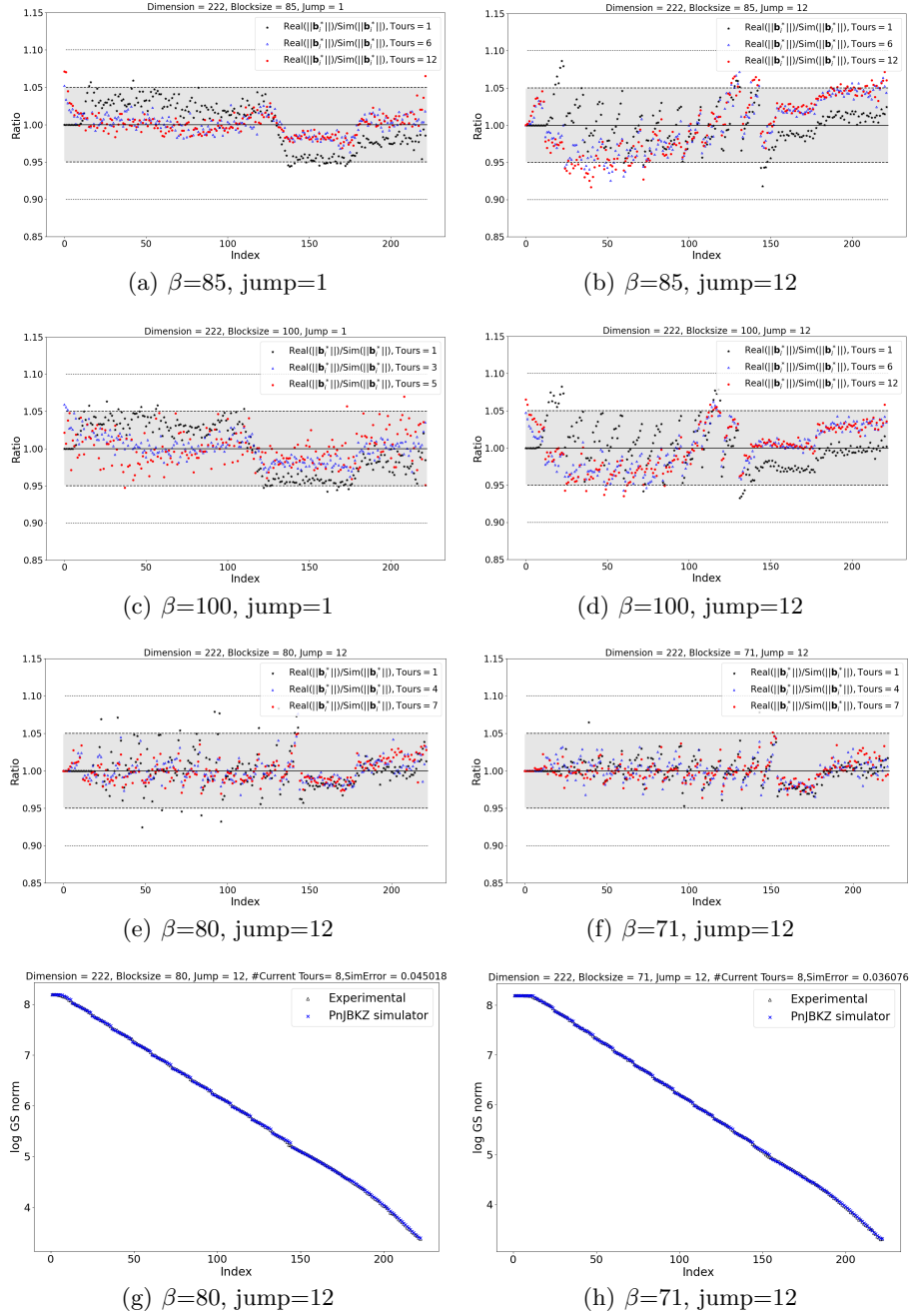


Fig. 10: Ratio $l''_i / \text{Sim}(l''_i)$. Run PnJBKZ(β, J) reduction on a 222-dimension LWE lattice basis ($n=60, \alpha=0.010$) and record the ratio values. We test 20 times for each reduction parameter.

Moreover, this also is verified by (e) and (f) in Fig. 11 when the tours increase to 13. Meanwhile, the PnJBKZ simulator using Eq. (3) as the approximate estimate of the actual value l_i'' can already reflect how the average of the norms of Gram-Schmidt vectors change during each tour's reduction of PnJBKZ(β, J) which uses the optimistic d4f setting in practice.

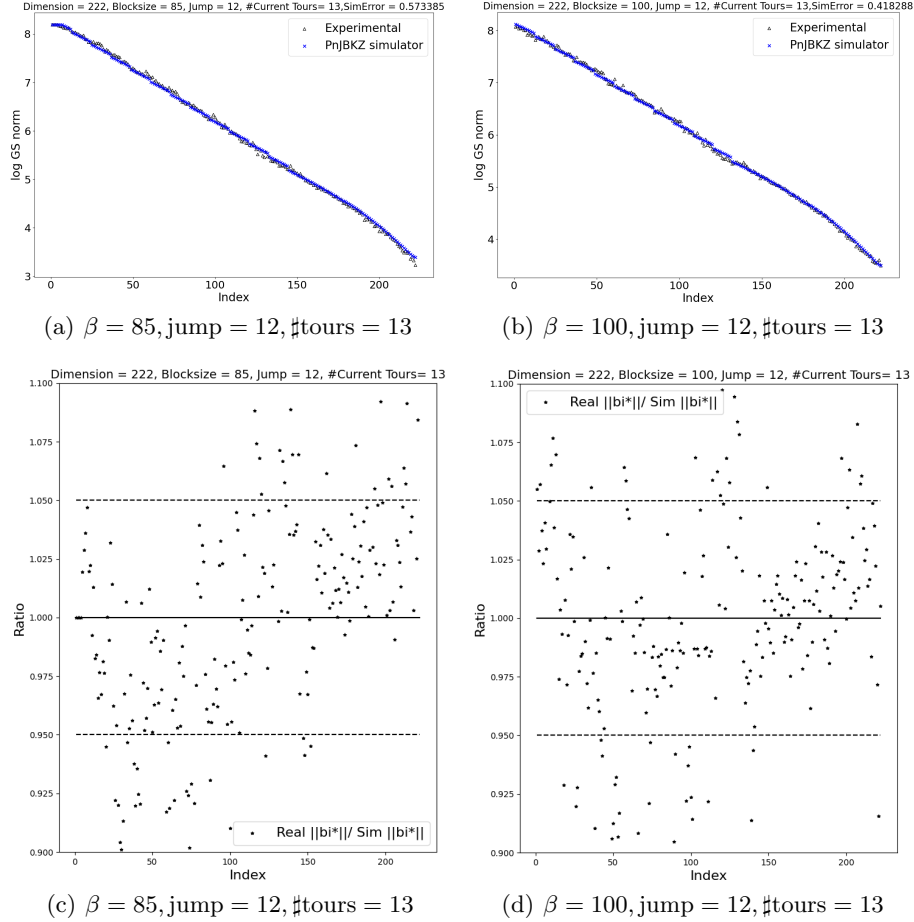


Fig. 11: Overall Prediction effect of PnJBKZ simulator. Ratio $l_i''/\text{Sim}(l_i'')$. We perform the experiments on a reduced lattice basis of LWE Challenge ($n=60, \alpha=0.010$) with slope value $s=-0.0248$ when jump increasing to the minimum theoretical upper bound $\lceil \text{d4f}_{\text{slope}}(s) \rceil = 12$. We test also 20 times for each reduction parameter and show the average value of the experiments.

We set $\text{SimError}(\# \text{tours}) = \sum_{i=0}^{d-1} (\|\mathbf{b}_i^*\|_{(\# \text{tours})} - \text{Sim}(\|\mathbf{b}_i^*\|)_{(\# \text{tours})})^2$, where $\# \text{tours}$ represents the number of current tours and $\text{Sim}(\|\mathbf{b}_i^*\|)_{(\# \text{tours})}$ are the lengths of Gram-Schmidt vectors predicted by PnJBKZ simulator with $\# \text{tours}$. Then we give Fig. 11 which shows that the overall prediction error of the PnJBKZ simulator with different jumps is similar to that of $\text{jump}=1$.

More verification experiment results with different reduction parameters on different lattice bases can be seen in Appendix D.4.

In addition, our experiments, detailed in Table 3 and Appendix K, demonstrate that for the LWE challenge lattice basis with various reduction parameters (e.g. different block sizes and jump sizes), the predicted slope values from our PnJBKZ simulator closely match those obtained from actual reductions. These results also show that, based on refined d4f estimation, the PnJBKZ simulator accurately predicts the average reduction effect of PnJBKZ using d4f. Therefore, the PnJBKZ simulator we have constructed is sufficiently effective for its intended purpose.

D.4 More experimental details about PnJBKZ simulator for predicting PnJBKZ with optimistic d4f

In this section, we present additional verification experiments of our PnJBKZ simulator for predicting the reduction effort of the PnJBKZ algorithm, which uses optimistic d4f setting, including optimistic d4f settings, on various LWE challenge lattice bases with different reduction parameters. Specifically, we varied the block size β from 55 to 100, the jump value from 1 to 12, and the number of tours from 1 to 13. For each set of reduction parameters, we conducted 20 independent experiments to calculate the practical average length of the Gram-Schmidt vectors after applying the PnJBKZ reduction.

First of all, to remove the influence of q-ary vectors in LWE challenge initial lattice basis, we do pre-processing for all LWE challenge lattice basis by using small block size reduction which can be done within a few minutes wall time. For example ($n = 70, \alpha = 0.005$) and ($n = 75, \alpha = 0.005$), after pre-processing we can get LWE challenge lattice basis ($n = 70, \alpha = 0.005$) with a slope equal to $-0.04921/2$ and LWE challenge lattice basis ($n = 75, \alpha = 0.005$) with a slope equals to $-0.04339/2$. Then corresponding $d4f_{slope}(s)$ between 11.7 to 13.7. As we need the maximum jump value $J \leq d4f_{slope}(s)$ to ensure the accuracy of the PnJBKZ simulator (see section 5 for details), we set the maximum jump value $J \leq 12$ in our test experiments.

Then we give the results of verification experiments on four different lattice bases with $\beta \in [50, 70]$ and jump within $J \in [1, 12]$: ($n = 70, \alpha = 0.005$), ($n = 75, \alpha = 0.005$), ($n = 60, \alpha = 0.010$) and ($n = 50, \alpha = 0.015$). See Figure 14 ~ 22 respectively. Verification experiments results indicate that our PnJBKZ simulator performs well in predicting the behavior of PnJBKZ which block size within $\beta \in [75, 100]$ and jump within $J \in [1, 12] \leq d4f_{slope}(s)$ on LWE challenge lattice basis on 4 different LWE challenge lattice bases.

Figures 12 ~ 22 show that on different lattice basis with different reduction parameters, as long as the jump $\leq d4f_{slope}(s)$, even the tours increase to 13, overall, the simulation values are closed to the actual values and most of ratios $\frac{l'_i}{\text{Sim}(l'_i)}$ are within $[0.95, 1.05]$. Meanwhile, the progressive reduction strategy will not run the same reduction parameter (β, J) for more than 10 tours, while our simulator is still accurate even if the tours increase to 13. The above results

indicate that when $J \leq \text{df}_{\text{slope}}(s)$ we can ensure Heuristic 3 is held and the PnJBKZ simulator calculates the estimation value of the actual value $\|\mathbf{b}_i''^*\|$ by Eq. (3) can already reflect how the average of the norms of Gram-Schmidt vectors change during each tour's reduction of PnJBKZ(β, J). Therefore our PnJBKZ simulator fits well with the actual PnJBKZ reduction result. In addition, Table 3 and Appendix K show that the practical slope of lattice Gram-Schmidt basis after each tour of the reduction of PnJBKZ with different blocksizes and different jump values is very close to that of our simulation, which also verified the accuracy of our PnJBKZ simulator. For selecting the optimized reduction strategy, our PnJBKZ simulator is accurate enough.

LWE challenge lattice basis ($n = 75, \alpha = 0.005$). Figure 12 ~ Figure 13.

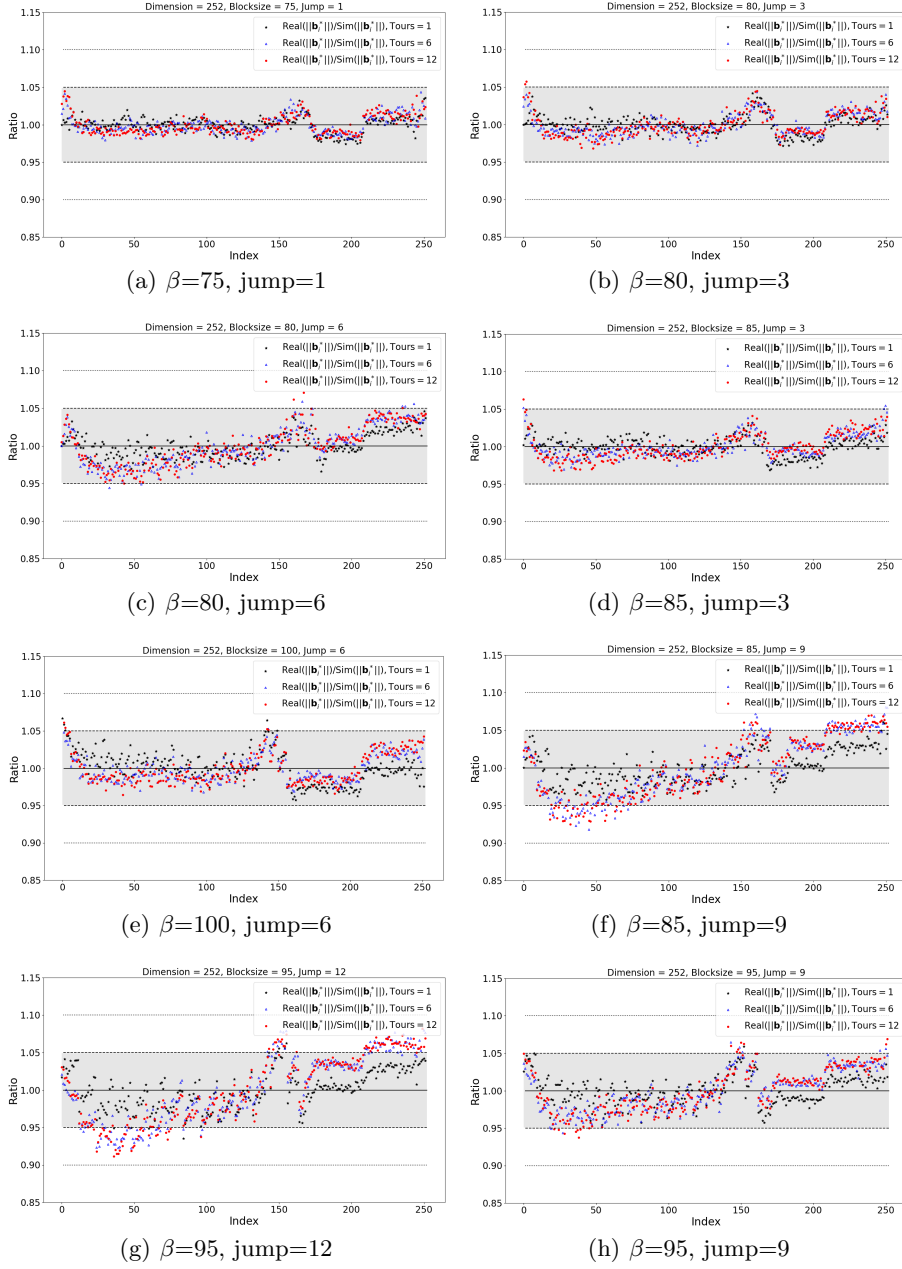


Fig. 12: Ratio $l_i''/\text{Sim}(l_i'')$. Run 12 tours of PnJBKZ(β, J) reduction on a 252-dimension LWE lattice basis ($n = 75, \alpha = 0.005$), and record the ratio values. We test 20 times for each reduction parameters.

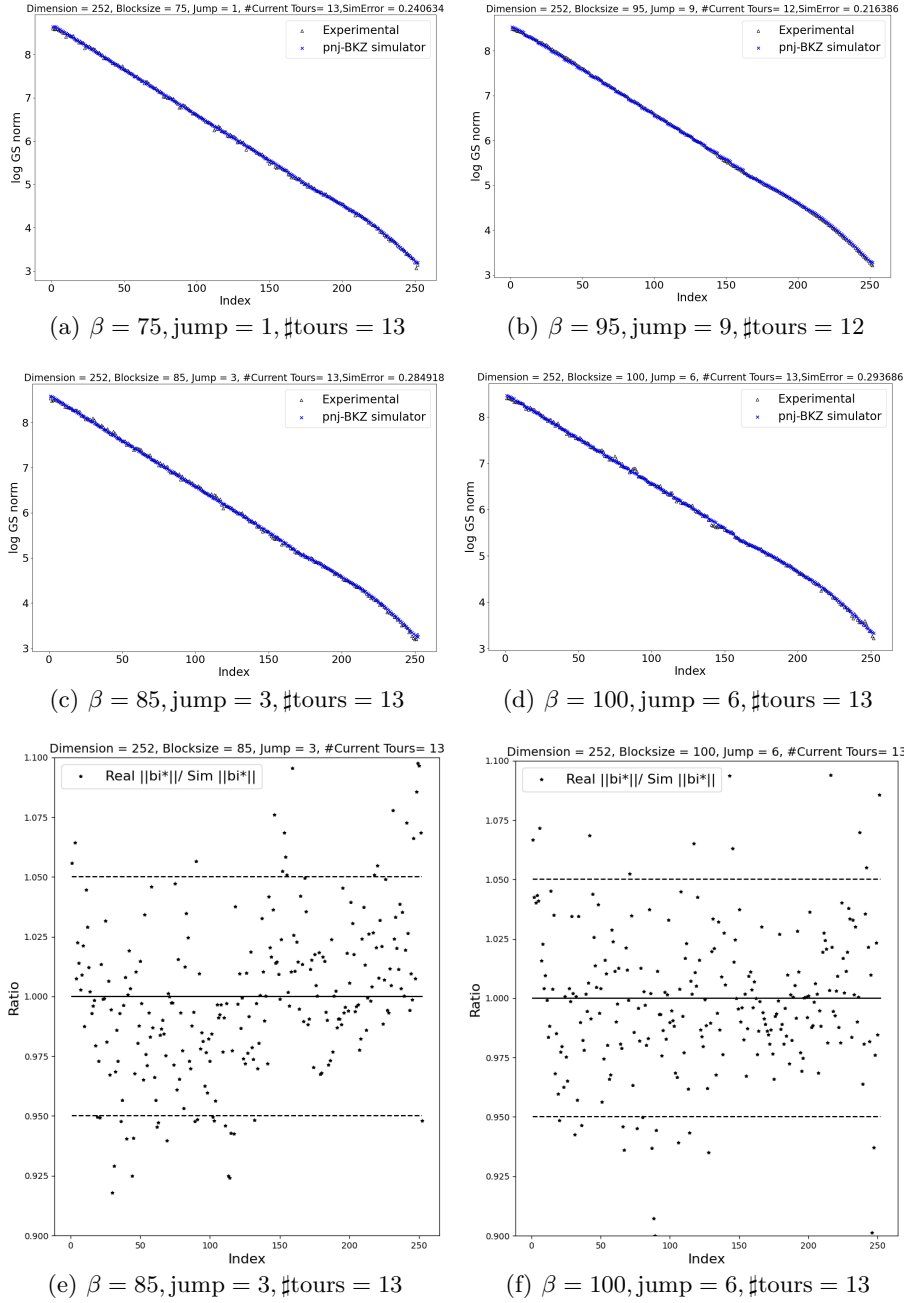


Fig. 13: Overall Prediction effect of PnJBKZ simulator. Ratio $l_i''/\text{Sim}(l_i'')$. We perform the experiments by reducing the lattice basis of LWE Challenge ($n = 75, \alpha = 0.005$). We test also 20 times for each reduction parameters.

LWE challenge lattice basis ($n = 70, \alpha = 0.005$). Figure 14 ~ Figure 16.

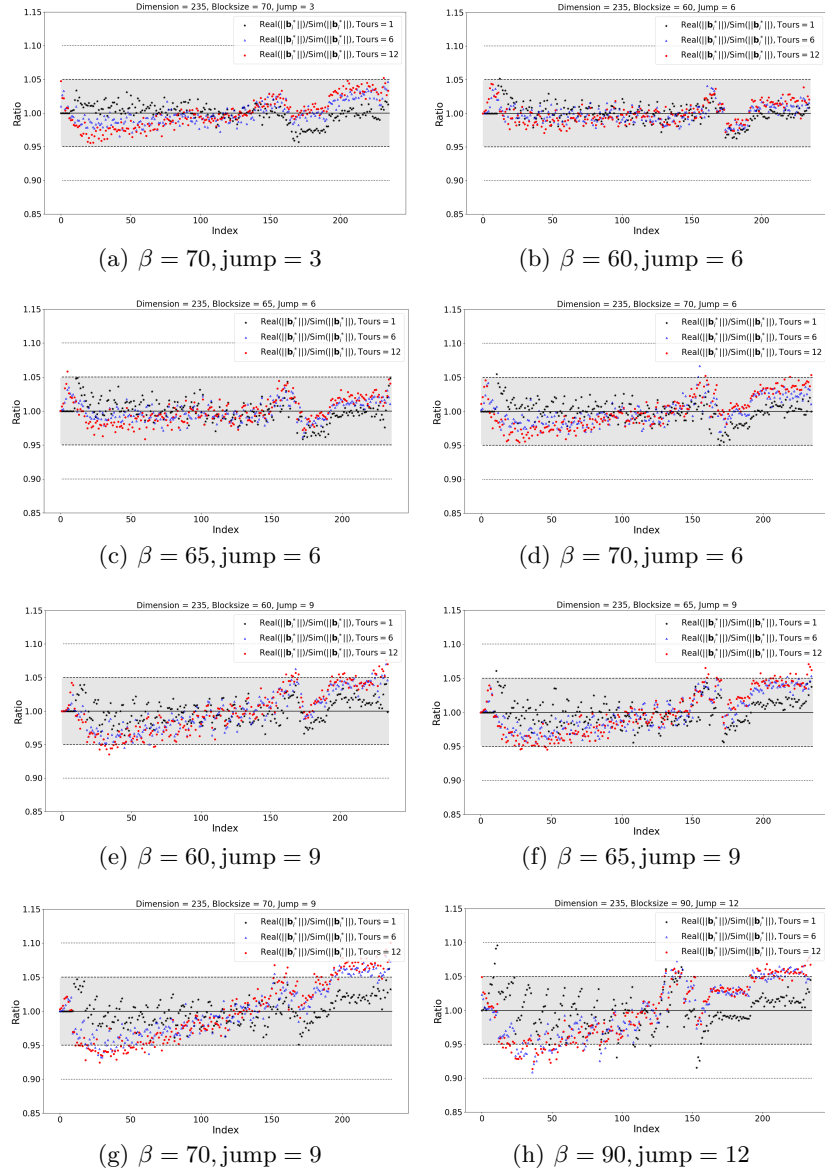
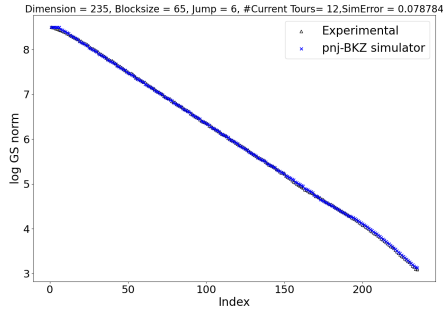
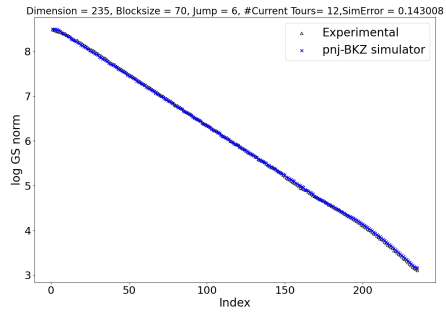


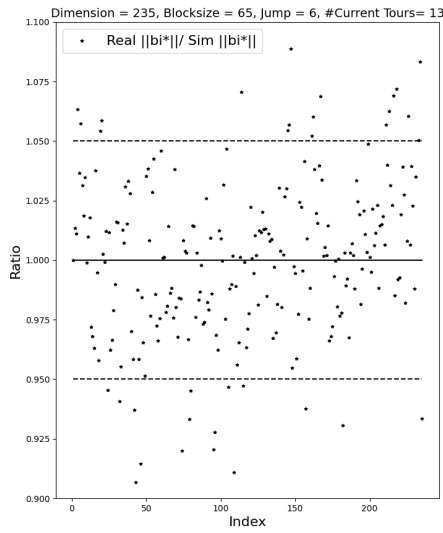
Fig. 14: Ratio $l_i'' / \text{Sim}(l_i'')$. Run 12 tours of PnJBKZ(β, J) reduction on a 235-dimension LWE lattice basis ($n = 70, \alpha = 0.005$), and record the ratio values. We test 20 times for each reduction parameter.



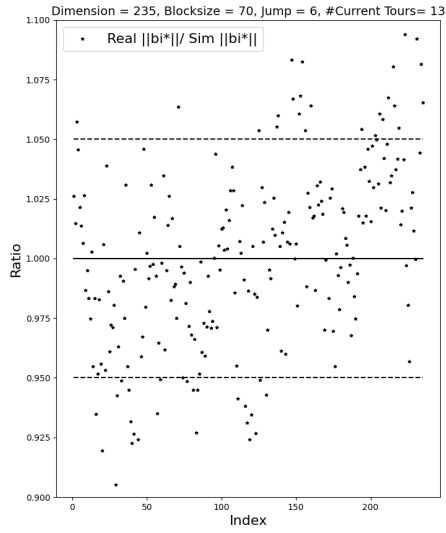
(a) $\beta = 65$, jump = 6, #tours = 12



(b) $\beta = 70$, jump = 6, #tours = 12

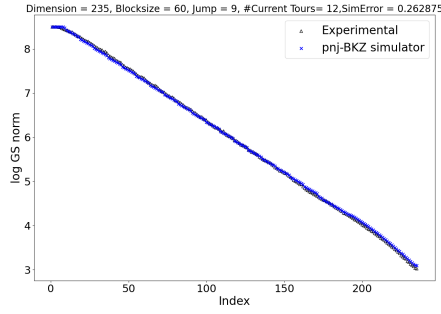


(c) $\beta = 65$, jump = 6

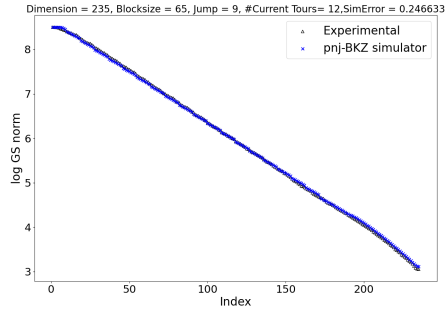


(d) $\beta = 70$, jump = 6

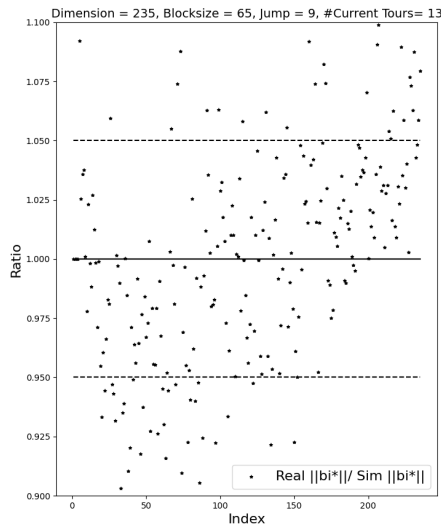
Fig. 15: Ratio $l_i''/\text{Sim}(l_i'')$. Run 13 tours of PnJBKZ(β, J) reduction on a 235-dimension LWE lattice basis ($n = 70, \alpha = 0.005$), different β with $J = 6$, and record the ratio values. We test 20 times for each reduction parameter.



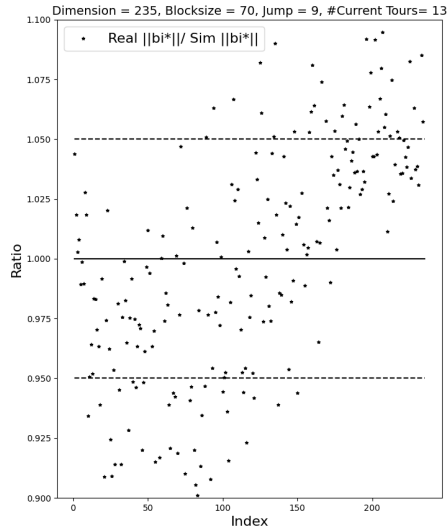
(a) $\beta = 60$, jump = 9, #tours = 12



(b) $\beta = 65$, jump = 9, #tours = 12



(c) $\beta = 65$, jump = 9



(d) $\beta = 70$, jump = 9

Fig. 16: ratio $l''_i/\text{Sim}(l''_i)$. Run 13 tours of PnJBKZ(β, J) reduction on a 235-dimension LWE lattice basis ($n = 70, \alpha = 0.005$), different β with $J = 9$, and record the ratio values. We test 20 times for each reduction parameter.

LWE challenge lattice basis ($n = 60, \alpha = 0.010$). Figure 17 ~ Figure 19.

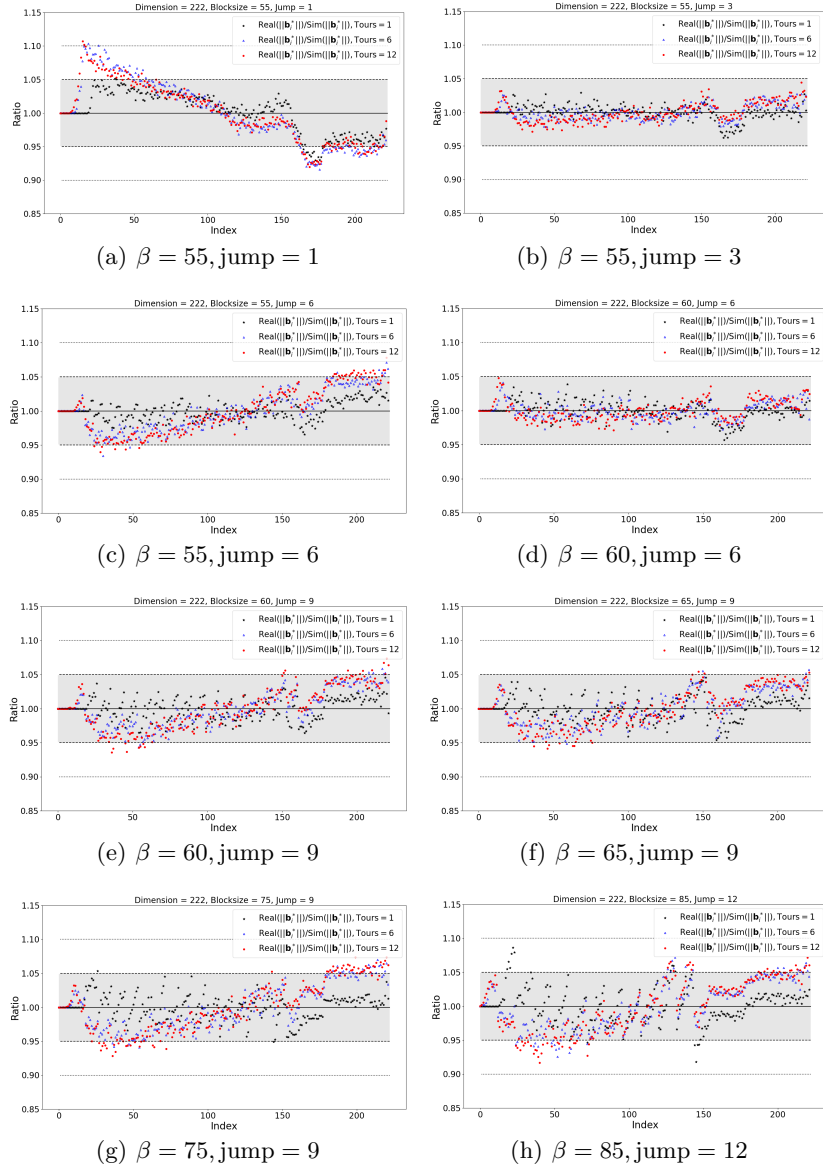


Fig. 17: Ratio $l'_i / \text{Sim}(l'_i)$. Run 12 tours of PnJBKZ(β, J) reduction on a 222-dimension LWE lattice basis ($n = 60, \alpha = 0.010$), and record the ratio values. We test 20 times for each reduction parameter.

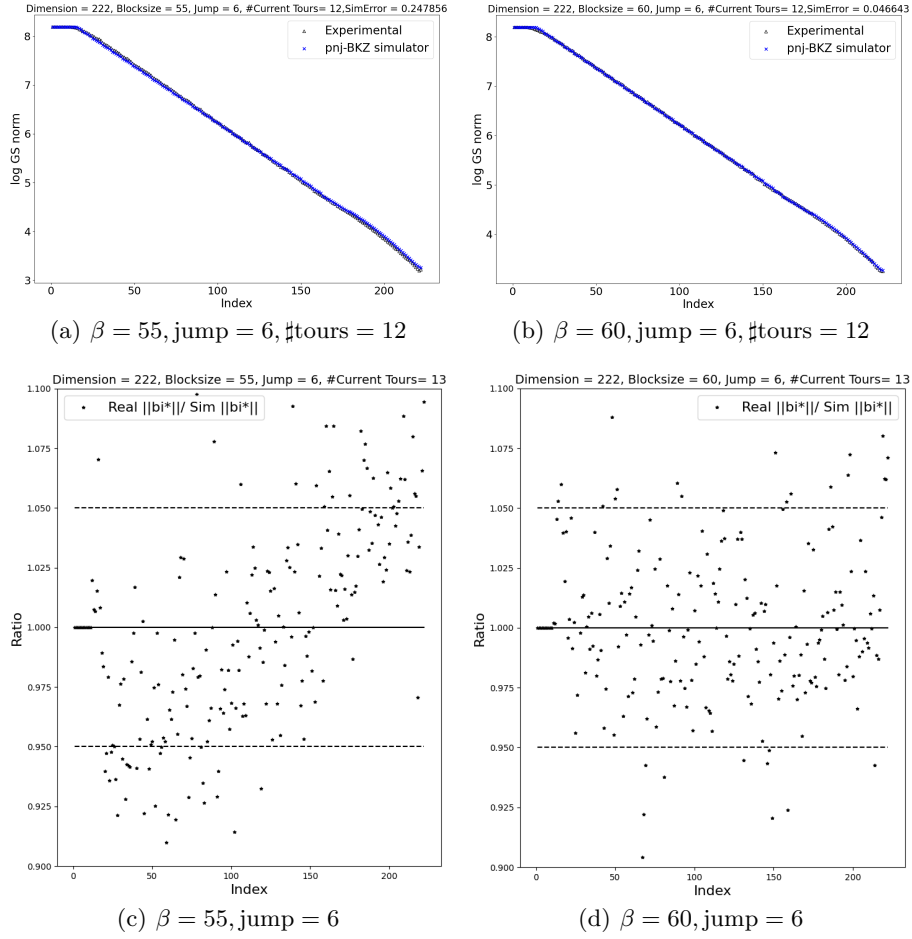
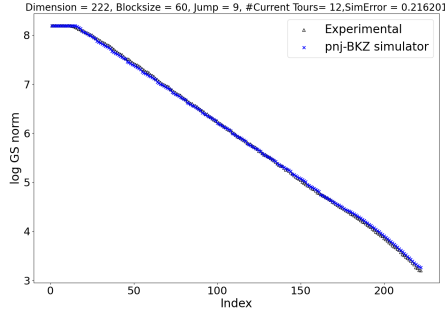
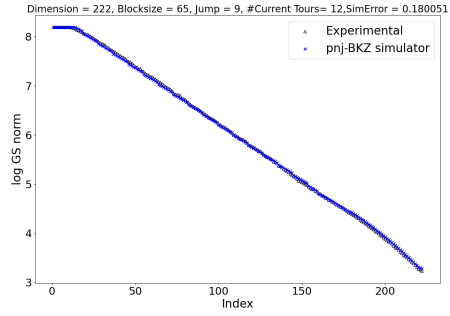


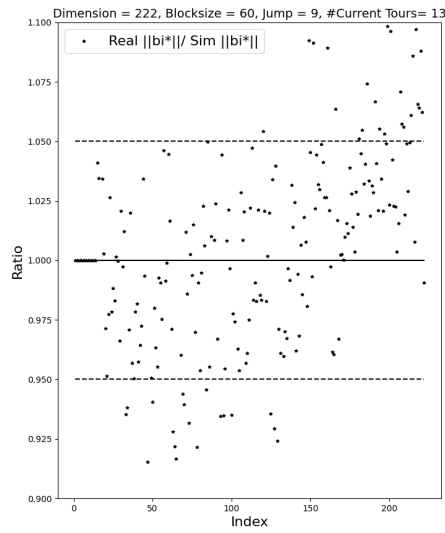
Fig. 18: Ratio $l_i'' / \text{Sim}(l_i'')$. Run 13 tours of PnJBKZ(β, J) reduction on a 222-dimension LWE lattice basis ($n = 60, \alpha = 0.010$), different β with $J = 6$, and record the ratio values. We test 20 times for each reduction parameter.



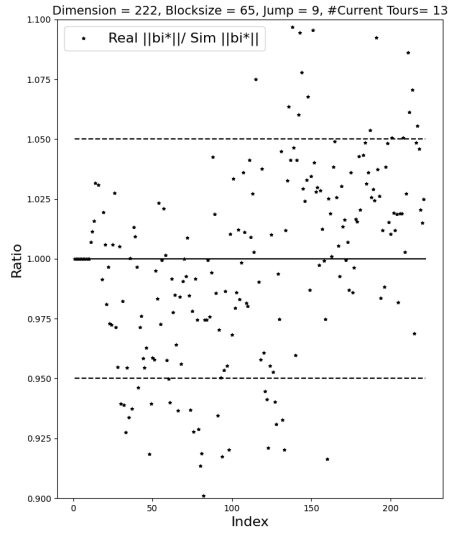
(a) $\beta = 60$, jump = 9, #tours = 12



(b) $\beta = 65$, jump = 9, #tours = 12



(c) $\beta = 60$, jump = 9



(d) $\beta = 65$, jump = 9

Fig. 19: Ratio $l_i''/\text{Sim}(l_i'')$. Run 13 tours of PñJBKZ(β, J) reduction on a 222-dimension LWE lattice basis ($n = 60, \alpha = 0.010$), different β with $J = 9$, and record the ratio values. We test 20 times for each reduction parameter.

LWE challenge lattice basis ($n = 50, \alpha = 0.015$). Figure 20 ~ Figure 22.

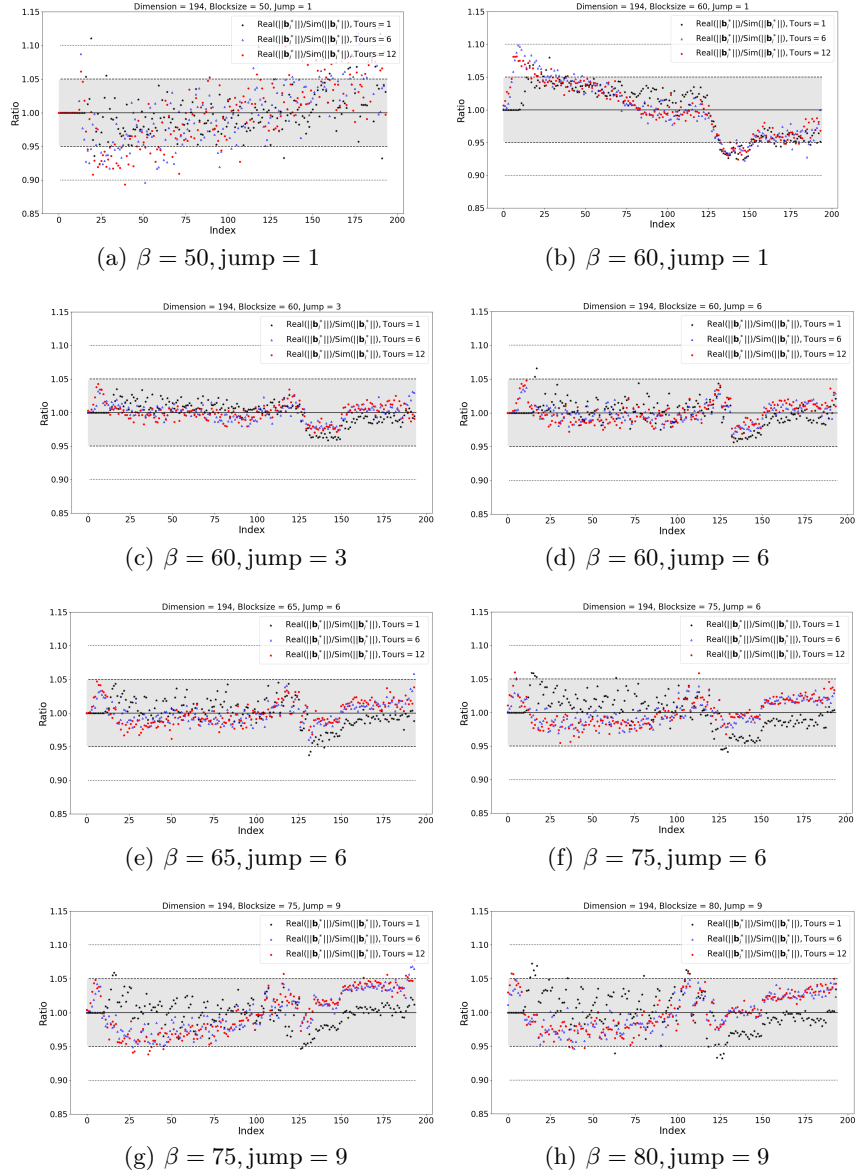
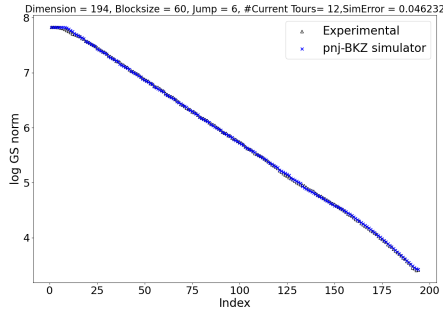
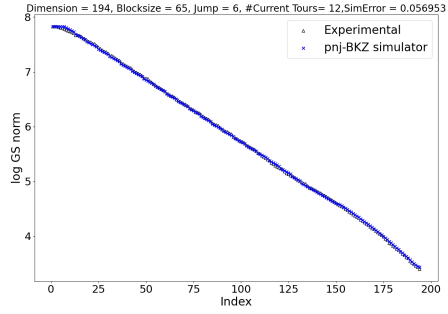


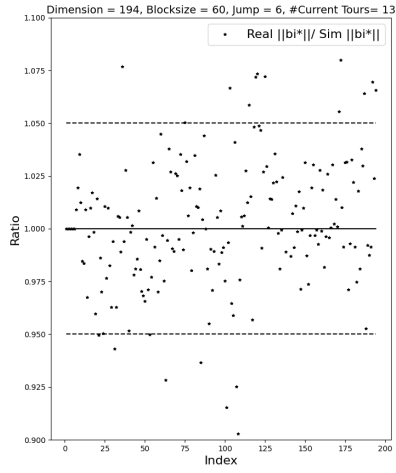
Fig. 20: Ratio $l_i'' / \text{Sim}(l_i'')$. Run 12 tours of PnJBKZ(β, J) reduction on a 194-dimension LWE lattice basis ($n = 50, \alpha = 0.015$), and record the ratio values. We test 20 times for each reduction parameter.



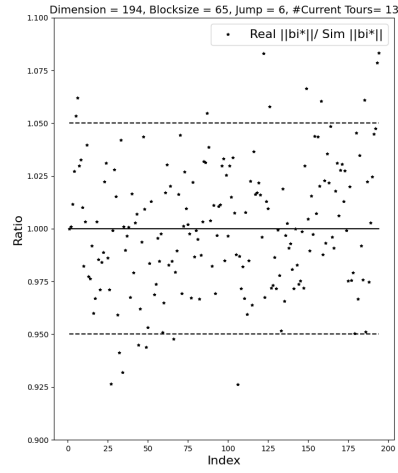
(a) $\beta = 60$, jump = 6, #tours = 12



(b) $\beta = 65$, jump = 6, #tours = 12



(c) $\beta = 60$, jump = 6



(d) $\beta = 65$, jump = 6

Fig. 21: Ratio $l_i''/\text{Sim}(l_i'')$. Run 13 tours of PnJBKZ(β, J) reduction on a 194-dimension LWE lattice basis ($n = 50, \alpha = 0.015$), different β with $J = 6$, and record the ratio values. We test 20 times for each reduction parameter.

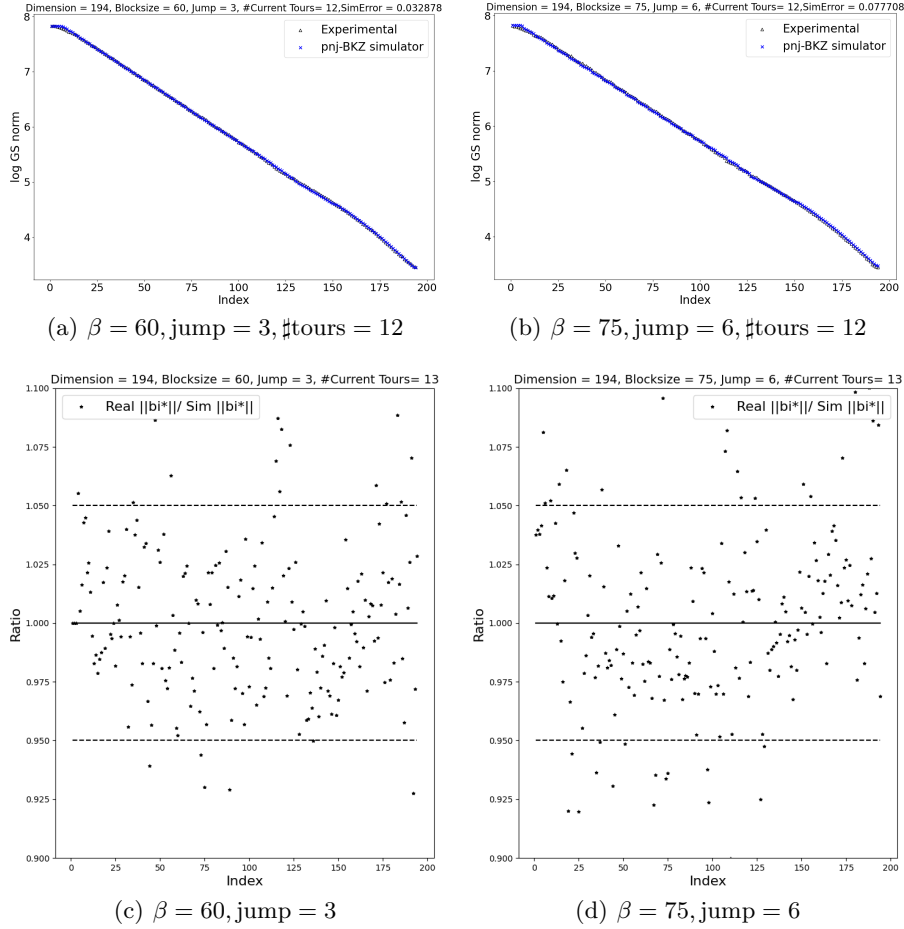


Fig. 22: Ratio $l_i''/\text{Sim}(l_i'')$. Run 13 tours of PnJBKZ(β, J) reduction on a 194-dimension LWE lattice basis ($n = 50, \alpha = 0.015$), different β and J , and record the ratio values. We test 20 times for each reduction parameter.

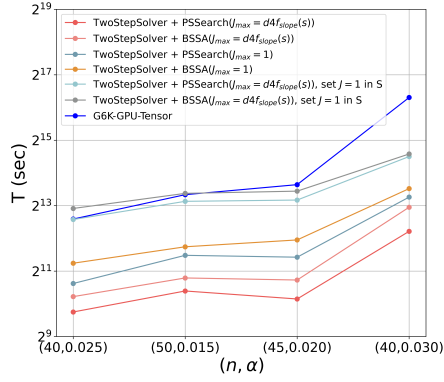
E Experiments and Application to LWE

E.1 Efficiency of MinTwoStepSolver for solving LWE

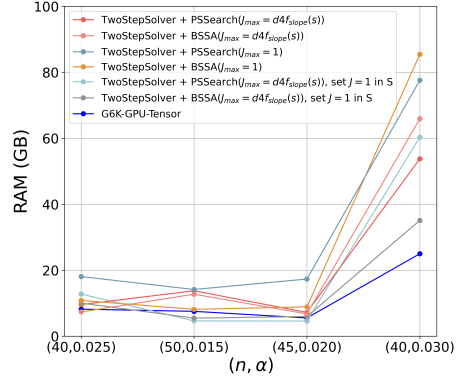
The default LWE solving algorithm in G6K is the script `lwe_challenge.py` in the implementation of G6K-GPU-Tensor [28]. Besides, for more detail about the default LWE solving algorithm in G6K-GPU-Tensor⁴. Fig. 23 gives the experimental result of different LWE-solving algorithms. We use the practical cost model proposed in Appendix G as \mathcal{T} .

⁴ https://github.com/WvanWoerden/G6K-GPU-Tensor/blob/main/lwe_challenge.py

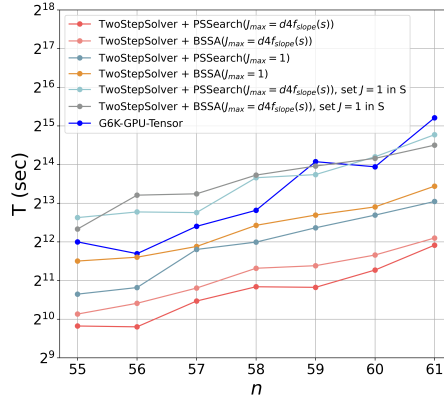
The blue lines and scatter points in Fig. 23 represent the experimental time or memory cost of the default strategy in G6K. The remaining lines and scatter points in Fig. 23 indicate the experimental time or memory cost of MinTwoStep-Solver using the strategy generated by PSSearch(Alg. 3) (BSSA(Alg. 6)). We modify the progressive blocksize selection strategy from ProBKZ [19] to adapt it for PnJBKZ by constructing the PnJBKZ simulator. We refer to this new strategy selection algorithm as the *blocksize and jump strategy selection algorithm based on ProBKZ* (BSSA). See Appendix H for more details about BSSA. The advantage of BSSA is that it runs in polynomial time; however, it cannot provide the time-minimal LWE solving strategy like PSSearch.



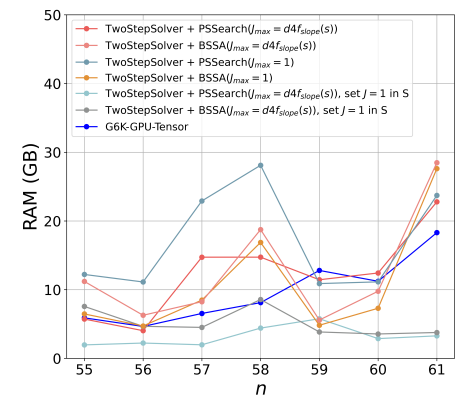
(a) LWE Challenges: Time.



(b) LWE Challenges: Memory.



(c) LWE Instances ($\alpha = 0.010$): Time.



(d) LWE Instances ($\alpha = 0.010$): Memory.

§ The experiment used “dd” float type and pump/down=True, under identical benchmark conditions on a machine C (Sec. 6.5) with threads=32 and GPUs=2. “default G6K” refers to the method in g6k implemented in lwe_challenge.py. TwoStepSolver + PSSearch(·) (resp. TwoStepSolver + BSSA(·)) represents the cost of running TwoStepSolver with strategies from PSSearch (resp. BSSA). J_{\max} denotes the maximum jump value in the strategy. “Set J=1 in S” means generating a strategy S and then setting the jump value as 1.

Fig. 23: Comparison of Different LWE-solving Algorithms under same benchmark. §

From the result of Fig. 23, we can see that using the strategy selected by PSSearch (BSSA) significantly decreased the walltime cost by about 7.2~17.0 (5.2~10.2) times compared to that of the default LWE solving strategy in G6K when all LWE solvers use the same float type “dd” to calculate. One can refer to the log files of Fig. 23 in the folders lwechal-test and lwe-instance-test. It can also be reproduced by running the test code implement_lwechal_forall.sh and implement_lwe_instance_forall.sh in source code².

As detailed in Fig. 2, our approach significantly improves the speed of solving LWE and effectively addresses practical challenges. All experimental results, except those in Table 4, were obtained using 32 threads and 2 GPUs on a workstation with an Intel Xeon 5128 (16 cores, 32 threads)@2.3 GHz, 1.48 TB of RAM, and two NVIDIA RTX 3090 GPUs, referred to as machine C.

Fig. 23(a) and Fig. 23(c) also show that TwoStepSolver using the strategy generated by PSSearch (abbreviated as MinTwoStepSolver) is faster than TwoStepSolver using the strategy generated by BSSA (abbreviated as TwoStepSolver + BSSA). Specifically, in Fig. 23(c), MinTwoStepSolver is 3.71 to 9.81 times faster than the default G6K, while TwoStepSolver + BSSA is 2.43 to 8.63 times faster than the default G6K when testing the given LWE instances $(n, \alpha) \in \{(n, 0.010) \mid n = 55, \dots, 61\}$. This indicates that the progressive reduction strategy generated by ProBKZ [19] is not minimal in time cost. This is particularly evident for large n , with an acceptable memory cost, as shown in Fig. 23(d).

Additionally, experiments shown in Fig. 23 indicate that the flexible use of the `jump>1` solving strategy can solve LWE 4.88 to 7.85 times faster than the $J_{\max}=1$ solving strategy. Specifically, in Fig. 23(c), we set S_1 as the strategy generated by PSSearch with $J_{\max}=\text{d4f}_{\text{slope}}(s)$ and S_2 as the strategy generated by PSSearch with $J_{\max}=1$. The walltime cost of using strategy S_1 is 2.02 to 2.91 times faster than that of S_2 and 4.88 to 7.85 times faster than the scenario where J is set to 1 in S_1 . Here, the s in $\text{d4f}_{\text{slope}}(s)$ denotes the simulated slope of the lattice basis during the reduction process. It demonstrates a substantial improvement in reduction efficiency when the maximum jump exceeds 1, potentially by skip some sufficiently reduced lattice bases, since each Pump turing on sieving at Pump-down stage, and there is a large intersection between each Pump.

Furthermore, one can also limit the maximum memory usage of solving LWE by changing parameter “`--max_RAM`” to generate the solving strategy. We also designed an LWE sample optimized selection algorithm (Alg. 7) to optimize the number of chosen LWE samples in Appendix I, although its efficiency improvement is not significant (at most 2.2% in our test).

E.2 New LWE Records

TU Darmstadt LWE Challenge website presents Challenges for testing the efficiency of solving LWE which helps to estimate the hardness of LWE in practice.

By our new algorithm, i.e. MinTwoStepSolver, we have solved the LWE instances $(n, \alpha) \in \{(80, 0.005), (40, 0.035), (90, 0.005), (50, 0.025), (55, 0.020), (40, 0.040)\}$ in TU Darmstadt LWE Challenge website¹. See Fig. 2 for more details. Specifically we denoted a service with AMD EPYC™ 7002 Series 128@2.6GHz, NVIDIA 3090 * 8, 1.5T RAM as Machine A, and denoted a service with AMD EPYC™ 7002 Series 64@2.6GHz, a100 * 4, 512 GB RAM as Machine B. A workstation with Intel Xeon 5128 16c 32@2.3GHz, 1.48T RAM and NVIDIA RTX 3090 * 2, denoted as machine C. Then we listed the walltime and RAM cost in solving the above LWE Challenges in Table 4. The units of T in Tables 3 and 4 are seconds and hours, respectively. In Table 4, we can see that the cost of solving the LWE

Challenge with parameters $(n, \alpha) = (40, 0.0035)$ using the G6K-GPU [28] is 23.4 times longer than that of our method.

E.3 Security Estimation for NIST schemes

We re-estimate the security bit of LWE-based NIST schemes [52] under consideration of the influence of optimized solving strategy. Our new concrete hardness estimation of LWE⁶ answers Question 7 in Section 5.3 of [8] and narrows the security estimation error interval. For more details about the construction of our new concrete hardness estimator of the Two-step mode of solving LWE, please refer to the citation [29]. Our evaluation code is available at Github⁶.

Table 6: Refined Security Estimation results for NIST schemes.[‡]

	$\log_2 G / \log_2(\text{gates})$			$\log_2 B / \log_2(\text{bit})$			$\Delta \log_2 G$	
	Previous	Two-step		Previous	Two-step		S_0	S_{op}
		S_0	S_{op}		S_0	S_{op}		
Kyber512	146	142.6	141.4	94.0	99.1	98.1	3.4	4.6
Kyber768	208.9	205.5	204.3	138.7	144.0	143.2	3.4	4.6
Kyber1024	281.1	277.7	276.5	189.8	195.4	194.3	3.4	4.4
Dilithium-II	152.9	150.8	149.5	98.0	104.3	103.3	2.1	3.4
Dilithium-III	210.2	207.9	206.7	138.8	145.3	144.3	2.3	3.5
Dilithium-V	279.2	277.0	275.7	187.52	194.1	193.0	2.2	3.5

[‡] “Previous” is the security estimation in the statement of Kyber and Dilithium. “ $S_0 = \{(\beta_i = i + 2, J_i = 1) \mid i = 1, \dots, \beta\}$ ” is a trivial progressive BKZ+Pump in Two-step mode to estimate security as [29] stated. “ S_{op} ” is a progressive PnJBKZ+Pump with the optimized strategy selected by PSSearch in Two-step mode to estimate security. $\Delta \log_2 G$ is the difference between “Previous” and “Two-step” under the RAM model in strategy S_0 and S_{op} in the logarithm of gate count with base 2. The gate count of all estimations in this Table uses the same improved list-decoding technique proposed by MATZOV [38].

Under the RAM model, the estimated security bit of LWE in NIST schemes [52] can be reduced by 3.4~4.6 bit compared to the estimation generated by Leaky-LWE-Estimator⁷ in [53] under gate-count model which adopts the improved list-decoding technique proposed in [38]. It fixed the estimate done in [61] of the list-decoding technique proposed in [18]. See Table 6 for details. Here G and B in Table 6 respectively represent the total number of logic circuits and the maximum memory needed for solving these LWE instances in NIST schemes [52] being solved, that both are calculated under same gate-count model.

⁶ <https://github.com/Summwer/lwe-estimator-with-PnJBKZ.git>

⁷ <https://github.com/lucas/leaky-LWE-Estimator>

F Application to SVP

The optimization of the solving strategy can be applied not only to solving LWE but also to solving SVP. The only difference is replacing the success condition for solving u-SVP from Section B with the success condition for solving SVP proposed in [26]. This section is dedicated to demonstrating the efficiency of the `MinTwoStepSolver` in solving SVP instances in practice. Here

$$\text{MinTwoStepSolver} = \text{TwoStepSolver} + \text{PSSearch} + \mathcal{T}$$

is an algorithm that utilizes `TwoStepSolver` in conjunction with a strategy generated by `PSSearch`, all under a specific time cost model \mathcal{T} , to effectively solve SVP instances. Sec. 6.1 presents verification experiments for the `PnJBKZ` simulator, which serves as a fundamental component of `PSSearch`. The executable code of `TwoStepSolver` for solving SVP based on `PSSearch` is publicly available on GitHub³.

G Practical time cost model of Pump and PnJBKZ

To find the progressive blocksize and jump size selection strategy with minimal expected time cost for solving TU Darmstadt LWE challenges, it is necessary to construct `PnJBKZ` and `Pump` time cost models. However, the asymptotic complexity of the sieving does not match the actual cost well in the low-dimensional case⁶ (dimension ≤ 128). The multi-threading technology used in `Pump` will balance part of the time cost increases when the dimension of sieving increases. Therefore, we construct a practical time cost model by using the experimental method to test the running time of the `Pump` in Appendix G.1 on a different lattice basis for finding the optimized reduction parameters of solving TU Darmstadt LWE challenges in shorter time cost.

Although the time-cost model based on the results of experiments can well fit the actual cost of running `PnJBKZ`, using testing machines with different configurations will inevitably lead to changes in the time-cost model in low-dimensional cases. Therefore, we only use this experimentally constructed time-cost model when looking for the optimized progressive blocksize and jump size selection strategy for solving LWE challenges in shorter time cost.

³ <https://github.com/Summwer/lwe-estimator-with-pnjbkz>

⁶ While dimension exceeds 128, the time cost for `Pump` fits the theoretical value well, we can directly use the time cost model of `triple_gpu` sieve declared in [28].

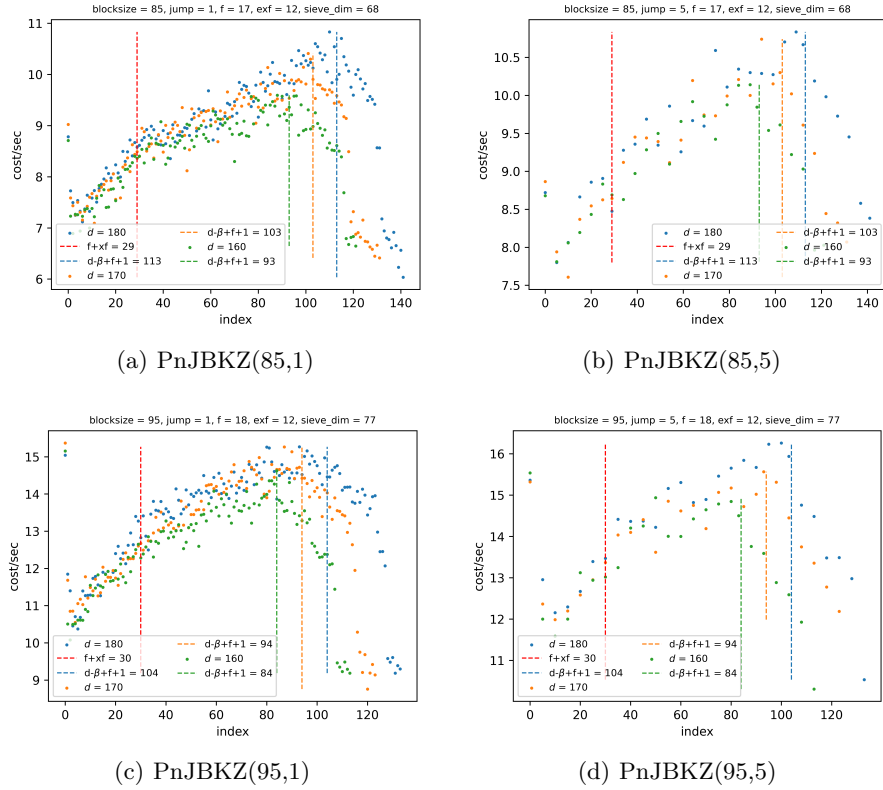


Fig. 24: Cost for each Pump under different index in a PnJBKZ tour by testing SVP Challenge with different dimension d using Machine C with threads = 32 and GPUs = 2.

Besides, when we construct the actual time cost model by testing the time cost of PnJBKZ on the specific machine, we find that each Pump in PnJBKZ takes a different time cost as Fig. 24 shown. Especially, the time cost of the first Pump is higher than subsequent Pumps and it increases under the incremental index from 2^{nd} to $(d - \beta + f + 1)^{\text{th}}$ and decreases after $d - \beta + f + 1$ indices. It infers that for a fixed blocksize β , the average Pump cost in PnJBKZ will increase with the growth of dimension d . It means that the simplified model of treating each SVP oracle inside BKZ as having the same time cost no longer applies in the context of PnJBKZ. Therefore, in Appendix G.2, we propose a new time cost model for PnJBKZ to more accurately reflect its time cost performance in practical applications.

G.1 Practical Cost Model of Pump

We can regard T_{Pump} as a computational cost model of the d_{svp} -dimensional progressive sieve. T_{Pump} in [20] is considered as

$$\begin{aligned} T_{\text{Pump}}(d_{\text{svp}}) &= \sum_{j=\beta_0}^{d_{\text{svp}}} T_{\text{sieve}}(j) = \sum_{j=\beta_0}^{d_{\text{svp}}} 2^{c \cdot j + o(j)} = 2^{c\beta_0} \left(1 + 2^c + \dots + 2^{c(d_{\text{svp}} - \beta_0)} \right) \\ &\leq 2^{c\beta_0} \cdot \frac{2^{c(d_{\text{svp}}+1) + o(d_{\text{svp}}+1)}}{1 - 2^c} = O\left(2^{cd_{\text{svp}}}\right) \approx 2^{cd_{\text{svp}} + c_1}, \end{aligned} \quad (12)$$

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

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

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 with different sieving dimensions on the projected lattice bases of an 180-dimensional SVP Challenge⁸ and with different block sizes β s. The experimental results show that our computational cost model above can fit well with the actual cost of Pump.

Take β as the independent variable, $\log_2(T_{\text{Pump}})$ can be 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 the 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 25, we can see that R^2 is close to 1. It means that the fitting effect is good. Figure 25 also shows that the logarithm of the computational cost of Pump is linearly correlated to d_{svp} under both float type “dd” and “qd”. Since the “qd” float type is more precise than “dd”, it is slower than “dd”. So we suggest setting “dd” float type.

Consider Cost of SimHash Generation. Moreover, given sieve dimension β , the G6K (or its GPU version) implementation requires $O(\beta)$ memory and $O(\beta)$ computational cost to generate the SimHash value, which is used to find the nearest neighbor of each vector. Thus, an $O(2^{c\beta})$ -time and $O(2^{c_2\beta})$ -space algorithm actually requires $O(2^{c\beta} + \beta * 2^{c_2\beta})$. Set $c = 0.367$ and $c_2 = 0.2075$

⁸ <https://www.latticechallenge.org/svp-challenge>

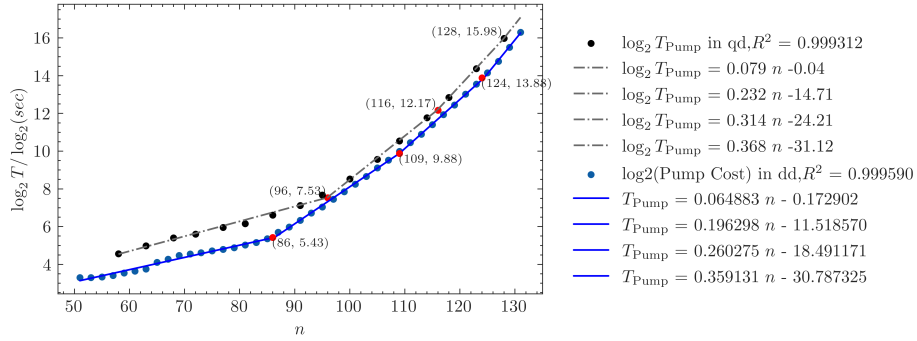


Fig. 25: Pump Cost Figure while $d = 180$, Sieve used in Pump is `gpu_sieve`, and it's running on Machine C with 2 GPUs and 32 threads: Relation between $\log_2(T_{\text{Pump}})$ and sieve dimension n .

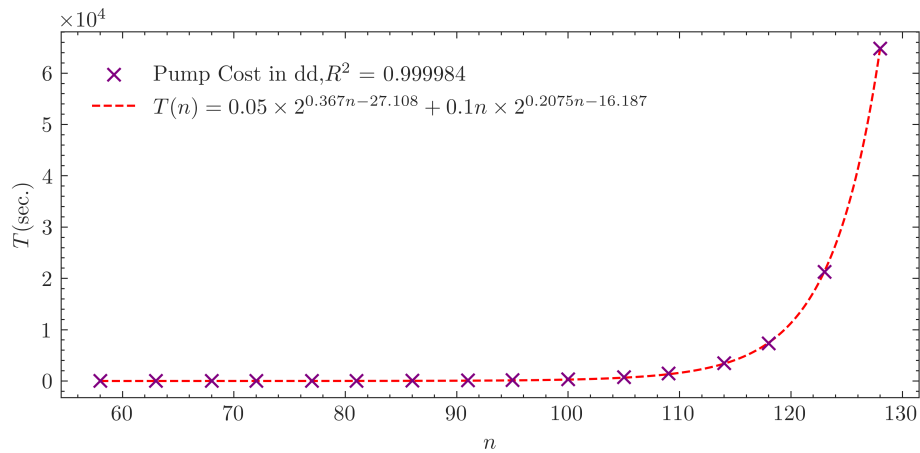


Fig. 26: Pump cost model considers the cost to generate the hash value: The vertical axis represents T_{Pump} and the horizontal axis represents the sieve dimension n .

according to Fig. 7 in [28] and construct the practical Pump model as

$$T_{\text{Pump}}(\beta) = a_1 \cdot 2^{c\beta+c_1} + a_2 \cdot 2^{c_2\beta+c_3},$$

then we can obtain the practical Pump cost model (as Fig. 26 shown) through the curve fitting method.

G.2 Practical Cost Model of PnJBKZ

PnJBKZ consists of a series of Pumps. If we regard PnJBKZ as a combination of Pumps with equal cost, the computational cost of PnJBKZ can be calculated by the sum cost of $\frac{d+2f-\beta}{J}$ progressive sieves on the $(\beta - f)$ -dimension projected sublattice with jump J . However, as Fig. 24 shows, each Pump in PnJBKZ has a different cost. Especially, the Pump cost increases from the 2nd to the $(d - \beta + f + 1)$ th index and decreases afterward. Here, in Figure 24, we can observe that the growth rate in the range of $[0, f + f_{\text{extra}}]$ differs from that of $[f + f_{\text{extra}}, d - \beta + f + 1]$, where f_{extra} represents the extra dimension-for-free value set in G6K to enhance the efficiency of PnJBKZ and it is 12 in the default setting. So we depart the PnJBKZ cost into 4 parts: first index of Pump, preceding indices in range of $[0, f + f_{\text{extra}})$, middle indices in range of $[f + f_{\text{extra}}, d - \beta + f + 1)$ and later indices in range of $[d - \beta + f + 1, d)$. Let the cost of each range be T_{first} , T_{pre} , T_{mid} and T_{former} . Let $T_{f+f_{\text{extra}}}$ and $T_{d-\beta+f+1}$ be the Pump cost at the index $f + f_{\text{extra}}$ and $T_{d-\beta+f+1}$. We have tested T_{first} , T_{pre} , T_{later} and the coefficients A and B in $T_{\text{mid}}(d, \beta, J, f, f_{\text{extra}}) = \frac{T_{f+f_{\text{extra}}} + T_{d-\beta+f+1}}{2} \cdot (d - \beta - f_{\text{extra}} + 1) = \frac{(A \cdot (f + f_{\text{extra}}) + B) + (A \cdot (d - \beta + f) + B)}{2} \cdot (d - \beta - f_{\text{extra}} + 1)$ in dimension $d = 180$, jump $J = 1$ and “dd” float type, then we obtain the simulated cost model as Fig. 27. Then, we can get that

$$\begin{aligned} T_{\text{PnJBKZ}}(d, \beta, J, f, f_{\text{extra}}) &= T_{\text{first}} + T_{\text{pre}} \cdot \frac{\lceil \frac{f+f_{\text{extra}}}{J} \rceil - 1}{f + f_{\text{extra}} - 1} \\ &\quad + T_{\text{mid}}(d, \beta, J, f, f_{\text{extra}}) \cdot \frac{\lceil \frac{f+f_{\text{extra}}}{J} \rceil}{f + f_{\text{extra}}} \\ &\quad + T_{\text{later}} \cdot \frac{\lceil \frac{d-\beta-f_{\text{extra}}}{J} \rceil + 1}{d - \beta - f_{\text{extra}} + 1}, \end{aligned} \quad (13)$$

where f is the dimension for free value of β .

We’ve also used the Eq. 13 to simulate the PnJBKZ cost of other dimensions (such as $d = 160, 170$) with blocksize from 51 to 119 and jump $J \geq 1$, and find it fits well in simulation as Figure 28.

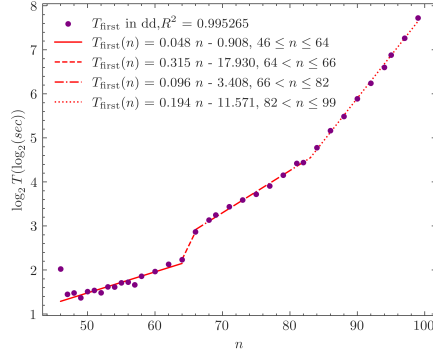
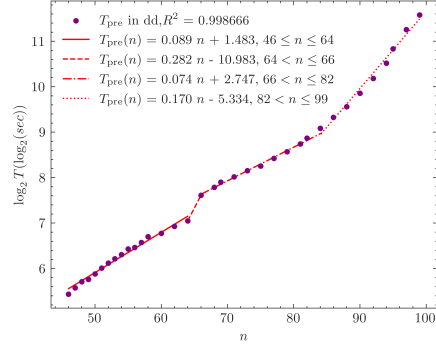
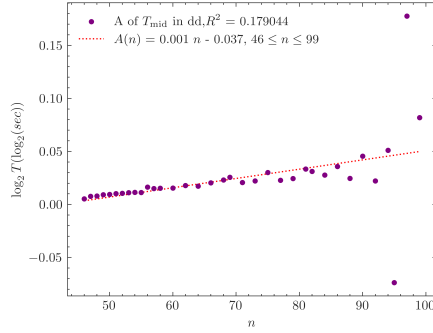
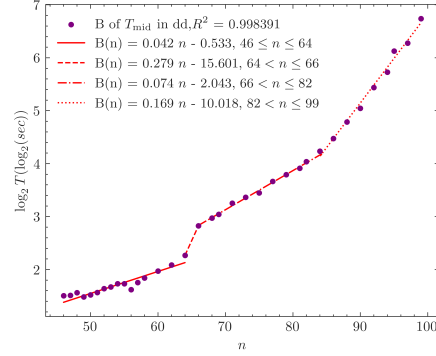
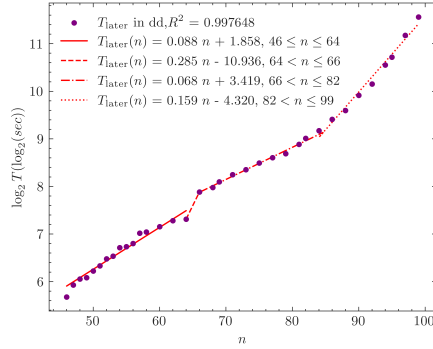
(a) T_{first} (b) T_{pre} (c) A of T_{mid} (d) B of T_{mid} (e) T_{later}

Fig. 27: Simulate T_{first} , T_{pre} , coefficients A and B , and T_{later} using the lattice basis generated from SVP Challenge with dimension $d = 180$. We test PnJBKZ with different β and setting $J = 1$, using f and f_{extra} setting in the G6K GPU version. We test the cost data on machine C with $\text{GPUs} = 2$ and $\text{threads} = 32$. The x-axis represents the index i of each Pump in a PnJBKZ tour, while the y-axis represents the time cost (in seconds) of PnJBKZ.

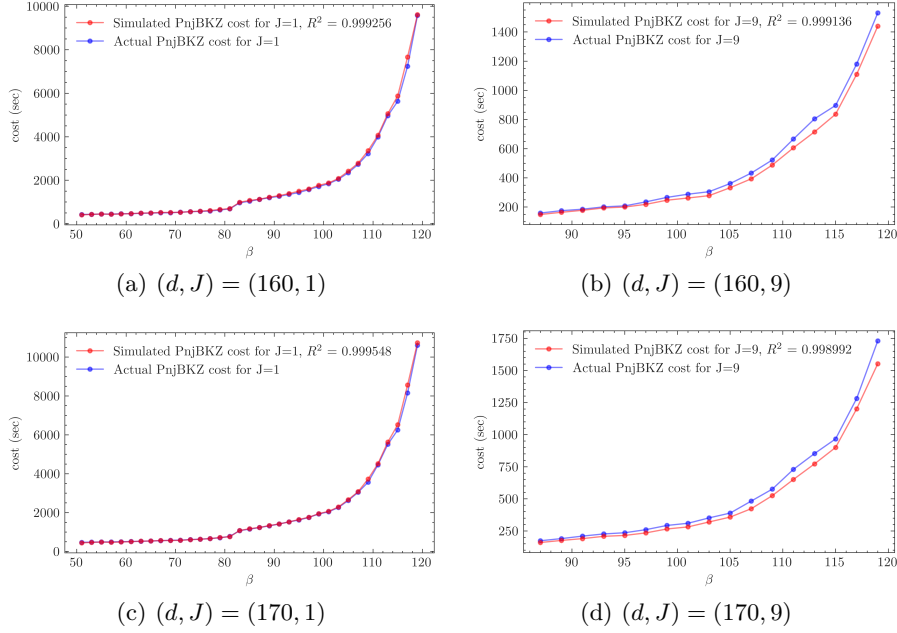


Fig. 28: Simulate each PnJBKZ Cost using Eq. (27 in $(d, J) \in \{(160, 1), (160, 9), (170, 1), (170, 9)\}$. The actual PnJBKZ cost is tested in machine C with GPUs = 2 and threads = 32. The test lattice basis is generated from the SVP Challenge with different dimensions d . We test PnJBKZ with different β and J , using f and f_{extra} settings in the G6K GPU version. The x-axis represents the blocksize β for PnJBKZ, while the y-axis represents the time cost (in seconds) of PnJBKZ.

H Blocksize and Jump Strategy Selection based on ProBKZ

The *blocksize and jump strategy selection algorithm based on ProBKZ* (BSSA, Fig. 29) applies the *Shortest Path Algorithm* to strategy selection.

BSSA initiates with a fully BKZ- β^{start} reduced lattice basis. It try to find the shortest path from BKZ- β^{start} to BKZ- β^{goal} reduced lattice basis by setting several middle nodes (such as $\beta^{\text{sstart}} = \beta_i$, for $\beta^{\text{start}} < \beta_i < \beta^{\text{goal}}$) from β^{start} to β^{goal} as a measure of basis quality. For edges between nodes β_i and β_j , BSSA determines the tuple $(\beta^{\text{alg}}, J^{\text{alg}}, t)$ that minimizes the simulated time cost T_{PnJBKZ} to reduce a BKZ- β_i basis to a BKZ- β_j basis, where $\beta_i < \beta^{\text{alg}} \leq d$.

For each node, we define a blocksize and jump strategy dictionary $\text{BS}[\beta^{\text{goal}}]$, in which the key is each middle node β_i and the value is a tuple of $\text{bs} = (\text{rr}, \text{S}, T_{\text{PnJBKZs}}, \text{PSC})$, where rr is the length of Gram-Schmidt vector which is fully BKZ- β^{goal} reduced, S means the blocksize and jump selection strategy which will improve the quality of lattice basis from fully BKZ- β^{start} reduced to fully

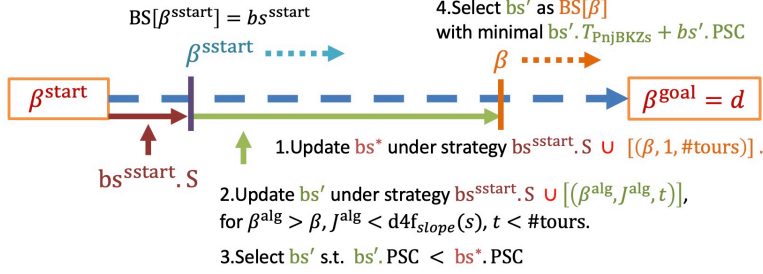


Fig. 29: BSSA Process.

BKZ- β^{goal} reduced, which is the combination of $(\beta^{\text{alg}}, J^{\text{alg}}, t, T_{\text{PnJBKZ}})$ stored on each edge in the shortest path from node β^{start} to β^{goal} with respect to the sum of simulated BKZ cost $T_{\text{PnJBKZs}} = \sum_{\beta^{\text{alg}}, J^{\text{alg}}, t} T_{\text{PnJBKZ}}(\beta^{\text{alg}}, J^{\text{alg}}, t)$, while the shortest path can be found using Dijkstra algorithm. PSC is one of the output from the Pump dimension estimation method (Alg. 5), which means the estimated time cost for uSVP_γ to be solved by processing Pump on the BKZ- β^{goal} reduced basis.

By setting different final β^{goal} , we can get different reduction strategy BS that improves the quality of lattice basis from β^{start} to β^{goal} and different sieving dimension of the last Pump corresponding to the different quality of the lattice that is fully β^{goal} reduced. Then we set multiple different final β^{goal} to choose the Two-step solving strategy whose total time cost is minimum. Here, the total time cost includes the time cost of improving the quality of lattice by a series of $\text{PnJBKZ}(\beta, J) \in \mathcal{S}$ and the time cost of final Pump. See Alg. 6 for more details about BSSA.

I Choosing the number of LWE Samples

BKZ-only mode is the mainstream method for estimating the security of an LWE-based cryptosystem at the current. It uses Kannan’s Embedding technique to reduce the LWE problem to the uSVP_γ 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 [33] and has been proved the correctness in [62], which has both given a lower bound of LWE samples and a blocksize β . We renamed it “2016 Estimation from GSA for LWE” (referred to as 2016 Estimate).

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 the 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

$$\min_{\beta \in \mathbb{N}} \left\{ T_{\text{BKZ}}(\beta) : \sigma\sqrt{\beta} \leq \delta(\beta)^{2\beta-d-1} \cdot q^{\frac{d-n-1}{d}} \right\}. \quad (14)$$

The strategy in the 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.

```

input :  $rr_0, F(\star, \mathcal{D}), \beta^{\text{start}} \leftarrow 50, J_{\text{max}}(\star) \leftarrow d4f(\star)/2;$ 
output:  $T_{\text{min}}, S_{\text{min}};$ 
1 Function BSSA( $rr_0, F(\star, \mathcal{D}), \beta^{\text{start}} \leftarrow 50$ ):
2    $d \leftarrow \text{len}(rr_0); \text{PSC}^{(0)} \leftarrow \text{ProSieveDimEst}(rr_0, F(\star, \mathcal{D}));$ 
    $\text{BS}[\beta^{\text{start}}] = (rr_0, [], 0, \text{PSC}^{(0)});$ 
3   for  $\beta \leftarrow \beta^{\text{start}}$  to  $d$  do
4      $T_{\text{PnJBKZs}}^{(\text{min})} \leftarrow +\infty;$ 
5     for  $\beta^{\text{sstart}} \leftarrow \beta^{\text{start}}$  to  $\beta - 1$  do
6        $\text{bs}^{\text{sstart}} \leftarrow \text{BS}[\beta^{\text{sstart}}]; \text{bs} \leftarrow (\emptyset, \emptyset, +\infty, +\infty);$ 
7       Update  $\text{bs}^*$  under strategy
        $\text{bs}^{\text{sstart}}.S \cup [(\beta, 1, \#\text{tours}(\text{bs}^{\text{sstart}}.rr, \text{BKZ-}\beta))];$ 
8       for  $\beta^{\text{alg}} \leftarrow \beta + 1$  to  $d$  do
9         for  $j \leftarrow J_{\text{max}}(\beta^{\text{alg}})$  to  $1$  do
10            $T' \leftarrow +\infty;$ 
11           for  $t \leftarrow 1$  to  $\#\text{tours}(\text{bs}^{\text{sstart}}.rr, \text{PnJBKZ-}(\beta^{\text{alg}}, j))$  do
12             Update  $\text{bs}'$  under strategy  $\text{bs}^{\text{sstart}}.S \cup [(\beta^{\text{alg}}, j, t)];$ 
13             if  $\text{bs}'.\text{PSC} < \text{bs}^*.\text{PSC}$  then
14                $T' \leftarrow \text{bs}'.T_{\text{PnJBKZs}};$ 
15               break;
16             if  $\text{bs}.T_{\text{PnJBKZs}} > T'$  then
17                $\text{bs} \leftarrow \text{bs}';$ 
18           if  $T_{\text{PnJBKZs}}^{(\text{min})} > \text{bs}.T_{\text{PnJBKZs}}$  then
19              $T_{\text{PnJBKZs}}^{(\text{min})} \leftarrow \text{bs}.T_{\text{PnJBKZs}}; \text{BS}[\beta] \leftarrow \text{bs};$ 
20    $\text{bs}_{\text{min}} \leftarrow \min_{\text{bs}.T_{\text{PnJBKZs}} + \text{bs}.\text{PSC}} \text{BS};$ 
21   return  $T_{\text{min}} \leftarrow \text{bs}.T_{\text{PnJBKZs}} + \text{bs}.\text{PSC}, S_{\text{min}} \leftarrow \text{bs}_{\text{min}}.S;$ 

```

Algorithm 6: BSSA

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 PnJBKZ with a fixed blocksize β simulated from GSA assumption and a progressive sieve algorithm in dimension d_{svp} . It simulates the above scenario and tries to find the minimal cost of (β, d_{svp}) from

$$\min_{\beta, d_{\text{svp}} \in \mathbb{N}} \left\{ 2 \cdot T_{\text{BKZ}}(\beta) + \text{PSC}(d_{\text{svp}}) : \|\pi_{d-d_{\text{svp}}}(\mathbf{v})\| \leq \text{GH}(\mathcal{L}_{\pi[d-d_{\text{svp}}]}) \right\}, \quad (15)$$

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

However, we have explained in Sec. 4.3 that the 2016 Estimate still has a probability of failing to find the target vector through its estimation. Thus, our

strategy for solving the LWE problem considers simulating a two-stage strategy using our PnJBKZ simulator and new Pump sieve dimension and PSC estimation scheme (as described in Alg. 5) In the first stage, it will call the PnJBKZ simulator to simulate the basis after a series of PnJBKZ. In the second stage, it tries to find the unique shortest vector by Pump. Based on the estimation scheme in the default G6K described above, we modify the time cost of two PnJBKZs and a progressive sieve to the time cost of serial PnJBKZs following the blocksize strategy and a progressive sieve. Besides, we use the new Pump estimation scheme to simulate the norm of the target vector.

Let $P(d_{\text{sVP}}) = \Pr \left[y \leftarrow \sigma^2 \chi_{d_{\text{sVP}}}^2 \mid y \leq (\text{GH}(\mathcal{L}_{\pi[d-d_{\text{sVP}}:d]})) \right]^2$. Thus, the formula becomes

$$\min_{\beta, d_{\text{sVP}} \in \mathbb{N}} \{T_{\text{PnJBKZs}}(\mathbf{B}) + \text{PSC}(d_{\text{sVP}}) : P(d_{\text{sVP}}) \geq P_{\text{success}}\}, \quad (16)$$

where δ is the basis quality after PnJBKZs. $T_{\text{PnJBKZs}}(\mathbf{B})$ will respectively call BSSA or 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 $\tau \in \mathbb{Z}^*$. We use dichotomization to find an m with minimum β and d_{sVP} satisfying the inequality (16). Furthermore, the concrete process is as the Algorithm 7.

```

input:  $n, q, \alpha, m_{\text{all}}, \beta_{\text{bound}}, d_{\text{bound}}^{(\text{sVP})}, \tau, \mathbf{A}^{m_{\text{all}} \times n}, \mathbf{b}^{m_{\text{all}} \times 1}$ ;
output:  $S_{\text{min}}, T_{\text{min}}, m$ ;
1  $\sigma, T_{\text{min}}, \mathbf{mRange} \leftarrow \alpha q, +\infty, \{\}$ ;
2  $m_0 \leftarrow$  LWE samples estimation in G6K as formula (15);
3  $m_{\text{min}} \leftarrow \min \{m \text{ satisfies equation (16)}\}$ ;  $S_{\text{min}}, T_{\text{min}} \leftarrow \text{None}, \text{None}$ ;
4 while  $\tau > 0$  do
5   Construct  $\mathbf{B}$  by  $(\mathbf{A}^{m_0 \times n}, \mathbf{b}^{m_0 \times 1}, q)$ ;
6    $m_1 \leftarrow m_0$ ;
7   for  $m \in \{\max\{m_{\text{min}}, m_0 - \tau\}, m_0, \min\{m_{\text{all}}, m_0 + \tau\}\}$  do
8      $d \leftarrow m + 1, M \leftarrow \sigma^2 m + 1$ ;
9     Construct  $\mathbf{B}$  by  $(\mathbf{A}^{m \times n}, \mathbf{b}^{m \times 1}, q)$ ;
10     $T_{\text{total}}, \mathbf{S} \leftarrow \text{EnumBS}(\text{rr}(\mathbf{B}), \sigma^2 \chi_{\star}^2)$ ;
11    if  $T_{\text{min}}$  is None or  $T_{\text{min}} < T_{\text{total}}$  then
12       $S_{\text{min}}, T_{\text{min}}, m_1 \leftarrow \mathbf{S}, T_{\text{total}}, m$ ;
13  if  $m_1 = m_0$  then
14     $\tau \leftarrow \lfloor \frac{\tau}{2} \rfloor$ ;
15   $m_0 \leftarrow m_1$ ;
16 return  $S_{\text{min}}, T_{\text{min}}, m_0$ ;

```

Algorithm 7: Our LWE Samples Number Selection Algorithm

Using the optimization strategy for LWE instance number selection, we can solve challenges faster than the G6K default strategy, although its efficiency improvement is not significant (at most 2.2% in the test). See the Table 7.

Table 7: LWE samples improvement simulated result generated by EnumBS with no RAM limit and $\tau = 10$.

(n, α)	G6K's m	Our m	Estimated T_{new} (sec)	Estimated T_{old} (sec)	$T_{\text{new}}/T_{\text{old}}$
(50,0.025)	219	221	4336037.42	4320454.232	99.6%
(55,0.020)	230	234	3937458.799	3870765.534	98.3%
(45,0.035)	210	220	74367286.54	73838336.19	99.3%
(45,0.030)	201	205	1420793.45	1404095.127	98.8%
(90,0.005)	306	316	1772710.1	1733158.312	97.8%

J The Optimized Strategy for the LWE Challenge

In Table 8, we give the optimized blocksize and jump strategy generated by EnumBS for solving TU Darmstadt LWE Challenge with

$$(n, \alpha) \in \{(40, 0.035), (40, 0.040), (50, 0.025), (55, 0.020), (90, 0.005)\}$$

successfully by running “implement_unsolved_lwechal.sh” in source code ².

K Comparison between simulated slope (cost) and real slope (cost) during reduction

In this part, we give the slope and cost comparison of two LWE Challenges under qd float type in Table 9, Table 10 and Table 11, which show the simulated slope and cost are close to the real slope and cost. They also indicate that our PnJBKZ simulator can already reflect how the average of the norms of Gram-Schmidt vectors change during the reduction of PnJBKZ(β, J) on different LWE lattice basis.

From Table 4, Table 9, Table 10 and Table 11 all show that although, at the first round of reduction, the gap between the slope value of simulated GS norms and the slope of real reduced GS norms is slightly bigger due to the influence of the q-ary vector in the initial LWE lattice basis, as the reduction proceeds, in the rounds of reduction before finally entering the Pump, the gap between the slope value calculated by simulation and the slope obtained by real reduction has been sufficiently small. For selecting the optimized blocksize and jump strategy, our PnJBKZ simulator is accurate enough.

Table 8: Blocksize and Jump strategy generated by EnumBS(threads = 10) using the practical cost model generated on Machine C with threads = 32 and GPUs = 2.

(n, α)	RAM limit	Strategy (β, jump)	EnumBSGen/s
(40,0.035)	1.5TB	[(72,9),(81,10),(102,11),(106,11), (117,12),(125,13),(133,12),(136,1)]	269.15
(40,0.040)	1.5TB	[(81, 10),(81, 10), (105, 11), (110, 12), (118, 11), (133, 12), (141, 10), (141, 1), (148, 1)]	289.17
(50,0.025)	1.5TB	[(77, 9), (81, 10), (102, 11), (102, 11), (105, 11),(115, 12), (119, 12), (127, 12), (132, 13), (140, 1), (148, 1)]	686.47
(55,0.020)	1.5TB	[(68, 9), (81, 10), (102, 11),(102, 11), (102, 11), (114, 12), (119, 12), (119, 9), (131, 13), (137, 12),(140, 1), (147, 1)]	831.98
(90,0.005)	512GB	[(68, 9), (81, 10), (81, 10), (81, 10), (102, 11), (102, 11), (102, 11), (102, 11), (104, 11), (114, 12), (119, 12),(119, 12), (119, 9), (127, 13), (129, 12), (133, 12), (133, 12), (141, 1),(141, 1)]	2592.26

(β, J)	Simulation Slope log(T)		Practical Slope log(T)	
(56,8)	-0.0307	6.2	-0.0297	6.4
(66,9)	-0.0279	6.2	-0.0273	6.4
(80,10)	-0.0254	6.5	-0.0250	6.8
(81,10)	-0.0238	6.6	-0.0237	6.9
(102,11)	-0.0215	7.8	-0.0216	8.1
(102,11,2)	-0.0205	7.8	-0.0208	8.1

Table 9: Quality and wall time (T in seconds) during reduction of LWE Challenge $(n, \alpha) = (45, 0.020)$.

Table 10: Quality and $\log(\text{walltime})$ ($\log(T)$ in seconds) during reduction of LWE Challenge $(n, \alpha) = (50, 0.015)$.

Table 11: Quality and $\log(\text{walltime})$ ($\log(T)$ in seconds) during reduction of LWE Challenge $(n, \alpha) = (40, 0.025)$.

(β, J)	Simulation Slope log(T)		Practical Slope log(T)	
(77,8)	-0.0281	6.5	-0.0265	6.6
(81,10)	-0.0249	6.2	-0.0241	6.6
(102,11)	-0.0217	7.5	-0.0215	7.8
(102,11)	-0.0205	7.5	-0.0207	7.8