Refined Strategy for Solving LWE in Two-step Mode

Wenwen Xia[1,2,4**], Leizhang Wang[3**], Geng Wang[1,2*], Dawu Gu[1*], and Baocang Wang[3*]

[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 the NIST standards Kyber and Dilithium. A refined estimation of LWE hardness is crucial for their security. Currently, the primal attack is considered the fastest method to solve LWE in practice. This method reduces the LWE problem to a unique Shortest Vector Problem (uSVP) and combines lattice reduction algorithms with a single SVP call, such as enumeration or sieve. However, finding the most time-efficient strategy for combining these algorithms remains a challenge. The Kyber designers highlighted this issue as open problem Q7: "A refined (progressive) lattice reduction strategy, along with a precise analysis of gains from reduction preprocessing plus a single SVP call in large dimensions, is still missing."

In this paper, we address this problem with a Strategy Search algorithm, named PSSearch, to solve both uSVP and LWE. PSSearch uses progressive BKZ as the lattice reduction and sieving as the SVP oracle. Compared to the heuristic strategy in G6K (Albrechet et al., Eurocrypt 2019), our algorithm offers the following advantages: (1) We design a pruning tree search algorithm, PSSearch, to find the minimum time-cost strategy in the Two-step mode and prove its correctness, showing that the fastest approach in the Two-step mode to solve uSVP and LWE can be achieved in a reasonable time-frame; (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 oracle.

We tested the accuracy of our new simulation algorithm and the efficiency of our new strategy through experiments. Furthermore, we applied the strategies generated by PSSearch to solve the TU Darmstadt LWE Challenges with $(n, \alpha) \in$ {(80, 0.005), (40, 0.035), (90, 0.005), (50,

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

0.025), (55, 0.020), (40, 0.040)} using the G6K framework, achieving improvements of 7.2 to 23.4 times over the heuristic strategy used in G6K. By combining the minimum 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 assess the impact 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. This results in a decrease of 1.1 to 1.3 bits compared to the refined Two-step estimation using a trivial strategy.

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

1 Introduction

Learning with Errors (LWE) [1] plays an crucial role in lattice-based cryptography, as the security of many lattice-based cryptographic schemes [2–5] depends on the hardness of LWE or its variants [1,6–8], including the NIST post-quantum standards Kyber and Dilithium [8,9]. Therefore, to provide concrete security levels for these schemes, a tight estimation of LWE hardness is necessary. This requires identifying the most efficient algorithm for solving LWE and determining its precise complexity.

Among the various methods for solving LWE, the primal attack [10] is currently considered the most efficient in practice. The primal attack uses Kannan's embedding technique [11] to transform LWE into the unique Shortest Vector Problem (uSVP) on a specific lattice, which has a unique shortest vector. After certain level of reduction on the lattice, the projection of the unique shortest vector becomes the shortest vector on a projected sublattice, we can recover it by first finding the shortest vector on the sublattice 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 since been used in many LWE estimators, such as [15].

The choice of lattice reduction and SVP solving algorithms is crucial in the LWE primal attack. In the literature, cryptanalysts commonly use BKZ [16] as the lattice reduction algorithm, which has a tunable parameter β called blocksize, allowing a balance between time cost and reduction effectiveness, and use either enumeration or lattice sieving as the SVP oracle. While efficiency could be further improved by developing better algorithms for lattice reduction or SVP solver, designing these fundamental algorithms seems to be extremely hard. However, even with improvements to these algorithms, optimization is still possible by selecting more effective solving strategy. Here, "strategies" refers to the different ways of calling the fundamental algorithms with various parameters.

All LWE solving algorithms and estimators use specific strategies, though some are presented implicitly. In BKZ 2.0 [17] and ADPS16 [18], several tours of BKZ with a fixed blocksize are called iteratively until a short vector is found, with no additional SVP call after the BKZ procedure. This approach was later

improved by a progressive strategy, where the blocksize is increased by each BKZ tour. Aono et al. introduced the Improved Progressive BKZ (ProBKZ) [19], which includes an explicit strategy selection algorithm that outputs 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 using enumeration. All of these algorithms use the enumeration as the SVP oracle, which has since been outperformed by the lattice sieve in higher dimensions.

In the LWE estimator designed by Albrecht et al. [15], the sieve is used as the SVP call. They presented two different modes for estimating the complexity of the LWE primal attack: one mode involves progressive BKZ with blocksize increment of 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, triggered heuristically. However, there is no guarantee that the sieving will yield a solution for LWE, as 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 posed an open problem Q7: "A refined (progressive) lattice reduction strategy, along with a precise analysis of the benefits from reduction preprocessing combined with a single SVP call in large dimensions is still missing." The designers anticipate that by optimizing the strategy for BKZ blocksizes and final SVP dimension, the security estimation for Kyber could potentially be decreased by 2 to 8 bits.

In this paper, we solve this open problem by presenting an minimum time-cost strategy for uSVP and LWE in the primal attack. First, we note that all possible solving strategies form a strategy space with a finite number of elements, meaning the minimum time-cost strategy can be found by searching over the strategy space. However, three main issues must be solved: (1) Provide a formal definition for the strategy space, ensuring it includes 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 search algorithm so that the minimum time-cost strategy can be found within a reasonable time frame. (3) To ensure the search algorithm functions effectively, we need to offer a precise time cost estimation for each strategy within the strategy space.

1.1 Our Contribution

We summarize our contributions in this work as follows:

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

2. We design a tree search algorithm for the strategy space with pruning, named Pruning Strategy Search (PSSearch). We prove that PSSearch can output the minimum time-cost Two-step strategy for solving LWE within the strategy space. As a result, PSSearch can be used to accelerate LWE solver and provide tight estimations for LWE-based cryptographic schemes.

3. We implement PSSearch using the most efficient lattice reduction and SVP algorithms available to date, namely PnJBKZ and Pump in G6K. Specifically, we design a simulator for PnJBKZ that can simulate behaviors for jump>1, enabling the search for more flexible and efficient reduction strategies. Additionally, we propose a refined dimension estimation method for the SVP oracle, taking into account the distribution of the LWE noise vector.

In particular, using the minimum time-cost Two-step strategy for solving LWE generated by PSSearch, we successfully cracked six previously unsolved TU Damstadt LWE Challenges[1], achieving a speedup of 23.4 times compared to the previously most efficient LWE solver, G6K [28]. For details, see Figs. 2, 6, and Table 4. In the reduction step, the time cost needed to achieve the same basis quality is 4 to 36.4 times faster than the trivial reduction strategy. For more details, see Fig. 7. The executable code of TwoStepSolver based on PSSearch, is publicly available on GitHub[2].

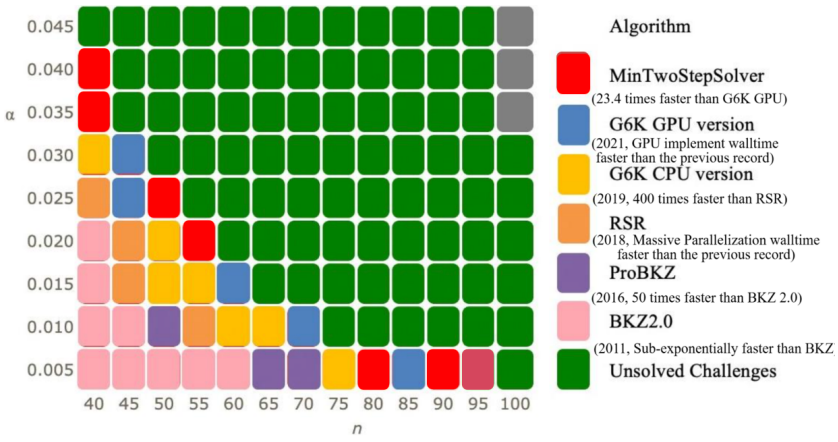


Fig. 2: LWE Challenges and the Algorithms to Solve it

Our new hardness estimation for the NIST standards Kyber and Dilithium shows that, when using the minimum time-cost strategy, the security levels are

[1] https://www.latticechallenge.org/lwe_challenge/challenge.php
[2] https://github.com/Summwer/pro-pnj-bkz

decreased by 3.4 to 4.6 bits, rather than 2 to 8 bits indicated in the Kyber documentation. This confirms the open problem Q7 in the Kyber document. Compared to the trivial strategy used in the "Refined Two-step Mode LWE Estimator" [29], the security levels decrease by 1.1 to 1.3 bits. For details, see Table 5.

1.2 Related Works

Lattice Solving Algorithms. BKZ [16] is the most popular lattice reduction algorithm for solving LWE. It combines the LLL [30] algorithm with an SVP solver to balance the time cost and the success probability using blocksize β. Many cryptanalysts have improved the BKZ algorithm, such as through 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, like *Improved Progressive BKZ* (ProBKZ) [19]. [14] demonstrated 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 for solving LWE as described in [33].

In 2019, Albrecht et al. [20] designed the *General Sieve Kernel* (G6K) and implemented the progressive sieve [27], named Pump, which selectively employs various sieving algorithms: the Gauss sieve [21, 22], the NV sieve [34], the k-list sieve [23, 24] or the BGJ1 sieve [25]. Pump utilizes the progressive sieve and dimension-for-free (d4f) technique [26] for acceleration. Specifically, Pump is both 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] further improved G6K with GPU acceleration (named G6K-GPU-Tensor) and implemented the fastest sieving algorithm, BDGL16 [18], in both G6K and G6K-GPU-Tensor. G6K offers an algorithm to solve LWE, conditionally calling Pump to find short vectors on the projected sublattice, which are then lifted into the full lattice basis after running several tours of PnJBKZ. PnJBKZ uses Pump as its SVP oracle, ensuring that even when the SVP oracle is called with a jump value, the skipped lattice basis vectors are still reduced by Pump. PnJBKZ solves TU Darmstadt LWE Challenges 400 times faster than the previous records.[1]

Lattice Reduction Simulators. The first BKZ simulator, refer to the BKZ 2.0 simulator, was proposed in [17] and uses the Gaussian heuristic to predict BKZ-β. In 2016, Aono et al. introduced a new BKZ simulator in [19] as part of their proposal for Progressive BKZ, aimed at predicting 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.

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

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 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], only one tour of BKZ reduction is run for each blocksize β. Additionally, the randomized simulator cannot accurately estimate the concrete security strength of cryptographic schemes when only one tour of BKZ reduction is run 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. The SSearch method is determined based on a given reduction simulator, named `ReductionSim`, for lattice reduction algorithm $\mathcal{R}$, a dimension estimation method named `SVPDimEst` for the SVP oracle $\mathcal{C}$ and a time-cost model. Next, in Sec. 4, we present a pruning version of SSearch (PSSearch) and prove its correctness in outputting the minimum time cost strategy for solving LWE. In Sec. 5, we propose a PnJBKZ simulator for instancing PSSearch with the lattice reduction algorithm PnJBKZ.

Finally, we give the experiments in Sec. 6 as follows, (1) accuracy verification of PnJBKZ simulator; (2) a compare the time cost of the LWE solver implemented in G6K GPU version with our Two-step LWE solver using the strategy generated by PSSearch and the practical time-cost model proposed in Sec. 2.6; (3) the optimized strategy for solving the LWE Challenge; (4) verification of the simulation accuracy of MinTwoStepSolver; (5) details of the new LWE records; and (6) a re-estimate NIST schemes Kyber and Dilithium based on PSSearch and gate count model [38].

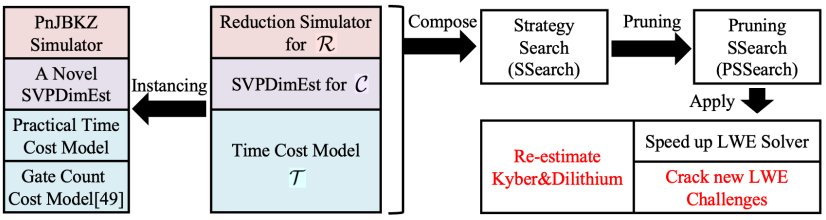


Fig. 3: Roadmap

2 Preliminaries

We denote vectors by lower-case bold letters, e.g. $\mathbf{a}, \mathbf{b}, \mathbf{c}, ...$, and matrices by upper-case bold letters, e.g. $\mathbf{A}, \mathbf{B}, \mathbf{C}, ...$ For a matrix $\mathbf{B} = (\mathbf{b}_0, \cdots, \mathbf{b}_{d-1})$, we refer to $\mathbf{b}_i$ 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 use $\langle \cdot, \cdot \rangle$ to denote inner products and $\cdot$ for matrix-vector products. By abuse of notation, we treat vectors as either row or column vectors depending on the context, so such that both $\mathbf{v} \cdot \mathbf{A}$ and $\mathbf{A} \cdot \mathbf{v}$ are meaningful. We denote $\mathbf{0}_{m \times d}$ as the $m \times d$ all-zero matrix. If the dimensions are clear from the context, we may omit the subscripts. $|\mathbf{B}|$ denotes the absolute value of the determinant of the matrix $\mathbf{B}$. $A := [\![a_0, ..., a_{n-1}]\!]$ represents an ordered list of size $\sharp A = n$ with $A[i] = a_i$. Additionally, we have $[\![a_0, ..., a_{n-1}]\!] \cup [\![a_n]\!] = [\![a_0, ..., a_{n-1}, a_n]\!]$. We define $[a, b] := \{a, a+1, ..., b-1\}$ 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 $\mathbf{B}_{[i,j]} := (\mathbf{b}_i, \cdots, \mathbf{b}_{j-1})$. We can also denote the lattice generated by basis $\mathbf{B}$ as $\mathcal{L}(\mathbf{B})$. Let $\mathbf{B}^* = (\mathbf{b}_0^*, \ldots, \mathbf{b}_{d-1}^*)$ 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 the Gram-Schmidt norm, i.e. $l_i = \ln(\|\mathbf{b}_i^*\|)$, for $i \in \{0, \cdots, d-1\}$. Let $\mathsf{rr}(\mathbf{B}) = (l_0, \cdots, l_{d-1})$, abbreviated as rr, $\mathsf{rr}_{[i,j]} = (l_i, \cdots, l_{j-1})$. For $i \in [d]$, we denote by π_i the orthogonal projection onto $(\mathbf{b}_0, ..., \mathbf{b}_{i-1})^\perp$. For $0 \leq j \leq k \leq d-1$, we denote by $\mathbf{B}_{\pi[j,k]}$ the local projected block $(\pi_j(\mathbf{b}_j), \pi_j(\mathbf{b}_{j+1}), ..., \pi_j(\mathbf{b}_{k-1}))$, and by $\mathcal{L}_{\pi[j,k]}$ the lattice spanned by $\mathbf{B}_{\pi[j,k]}$.

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

2.2 Lattice Reduction

Definition 1 (Size-reduced). *The Gram-Schmidt coefficients of a d-dimension lattice basis $\mathbf{B}$ are denoted by $\mu_{i,j}$ for any $0 \leq j < i \leq d-1$. The basis $\mathbf{B}$ is considered size-reduced if the following condition holds: For $0 \leq j < i \leq d-1$: $|\mu_{i,j}| \leq 1/2$.*

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 lattice $\mathcal{L}$ 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* $\delta(\mathbf{B}) = \left(\|\mathbf{b}_0\|/|\mathbf{B}|^{1/d}\right)^{1/d}$*. 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 the current quality of the lattice basis $\mathbf{B}$. A higher lattice basis quality of $\mathbf{B}$ corresponds to a smaller value of $\delta(\mathbf{B})$.

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

By Combining the GSA with the root Hermite factor (Definition 3) and noting that $\mathrm{Vol}(\mathcal{L}) = \prod_{i=0}^{d-1}\|\mathbf{b}_i^*\|$, we infer that $\alpha = \delta^{-\frac{2d}{d-1}} \approx \delta^{-2}$. Let s be the slope value of the logarithm of GS norms l_i, $\forall i \in \{0, \ldots, d-1\}$, $s \approx \ln\alpha$ and $\delta \approx e^{-\frac{s}{2}}$. Even if the GSA does not strictly hold, we can still estimate the slope s using least squares fitting on $\ln\|\mathbf{b}_i^*\|$. In earlier works, such as [20], s serves as a measure of the basis quality: the lattice basis quality improves as s approaches 0.

2.3 Lattice Sieving Algorithms and Progressive Sieve

Lattice sieving algorithms are used to solve the Shortest Vector Problem (SVP) by iteratively combining vectors to find shorter ones. The first practical lattice sieve, the NV sieve, used a database of $N_0 = O(2^{0.2075d})$ vectors and ran in $O(2^{0.415d})$ time [34]. Later improvements, focusing on nearest neighbor search techniques, reduced the time complexity to $O(2^{0.292d})$ [25, 41, 42]. Rather than checking all pairs, a bucketing strategy was introduced, where close vectors are more likely to fall into the same bucket. This reduces the number of pairs that need to be checked. To further reduce memory usage, triplets of vectors can be considered, leading to a time-memory trade-off. The current best triple sieve (denoted as hk3-sieve) minimizes memory to $O(2^{0.1887d})$ and has a time complexity of $O(2^{0.3588d})$ [24].

Progressive sieve [27] is a variant of lattice sieving that performs sieving as the dimension increases, rather than directly sieving on a full-dimensional lattice at once. The original right-progressive sieve extends the sieving process from $\mathbf{B}_{[0,n]}$ to $\mathbf{B}_{[0,n+1]}$, gradually increasing the dimension n. In contrast, the left-progressive sieve [20, 28] first performs sieving in the projected sublattice basis $\mathbf{B}_{\pi[d-n,d]}$, then lifts the vectors using Babai's nearest plane algorithm [12] to $\mathbf{B}_{\pi[d-n-1,d]}$ before continuing the sieving process. In this paper, the term "progressive sieve" refers specifically to the left-progressive variant. It repeats this process until the short vectors are lifted onto the full-dimensional lattice.

2.4 Technologies in G6K

G6K [20] is an abstract machine designed to run sieve and reduction algorithms, built by generalizing and extending the previous sieving algorithms. G6K-GPU-Tensor [28], a state-of-the-art SVP solver, improves the efficiency of G6K through

GPU implementations. It holds many records in the TU Darmstadt SVP Challenges, outperforming prior records by at least 400 times.

Dimension for Free (d4f) Technique. D4f technology [26] can provide sub-exponential time speedup and a reduction in memory for sieving algorithms. [26] provides two theoretical d4f estimations for solving the β-dimension SVP: $\mathrm{d4f}(\beta) = \beta\ln(4/3)/\ln(\beta/2\pi)$ and $\mathrm{d4f}(\beta) = \beta\ln(4/3)/\ln(\beta/2\pi e)$, while in the implementation of G6K [20], a more relaxed value, referred to as the "optimistic d4f," is used. β<40, $\mathrm{d4f_{op}}(\beta) = 0$; $40 \leq \beta \leq 75$, $\mathrm{d4f_{op}}(\beta) = \lfloor \beta - 40/2 \rfloor$; β>75, $\mathrm{d4f_{op}}(\beta) = \lfloor 11.5 + 0.075\beta \rfloor$.

However, the d4f estimations in G6K are only related to β, which is considered an overestimate. [43] proposed a refined d4f value estimation function based on the quality of the current lattice basis and proved its correctness under the GSA. This function shows that $\mathrm{d4f}_\delta = \ln_\delta \sqrt{4/3} \approx -\ln(4/3)/s$. Here, s represents the slope value of the logarithm of the Gram-Schmidt norms. More details about $\mathrm{d4f_{slope}}(s)$ can be found in Eq. (5) in [43].

Pump in G6K. Albrecht et al. proposed a sieving style algorithm, Pump, in [20], which combines *Progressive Sieve* [27] with the d4f technique [26] and a novel insertion trick. The Pump algorithm takes four input parameters: the lattice basis $\mathbf{B}$, the left insertion bound κ, the insertion upper bound d_{svp} and the d4f value $\mathrm{d4f}(d_{\mathrm{svp}})$. Here, κ+d_{svp}=d and the upper bound of sieve dimension is $d_{\mathrm{svp}} - \mathrm{d4f}(d_{\mathrm{svp}})$. The Pump will insert up to $d_{\mathrm{svp}} - \mathrm{d4f}(d_{\mathrm{svp}})$ short vectors into the basis index from κ to d, choosing the shortest vectors from the sieved set on the projected sublattice $\mathcal{L}_{\pi[d-d_{\mathrm{svp}}+i,d]}$ for i from κ to d. In this way, it performs a partial HKZ reduction and outputs a nearly HKZ-reduced basis, as described in the G6K paper [20].

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

2.5 Primal Attack and SVP Dimension Estimation in Search Step

For a standard LWE instance $(\mathbf{A}, \mathbf{b}) \in \mathbb{Z}_q^{m\times n} \times \mathbb{Z}_q^m$, where the error vector $\mathbf{e}$ has standard deviation σ, we assume that each element in $\mathbf{e}$ follows a discrete Gaussian distribution $\mathcal{D}_{0,\sigma}$ and that $m > n$. The *primal attack* [44] 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 $\mathbf{t} = (\mathbf{e}, 1)$ is an unique shortest vector in $\mathbf{B}$ and it is also the target vector of the uSVP.

In the search step of the Two-step method for solving LWE, an individual SVP oracle is invoked after lattice reduction to find the target solution for LWE.

Meanwhile, given a lattice basis $\mathbf{B}$ that has been reduced by using a lattice reduction algorithm, one can estimate the dimension of the SVP that needs to be solved in the search step to ultimately solve the LWE problem. By considering the norm of target vector $\mathbf{t}$ of LWE and its expected value, one can determine the minimum $d_{\text{svp}} \in [1, d]$ such that $\sigma\sqrt{d_{\text{svp}}} < \text{GH}\left(\mathbf{B}_{\pi[d-d_{\text{svp}},d]}\right)$. According to the success conditions for solving LWE [33], as verified by [10], the minimum dimension of the SVP that must be solved in search step to solve LWE is d_{svp}. Most efficient LWE solvers, such as those described in [20, 28], utilize this estimation in their implementations.

In this paper, we build upon the probabilistic model using BKZ simulation proposed in [45]. Specifically, we adapt the BKZ-based estimator to fit the SVP model, as detailed in Section 4.3, resulting in a more tailored dimension estimation for the SVP oracle. We use $\texttt{SVPDimEst}(\mathsf{rr}, \sigma)$ to denote the dimension estimation method for determining the required SVP dimension in the search step when solving LWE. This estimator takes as input the Gram-Schmidt norms rr of the 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 critical to the effectiveness of our strategy search algorithm. If enumeration is used as the SVP oracle in the lattice reduction algorithm, we can apply the time-cost model proposed in [46, 47]. However, if sieving is the SVP-solving algorithm in the lattice reduction process, the time-cost model becomes more complex. In this paper, we primarily focus on the time cost of the progressive sieve, specifically Pump with hk3-sieve, as it is currently the most efficient practical method for solving high-dimensional SVP.

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

For practical time cost, we note that Pump implementation in G6K-GPU-Tensor (resp. G6K) incurs additional overhead, requiring an extra time cost of $O(\beta)$ times of memory cost to generate DualHash (resp. SimHash) values. These hash values are used to find the nearest neighbors of each vector. Thus, while the theoretical time cost of Pump is $O(2^{c\beta})$ and its space cost is $O(2^{c_2\beta})$, the actual time complexity is $O(2^{c\beta} + \beta\cdot 2^{c_2\beta})$. By setting $c = 0.367$ and $c_2 = 0.2075$ according to Fig. 7 in [28], we construct a practical time-cost model for Pump as $T_{\text{Pump}}(\beta) = a_1\cdot 2^{c\beta+c_1} + a_2\cdot 2^{c_2\beta+c_3}$. The coefficients for this model are determined through curve fitting, as shown in Fig. 12. More details on the practical Pump cost model can be found in Appendix B.2.

The time complexity model for lattice basis reduction algorithms, such as BKZ, is given by $T_{\text{BKZ}}(\beta) = (d-\beta)T_{\text{SVP}}(\beta)$. For PnJBKZ, this can be expressed

as $T_{\mathrm{BKZ}}(\beta, J) = (d-\beta)/J \cdot T_{\mathrm{Pump}}(\beta)$. However, experimental results have shown that this theoretical model diverges significantly from the actual time complexity model, as the time cost of the Pump increases with the index. Therefore, we present a more accurate and practical time complexity model for PnJBKZ, which is detailed in Appendix B.2.

3 Strategy Space and the Strategy Search Algorithm

In this section, we first formalize the definitions related to the strategy space in Sec. 3, then propose a Simplified Strategy Search method (SSearch) in Sec. 3.2, which is designed to search the minimum time-cost strategy in the Two-step mode.

3.1 Strategy Space

Before introducing the strategy space, it is important to first clarify the definition of lattice reduction algorithms. Most lattice reduction algorithms are designed as combinations of multiple SVP solver calls, with the primary goal of improving the quality of the lattice basis. This is achieved 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 $\mathcal{R}$-ξ)). *Reduce the lattice basis* $\mathbf{B}$ *using a blocksize list* $\xi = [\![(\kappa_0, \beta_0), \cdots, (\kappa_{n-1}, \beta_{n-1})]\!]$, *where* $\kappa_i, n \in [d-1]$, *and* $\beta_i \in [1, d-\kappa_i+1]$. *For* $i \in [n]$, *define* $\beta'_i = \min\{\beta_i, d-\kappa_i\}$, *and then iteratively apply the following two operations:*

1. *Find the shortest vector* $\mathbf{v}$ *by running a* β'_i*-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* $(\mathbf{b}_0, \cdots, \mathbf{b}_{\kappa_i-1}, \mathbf{v}, \mathbf{b}_{\kappa_i}, \cdots, \mathbf{b}_{d-1})$.

We present the initialization $\mathcal{R}$-ξ for the most commonly used lattice reduction algorithms in practice, as shown in Table 1. Each specific reduction algorithm is associated with particular parameters, which 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 $\xi = [\![(\kappa_0, \beta_0), \cdots, (\kappa_{n-1}, \beta_{n-1})]\!]$
BKZ [16]	β	$n = d-2,\ \kappa_i = i,\ \beta_i = \min\{\beta, d-i\},\ i \in [d-1]$
PnJBKZ [20]	(β, J)	$n = d-2,\ \kappa_i = i,\ \beta_i = \min\{\beta - (i \mod J), d-i\},\ i \in [d-1]$
HKZ [11, 48]	-	$n = d-2,\ \kappa_i = i,\ \beta_i = d-i,\ i \in [d-1]$

Lattice Basis Quality. There are various ways to characterize the quality of a lattice basis. In the context of BKZ reduction, where the GSA holds, the quality of the reduced basis can be described by the slope of $(\ln(\|\mathbf{b}_0^*\|), ..., \ln(\|\mathbf{b}_{d-1}^*\|))$. However, when evaluating the quality of a lattice basis after a tour of PnJBKZ or another reduction algorithm, the GSA may no longer holds. In this paper, we provide a general definition of lattice basis quality that encompasses the reductions described in Def. 4.

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

Moreover, for two different bases $\mathbf{C}$ *and* $\mathbf{D}$ *of the same lattice, we say that* $\mathbf{D}$ *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* $k \in [d-1]$*. This is denoted as* $\mathbf{D} \geq_{\mathcal{Q}} \mathbf{C}$ *or* $\mathsf{rr}(\mathbf{D}) \geq_{\mathcal{Q}} \mathsf{rr}(\mathbf{C})$*.*

If we set $n = d - 2$ and $\kappa_i = i$ for ξ in the generic reduction $\mathcal{R}$-ξ, then the Gram-Schmidt norms of the lattice basis after lattice reduction $\mathcal{R}$-ξ can be simulated using the Gaussian Heuristic. In this case, the following property holds:

Property 1 (Property of the Generic Reduction) *Assume Gaussian Heuristic holds. For a generic reduction tour* $\mathcal{R}$-ξ*, where* $\xi = [\![(0, \beta_0), (1, \beta_1), \cdots, (d-2, \beta_{d-2})]\!]$*, 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* $\mathbf{D}' \geq_{\mathcal{Q}} \mathbf{C}'$ *.*

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

If k=0, $|\mathbf{B}'_{\pi[0,1]}| = l'_0 \approx \min\{\mathrm{GH}(\mathbf{B}_{\pi[0,\beta_0]}), l_0\} \leq l_0$ obviously. Assume $|\mathbf{B}'_{\pi[0,n]}| \leq |\mathbf{B}_{\pi[0,n]}|$ holds while $k = n$. When $k = n+1$, if $l_n < \sqrt{\frac{\beta'}{2\pi e}} \cdot \left(\frac{|\mathbf{B}_{\pi[0,n+\beta']}|}{|\mathbf{B}'_{\pi[0,n]}|}\right)^{1/\beta'}$, it implies that the norm of short vector found in $\mathbf{B}_{\pi[n,n+\beta']}$ is greater than l_n, the norm of vector in the current lattice basis. In this case, l_n will not be replaced, and $l_n = l'_n$.

Otherwise, $l'_n = \sqrt{\frac{\beta'}{2\pi e}} \cdot \left(|\mathbf{B}'_{\pi[n,n+\beta']}|\right)^{1/\beta'}$. Since the reduction on $\mathbf{B}_{\pi[n,n+\beta']}$ does not affect the volume of the projected sublattice basis $\mathbf{B}_{\pi[0,n+\beta']}$, $|\mathbf{B}_{\pi[0,n+\beta']}| = |\mathbf{B}'_{\pi[0,n+\beta']}|$, 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 cases: (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

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

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 $\mathbf{x} = (x_0, \cdots, x_{d-1})$ and $\mathbf{y} = (y_0, \cdots, y_{d-1})$. $|\mathbf{C}_{\pi[0,k+1]}| \geq |\mathbf{D}_{\pi[0,k+1]}|$ 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 $\mathbf{x}' = (x'_0, \cdots, x'_{d-1})$ and $\mathbf{y}' = (y'_0, \cdots, y'_{d-1})$. 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 $k{=}n{+}1$, since $|\mathbf{C}_{\pi[0,n+\beta']}| = |\mathbf{C}'_{\pi[0,n+\beta']}|$ and $|\mathbf{D}_{\pi[0,n+\beta']}| = |\mathbf{D}'_{\pi[0,n+\beta']}|$, we have

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

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

Then, we formally define the strategy space in the Two-step mode to solve the LWE problem.

Definition 6 (Strategy Space). *Given a lattice basis $\mathbf{B}$ of an uSVP instance. Suppose there exists a lattice reduction algorithm $\mathcal{R}$ that has an adjustable parameter ξ, along with an SVP oracle $\mathcal{C}$. Let $\mathcal{S}(\mathcal{R}, \mathcal{C})$ denote the corresponding strategy space. Each Two-step strategy $(\mathsf{S} = [\![\xi_0, \cdots, \xi_i, \cdots]\!], d_{\text{svp}}) \in \mathcal{S}$ can find the unique shortest vector of the lattice $\mathcal{L}(\mathbf{B})$ through the following Two-step procedure:*

1. *Reduction Step: Iteratively call the reduction method $\mathcal{R}$ with parameter $\xi_i \in \mathsf{S}$ on the lattice basis $\mathbf{B}$, where each $\mathcal{R}$-ξ_i improves the quality of the basis. The output is the reduced lattice basis $\mathbf{B}'$ after all reductions.*
2. *Search step: Find the unique shortest vector by calling a d_{svp}-dimensional SVP oracle on $\mathbf{B}'_{[d-d_{\text{svp}},d]}$ to obtain the projection of the target vector, then lift it to the full-dimensional lattice.*

BKZ is the most commonly used lattice reduction algorithm, while PnJBKZ (Sec. 2.4) with hk3-sieve is currently the most efficient lattice reduction algorithm in practice. We can choose either BKZ or PnJBKZ as the reduction algorithm $\mathcal{R}$, and use lattice enumeration or (progressive) lattice sieve as the SVP oracle in the search step.

Progressive Reduction. In 2020, Li and Nguyen [49] presented the first rigorous dynamic analysis of BKZ, followed by Wang's [50] dynamic analysis of PnJBKZ in 2024. These studies, along with others [49–51], have established that lattice reduction algorithms, such as BKZ [16] or its variant PnJBKZ [20], and slide reduction [37], when applied with a fixed reduction parameter, will eventually converge to a specific reduced lattice basis after a polynomial number of iterations. For a well-reduced lattice basis, further application of the same lattice reduction algorithm $\mathcal{R}$ (e.g. BKZ or PnJBKZ) with identical or weaker parameters becomes ineffective, as it no longer significantly improves the basis quality. Therefore, to achieve a higher reduction quality, it is essential to dynamically adapt the reduction parameters, enforcing progressively stronger reductions. This adaptive approach is commonly referred to as the progressive reduction strategy.

A valid reduction strategy to improve the quality of the lattice basis is finite, which implies that the strategy space $\mathcal{S}$ finite, as stated in Theorem 1 and proven in Appendix C.1.

Theorem 1. *If a lattice basis* $\mathbf{B}$ *reduced by repeatedly calling a 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 present a time-cost model for each strategy in solving the uSVP problem, as our goal is to find the strategy with the minimum time cost. Given a time-cost model $\mathcal{T} = (T_{\mathcal{R}}, T_{\mathcal{C}})$ and a Two-step strategy $(\mathsf{S}, d_{\text{svp}})$, let the time cost of the entire reduction step be $T_{\mathcal{R}s}(\mathsf{S}) = \sum_{\xi \in \mathsf{S}} T_{\mathcal{R}}(\xi)$, and the time cost of search step be $T_{\mathcal{C}}(d_{\text{svp}})$. Then, the total time cost of the strategy $(\mathsf{S}, d_{\text{svp}})$ for solving the uSVP problem is given by $T_{\text{total}}(\mathsf{S}, d_{\text{svp}}) = T_{\mathcal{R}s}(\mathsf{S}) + T_{\mathcal{C}}(d_{\text{svp}})$.

According to Theorem 1, the strategy space $\mathcal{S}$ for solving an uSVP instance is finite. There exists a strategy $(\mathsf{S}_{\min}, d_{\text{svp}}^{(\min)})$ such that $T_{\text{total}}(\mathsf{S}_{\min}, d_{\text{svp}}^{(\min)}) = \min\{T_{\text{total}}(\mathsf{S}, d_{\text{svp}}) : (\mathsf{S}, d_{\text{svp}}) \in \mathcal{S}\}$. We now formalize the Q7 in Kyber [8] as a combinatorial optimization problem:

Definition 7 (The Problem of Searching a Refined Strategy for Solving LWE). *Given an LWE instance* $(\mathbf{A}, \mathbf{b})$*, it can be transform into an uSVP instance by using a primal attack. Consider an arbitrary lattice reduction algorithm* $\mathcal{R}$*, an SVP oracle* $\mathcal{C}$*, and their corresponding time-cost model* $\mathcal{T}$*. The objective is to identify a refined Two-step strategy* $(\mathsf{S}, d_{\text{svp}}) \in \mathcal{S}(\mathcal{R}, \mathcal{C})$ *that minimizes the total time cost for solving the LWE instance.*

Let TwoStepSolver (Alg. 1) be a Two-step strategy algorithm for solving LWE. It employs the lattice reduction algorithm $\mathcal{R}$ and the SVP oracle $\mathcal{C}$ to solve LWE based on the input Two-step strategy $(\mathsf{S}, d_{\text{svp}})$. This approach first applies a series of reductions and, at the appropriate point, calls an SVP oracle to search for the target vector.

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

Since the time cost of the reduction algorithm increases exponentially with respect to the reduction parameter ξ, exhaustively executing all possible strategies to identify the optimized reduction strategy is computationally infeasible. To accurately estimate the reduction effect for each strategy, we introduce two key components: `ReductionSim`, a polynomial-time simulator of the reduction algorithm $\mathcal{R}$, and `SVPDimEst`, a dimension estimation method for the SVP oracle.

`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. The PnJBKZ simulator in Sec.5 is the first polynomial-time simulator for PnJBKZ. Given the initial Gram-Schmidt norms rr_0 and a reduction strategy S, $\texttt{ReductionSim}(\mathsf{rr}_0, \mathsf{S})$ predicts how the Gram-Schmidt norms changes after iteratively calling $\mathcal{R}$-ξ reduction for $\xi \in \mathsf{S}$ with input rr_0. If S contains a single reduction strategy ξ, we can also re-write $\texttt{ReductionSim}(\mathsf{rr}_0, \mathsf{S})$ as $\texttt{ReductionSim}(\mathsf{rr}_0, \xi)$.

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 oracle $\mathcal{C}$, a Two-step strategy $(\mathsf{S}, d_{\text{svp}}) \in \mathcal{S}(\mathcal{R}, \mathcal{C})$;
output: The unique shortest vector $\mathbf{t}$;

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

Algorithm 1: Two-step Solver

Condition 1 *Given a lattice reduction algorithm $\mathcal{R}$ and an SVP oracle $\mathcal{C}$, we have:*

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

We denote the actual GS values as l_i'' (actual GS values output from $\mathcal{R}$) and the simulated GS values as $\text{Sim}(l_i'')$. Then, In Condition 1, the phrase

"`ReductionSim` can accurately predict the reduction effect of $\mathcal{R}$", means that $\forall i \in [0, d] : l_i''/\mathrm{Sim}(l_i'')$ falls within the range [0.95, 1.05].

If Condition 1 is satisfied, SSearch can find the minimum time cost strategy $(\mathsf{S}_{\min}, d_{\mathrm{svp}}^{(\min)})$. Then, we present a simplified version of SSearch as Alg. 2. The algorithm aims to (1) enumerate all possible lattice reduction strategies and estimate the dimension of the SVP oracle, and (2) compute the total time cost of each LWE solving strategy, ultimately outputting the strategy $(\mathsf{S}_{\min}, d_{\mathrm{svp}}^{(\min)})$ with the lowest time cost.

Since $\mathcal{S}$ is finite, as proved in Theorem 1, and the time cost of each strategy $(\mathsf{S}, d_{\mathrm{svp}})$ is fixed, enumerating all possible reduction strategies S allows us to identify a strategy with minimum time cost. Thus, Corollary 1 holds naturally.

Corollary 1. *If Condition 1 is satisfied, Simplified SSearch outputs a strategy of `Two-step Solver` with minimum 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: $(\mathsf{S}_{\min}, d_{\mathrm{svp}}^{(\min)})$;

Function `SimplifiedSSearch(`$(\mathbf{A}, \mathbf{b}, \sigma)$`, ReductionSim, SVPDimEst,` $\mathcal{T}$`):`
  $\mathsf{rr}_0 \leftarrow$ The Gram-Schmidt norms of a lattice basis $\mathbf{B}$ from LWE instance $(\mathbf{A}, \mathbf{b})$ by the primal attack;
  $(T_{\mathcal{R}}, T_{\mathcal{C}}) \leftarrow \mathcal{T}$; $d_{\mathrm{svp}} \leftarrow$ `SVPDimEst(`rr_0, σ`)`;
  $T_{\min} \leftarrow T_{\mathcal{C}}(d_{\mathrm{svp}})$; $(\mathsf{S}_{\min}, d_{\mathrm{svp}}^{(\min)}) \leftarrow (\llbracket\ \rrbracket, d_{\mathrm{svp}})$;
  for each possible non-empty reduction strategy S **do**
    $\mathsf{rr} \leftarrow$`ReductionSim(`rr_0,S`)`; $d_{\mathrm{svp}} \leftarrow$ `SVPDimEst(`rr, σ`)`;
    $T_{\mathrm{total}}(\mathsf{S}, d_{\mathrm{svp}}) \leftarrow T_{\mathcal{R}s}(\mathsf{S}) + T_{\mathcal{C}}(d_{\mathrm{svp}})$;
    if $T_{\mathrm{total}} < T_{\min}$ **then**
      $(\mathsf{S}_{\min}, d_{\mathrm{svp}}^{(\min)}) \leftarrow (\mathsf{S}, d_{\mathrm{svp}})$; $T_{\min} \leftarrow T_{\mathrm{total}}$;
  return $(\mathsf{S}_{\min}, d_{\mathrm{svp}}^{(\min)})$;

Algorithm 2: Simplied SSearch

When TwoStepSolver employs the minimum time-cost Two-step strategy $(\mathsf{S}_{\min}, d_{\mathrm{svp}}^{(\min)})$ determined by the SSearch algorithm, 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 minimum time-cost strategy, the strategy space is vast, making it challenging to quickly find the minimum time-cost strategy. To address this, we propose the Pruning SSearch method (PSSearch, Alg. 3), which reduces the complexity of searching the entire strategy space by pruning strategies that will not evolve into the final solution. In Sec. 4.2, we provide a formal correctness proof for our algorithm.

4.1 Main Algorithm

In PSSearch, we search for the strategy with the minimum time cost by traversing a 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* $\mathsf{root} = ([\![\,]\!], d_{\mathrm{svp}}^{(0)})$, *where* $d_{\mathrm{svp}}^{(0)}$ *is calculated by* `SVPDimEst`.
2. *The i-th level node is in the form* $(\mathsf{S}_i = [\![\xi_1, ..., \xi_i]\!], d_{\mathrm{svp}}^{(i)})$, *which consists of a reduction strategy with length i, and* $d_{\mathrm{svp}}^{(i)} = \texttt{SVPDimEst}(\texttt{ReductionSim}(\mathsf{rr}_0, \mathsf{S}_i), \sigma)$.
3. *For any i-th level strategy* $(\mathsf{S}_i, d_{\mathrm{svp}})$, *its child node is* $(\mathsf{S}_{i+1} = \mathsf{S}_i \cup [\![\xi_{i+1}]\!], d_{\mathrm{svp}}^{(i+1)})$, *where ξ_{i+1} is any reduction parameter that can further improve the basis quality.*

Now, we describe the procedure of the PSSearch algorithm. Fig. 4 presents an example of a strategy tree and the pruning process for each strategy node.

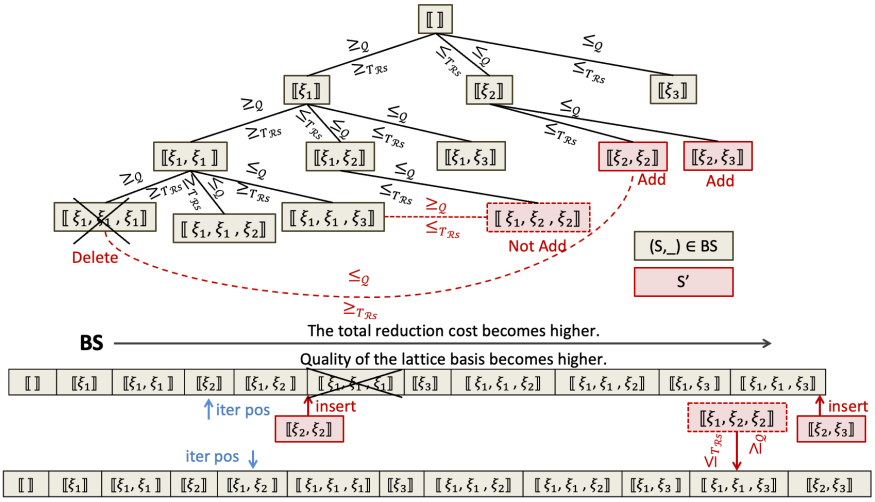


Fig. 4: Illustrative Example of PSSearch. "$\geq_{T_{\mathcal{R}s}}$" indicates that the reduction time cost of the strategy on the left side is greater than that of the strategy on the right side, i.e., $T_{\mathcal{R}s}(\mathsf{S}_{\mathrm{left}}) \geq T_{\mathcal{R}s}(\mathsf{S}_{\mathrm{right}})$. Each box displays a reduction strategy S.

At the beginning, PSSearch generates an empty strategy list BS, where $\sharp\mathsf{BS}$ denotes the size of BS. Each element in BS is a pair $(\mathsf{S}, d_{\mathrm{svp}})$. Let rr_0 be the sequence of Gram-Schmidt norms for the input lattice basis $\mathbf{B}$. We start by adding a strategy $(\mathsf{S}_0 = [\![\,]\!], d_{\mathrm{svp}}^{(0)})$ with no reduction to the list BS, corresponding to the root node of $\mathcal{W}$, denoted as root. $d_{\mathrm{svp}}^{(0)} = \mathtt{SVPDimEst}(\mathsf{rr}_0, \sigma)$. The time

cost of this strategy is as follows: $T_{\text{total}}(\mathsf{S}_0, d_{\text{svp}}^{(0)}) = T_{\mathcal{R}s}(\mathsf{S}_0) + T_{\mathcal{C}}(d_{\text{svp}}^{(0)}) = 0 + T_{\mathcal{C}}(d_{\text{svp}}^{(0)}) = T_{\mathcal{C}}(d_{\text{svp}}^{(0)})$.

Then, access the strategies in BS in order and determine whether to add a child strategy node to BS based on a specific pruning rule, continuing until no child nodes remain to be searched. For example, consider all possible strategies of size 1, which are the child nodes of root, i.e. $(\llbracket \xi \rrbracket, d_{\text{svp}})$ for different values of ξ. After searching all the nodes are searched, the strategy with the minimum time cost in BS is output. The strategies in the list BS are organized such that the time cost of the reduction step $T_{\mathcal{R}s}(\mathsf{S})$ increases, while the quality of the lattice basis improves (which implies that the time cost of the search step $T_{\mathcal{C}}(d_{\text{svp}})$ decreases, as stated in Lemma 1).

The pruning strategy is not based solely on the total time cost, as there may exist a descendant node with a lower total time cost. Instead, both the time cost and the quality of the basis after the reduction step are compared between nodes. If a node has both a higher time cost and worse basis quality than another, it cannot evolve into a strategy with the lowest total time cost (as stated in item 2 of Property 1) and therefore can be discarded.

In other words, for each newly proposed strategy $(\mathsf{S}', d'_{\text{svp}})$ awaiting addition, we compare it against each current strategy $(\mathsf{S}, d_{\text{svp}}) \in \mathsf{BS}$ and apply the following rules to decide whether to preserve or discard it.

Additionally, if the goal is to output the strategy with the minimum time overhead while considering memory constraints, we need to modify the conditions in lines 6 and 13 to preserve the small blocks strategy, even if its reduction time cost is higher than that of another strategy. Under memory limitations, the strategy with the lowest time overhead will be selected as the output.

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

1. *If there exists a $(\mathsf{S}, d_{\text{svp}}) \in \mathsf{BS}$ that has both lower reduction time cost and better basis quality than which of $(\mathsf{S}', d'_{\text{svp}})$, i.e. $T_{\mathcal{R}s}(\mathsf{S}) < T_{\mathcal{R}s}(\mathsf{S}')$ and $\texttt{ReductionSim}(\mathsf{rr}_0, \mathsf{S}) \geq_{\mathcal{Q}} \texttt{ReductionSim}(\mathsf{rr}_0, \mathsf{S}')$, then don't add $(\mathsf{S}', d'_{\text{svp}})$.*

2. *Otherwise, add $(\mathsf{S}', d'_{\text{svp}})$ and remove all strategies in BS that have both a higher reduction time cost and worse basis quality than $(\mathsf{S}', d'_{\text{svp}})$.*

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: $(\mathsf{S}_{\min}, d_{\text{svp}}^{(\min)})$;

Function `PSSearch`$((\mathbf{A}, \mathbf{b}, \sigma)$, `ReductionSim`, `SVPDimEst`, $\mathcal{T}$`):`
  $\mathsf{rr}_0 \leftarrow$ The Gram-Schmidt norms of a lattice basis $\mathbf{B}$ from LWE instance $(\mathbf{A}, \mathbf{b})$ by the primal attack;
  $(T_{\mathcal{R}}, T_{\mathcal{C}}) \leftarrow \mathcal{T}$; $k \leftarrow 0$; $d_{\text{svp}} \leftarrow$ `SVPDimEst`(rr_0, σ); $\mathsf{BS} \leftarrow [\![([\![\,]\!], d_{\text{svp}})]\!]$;
  while $k < \sharp\mathsf{BS}$ **do**
    $(\mathsf{S}_1, d_{\text{svp}}) \leftarrow \mathsf{BS}[k]$;
    for $\forall \xi'$ s.t. `ReductionSim`$(\mathsf{rr}_0, \mathsf{S}_1 \cup [\![\xi']\!]) \geq_{\mathcal{Q}}$ `ReductionSim`$(\mathsf{rr}_0, \mathsf{S}_1)$ **do**
      $\mathsf{S}' \leftarrow \mathsf{S}_1 \cup [\![\xi']\!]$; $\mathsf{rr}' \leftarrow$ `ReductionSim`$(\mathsf{rr}_0, \mathsf{S}')$;
      $d'_{\text{svp}} \leftarrow$ `SVPDimEst`(rr', σ);
      if $\exists\ (\mathsf{S}, d_{\text{svp}}) \in \mathsf{BS}$ s.t. $T_{\mathcal{R}s}(\mathsf{S}') \geq T_{\mathcal{R}s}(\mathsf{S})$ **and** `ReductionSim`$(\mathsf{rr}_0, \mathsf{S}') \leq_{\mathcal{Q}}$ `ReductionSim`$(\mathsf{rr}_0, \mathsf{S})$ **then**
        continue;
      else
        $\mathsf{BS} \leftarrow \mathsf{BS} \cup [\![(\mathsf{S}', d'_{\text{svp}})]\!]$;
        for $\forall (\mathsf{S}, d_{\text{svp}}) \in \mathsf{BS}$ s.t. $T_{\mathcal{R}s}(\mathsf{S}') \leq T_{\mathcal{R}s}(\mathsf{S})$ **and** `ReductionSim`$(\mathsf{rr}_0, \mathsf{S}') \geq_{\mathcal{Q}}$ `ReductionSim`$(\mathsf{rr}_0, \mathsf{S})$ **do**
          Remove $(\mathsf{S}, d_{\text{svp}})$ from BS;
    $k \leftarrow k + 1$;
  Find the strategy $(\mathsf{S}_{\min}, d_{\text{svp}}^{(\min)}) \in \mathsf{BS}$ s.t. $T_{\text{total}}(\mathsf{S}_{\min}, d_{\text{svp}}^{(\min)}) = \min_{(\mathsf{S}, d_{\text{svp}}) \in \mathsf{BS}} T_{\text{total}}(\mathsf{S}, d_{\text{svp}})$;
  return $(\mathsf{S}_{\min}, d_{\text{svp}}^{(\min)})$;

Algorithm 3: Pruning SSearch

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.

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

Before proving Theorem 2, we present Lemma 1, which supports Theorem 2 by proving that for two different lattice bases with the same dimension, the inequality between their dimension estimation values obtained from `SVPDimEst` remains unchanged after performing the same lattice basis reduction.

Lemma 1. *Suppose the Gaussian Heuristic holds. Given an `SVPDimEst` described as described in Sec. 2.5, and two arbitrary bases $\mathbf{D} \neq \mathbf{C}$ for the same d-dimensional lattice generated from an LWE instance with standard deviation*

σ by the primal attack, assume that $\mathbf{D} \geq_{\mathcal{Q}} \mathbf{C}$. Let $\mathbf{C}'$ (resp. $\mathbf{D}'$) be the Gram-Schmidt norms of the lattice basis after calling a tour of $\mathcal{R}$-ξ on $\mathbf{C}$ (resp. $\mathbf{D}$). Then, it holds that $\texttt{SVPDimEst}(\mathsf{rr}(\mathbf{C}'), \sigma) \geq \texttt{SVPDimEst}(\mathsf{rr}(\mathbf{D}'), \sigma)$.

The proof of Lemma 1 follows a similar approach to that of Property 1, and thus, we omit the details for brevity. For the full proof, refer to the Appendix C. Now, we prove Theorem 2 as follows:

Proof. (Proof of Theorem 2.) Let rr_0 be the Gram-Schmidt norms of a randomly chosen lattice basis. Suppose the strategy in $\mathcal{S}$ with the minimum time cost is $(\mathsf{S} = [\![\xi_0, ..., \xi_{z-1}]\!], d_{\text{svp}})$. We denote the sub-strategy $[\![\xi_0, ..., \xi_i]\!]$ of S by S_i and define $d_{\text{svp}}^{(i)} := \texttt{SVPDimEst}(\texttt{ReductionSim}(\mathsf{rr}_0, \mathsf{S}_i), \sigma)$ for $0 \leq i \leq z$.

For all $1 \leq i \leq z$, $T_{\mathcal{C}}(d_{\text{svp}}^{(i-1)}) > T_{\mathcal{C}}(d_{\text{svp}}^{(i)})$. Otherwise, removing ξ_i from S_i would result in a strategy that solves an LWE instance in less time.

From the description of PSSearch, two cases might arise: (1) S is included in the final strategy set BS, or (2) there exists a sub-strategy S_i of S that has been removed from BS, in which case S will no longer appear in BS. Since S has the minimum time cost among all strategies in BS, it must be the final output strategy, satisfying the first case. We now prove that the second case cannot occur.

If $(\mathsf{S}_i, d_{\text{svp}}^{(i)})$ is removed from BS, then there must exist another strategy $(\mathsf{S}', d'_{\text{svp}})$ such that $\texttt{ReductionSim}(\mathsf{rr}_0, \mathsf{S}') \geq_{\mathcal{Q}} \texttt{ReductionSim}(\mathsf{rr}_0, \mathsf{S}_i)$, and the reduction time cost satisfies $T_{\mathcal{R}s}(\mathsf{S}_i) \geq T_{\mathcal{R}s}(\mathsf{S}')$. Now, consider a new strategy $\mathsf{S}^* := \mathsf{S}' \cup [\![\xi_{i+1}, ..., \xi_{z-1}]\!]$. This implies that $T_{\mathcal{R}s}(\mathsf{S}) \geq T_{\mathcal{R}s}(\mathsf{S}^*)$. Let $d^*_{\text{svp}} = \texttt{SVPDimEst}(\texttt{ReductionSim}(\mathsf{rr}_0, \mathsf{S}^*), \sigma)$. From Lemma 1 and the increasing property of $T_{\mathcal{C}}$, it deduces that $T_{\mathcal{C}}(d_{\text{svp}}) \geq T_{\mathcal{C}}(d^*_{\text{svp}})$. Since $T_{\mathcal{R}s}(\mathsf{S}_i) \geq T_{\mathcal{R}s}(\mathsf{S}^*)$, this leads to a contradiction, as it implies that S is not the strategy with the minimum total time cost. □

4.3 A Refined SVP Dimension Estimation Method

We have introduced the classical `SVPDimEst` method in Sec. 2.5. However, estimating the projected norm of the target vector as $\|\pi_{d-d_{\text{svp}}}(\mathbf{t})\| = \sigma\sqrt{d_{\text{svp}}}$ carries a failure probability in the last SVP call when solving LWE. This is because $\mathbf{t}$ follows a specific distribution (typically discrete Gaussian), and 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 address this issue, we propose `SVPDimEst` (Alg. 5) for estimating the sieving dimension to solve general LWE problems with arbitrary target vector distributions. Our estimation builds based on the probabilistic model used in BKZ-simulation of [45]. Specifically, we adapt and refine this estimator to suit the SVP oracle, allowing us to estimate the required dimension for a single SVP call. The key difference is in the underlying model: while previous works relied on the BKZ simulator for estimation, our approach tailors the estimation process specifically for the SVP oracle. This adaptation ensures a more accurate and practical dimension estimation for the SVP oracle in this context. Similarly to Lemma 2, a detailed description is provided in Appendix B.1. It retains the same properties as stated in Lemma 1.

5 The Design of PnJBKZ Simulator

PnJBKZ [20], when combined with hk3-sieve, is considered the most efficient lattice reduction algorithm in practice. However, PnJBKZ currently lacks a polynomial-time accurate simulator. Since `ReductionSim`, as mentioned in Condition 1, is an essential part of both SSearch and PSSearch, we must develop an accurate simulator for PnJBKZ, especially when jump>1. Only then will we be able to generate optimized reduction strategies for PnJBKZ using PSSearch.

The rigorous dynamic analysis proposed in [49, 50] analyzes the reduction effect, which only converges after running numerous tours of a fixed blocksize BKZ-β or PnJBKZ-(β, J) reduction. However, in practice, a polynomial-time PnJBKZ-(β, J) simulator is needed to predict the practical reduction effects of PnJBKZ-(β, J) with a more flexible number of tours. This is because, during most progressive reduction processes, the number of BKZ-β reduction tours is often relatively small [17, 19, 20, 37].

Meanwhile, the classical BKZ simulator is unable to predict the performance of PnJBKZ when jump>1. Fig. 5 shows that a PnJBKZ reduction strategy with jump>1 can significantly improve the efficiency of reduction. Specifically, the progressive reduction strategy with jump=9 is four times faster than the strategy with jump=1 in achieving the same slope of -0.019. To optimize strategies for PnJBKZ, selecting the appropriate (β, J) is critical. Therefore, it is essential to develop a polynomial-time PnJBKZ simulator that accurately predicts the performance of PnJBKZ reduction.

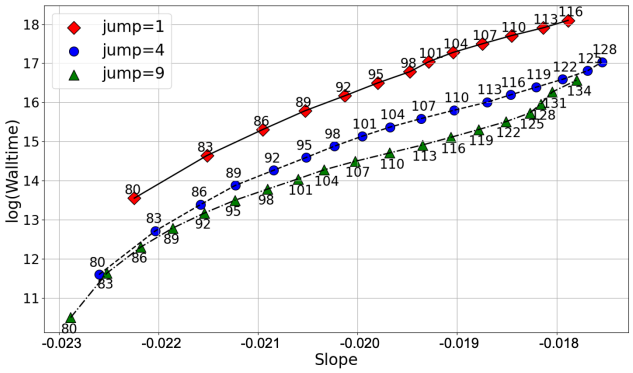


Fig. 5: Speedup in Reduction Efficiency via the Jump Strategy.The test conducted on a 252-dimensional lattice basis. The walltime and slope values are averaged over 5 instances for each test. Each instance was run on machine C (2 GPUs and 32 threads). The points are labeled by β.

5.1 The PnJBKZ Simulator Construction.

Before presenting the detailed construction of the PnJBKZ simulator, we first briefly review the main idea behind the BKZ 2.0 simulator proposed in [17], which uses the Gaussian Heuristic (GH) to predict BKZ-β reduced lattice basis.

Let $\mathcal{L}(\mathbf{B}^{(i)})$ denote the lattice basis after the reduction of the first i blocks, and $\mathcal{L}(\mathbf{B}^{(0)})$ as the initial lattice basis. $\forall i \in [d]$, define $l_i = \ln(\|\mathbf{b}_i^*\|)$, $l_i' = \ln(\|\mathbf{b}_i^{*'}\|)$, and $l_i'' = \ln(\|\mathbf{b}_i^{*''}\|)$ as the logarithmic norms of the i-th Gram-Schmidt basis vectors reduced by one tour of BKZ-β and PnJBKZ(β, J), respectively.

The BKZ 2.0 simulator first calculates $\mathrm{Sim}(l_0') = \ln\left(\mathrm{GH}(\mathcal{L}(\mathbf{B}^{(0)}_{\pi[0,\beta]})\right) \approx \frac{1}{2}\ln(\beta/2\pi e) + \frac{1}{\beta}\ln\left(\mathrm{Vol}(\mathcal{L}(\mathbf{B}^{(0)}_{\pi[0,\beta]}))\right)$ under the GH. Then, it calculates $\mathrm{Sim}(l_1') := \ln\left(\mathrm{GH}(\mathcal{L}(\mathbf{B}^{(1)}_{\pi[1,\beta+1]}))\right) \approx \frac{1}{2}\ln(\beta/2\pi e) + \frac{1}{\beta}\left(\ln \mathrm{Vol}(\mathcal{L}(\mathbf{B}^{(0)}_{\pi[0,\beta+1]})) - \mathrm{Sim}(l_0')\right)$, using GH and the previously calculated value of $\mathrm{Sim}(l_0')$.

Since inserting the new $\mathbf{b}_0$ alters the norms of the Gram-Schmidt (GS) vectors, for $\forall i \in \{1, ..., d-1\}$, l_i changes to some unknown values. Although l_0 changes to l_0' after inserting the new $\mathbf{b}_0$, the reduction on $\mathcal{L}(\mathbf{B}^{(0)}_{\pi[0,\beta]})$ does not affect the volume of the projected sublattice basis $\mathcal{L}(\mathbf{B}^{(0)}_{\pi[0,\beta+1]})$. Therefore, we have $\mathrm{Vol}(\mathcal{L}(\mathbf{B}^{(1)}_{\pi[0,\beta+1]})) = \mathrm{Vol}(\mathcal{L}(\mathbf{B}^{(0)}_{\pi[0,\beta+1]}))$.

Thus, as $\mathrm{Vol}(\mathcal{L}(\mathbf{B}^{(0)}_{\pi[1,\beta+1]})) = \prod_{i=1}^{\beta}\|\mathbf{b}_i^*\|$, after the insert of new $\mathbf{b}_0$ it will change to $\mathrm{Vol}(\mathcal{L}(\mathbf{B}^{(1)}_{\pi[1,\beta+1]})) = \left(\prod_{i=0}^{\beta}\|\mathbf{b}_i^*\|\right) / \|\mathbf{b}_0^{*'}\|$. Here, $\mathrm{Sim}(l_0')$ is a simulated approximate value of l_0' by GH. Iteratively calculating all remaining unknown $\mathrm{Sim}(l_i') := \ln\left(\mathrm{GH}(\mathcal{L}(\mathbf{B}^{(i)}_{\pi[i,i+\beta]}))\right) \approx \frac{1}{2}\ln(\beta/2\pi e) + \frac{1}{\beta}\left(\ln \mathrm{Vol}(\mathcal{L}(\mathbf{B}^{(0)}_{\pi[0,i+\beta]})) - \sum_{j=0}^{i-1}\mathrm{Sim}(l_j')\right)$, for $\forall i \in [d-\beta]$. For $\forall i \in \{d-\beta, ..., d-1\}$, $\mathrm{Sim}(l_i')$, 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 simulate the behavior of PnJBKZ when $\mathsf{jump}>1$. Let $\mathsf{jump}=J$ and $\mathcal{L}(\mathbf{B}^{(i)})$ represent the lattice basis after the final vector has been inserted into the i-th position of the GS vectors. Also, let $\mathcal{L}(\mathbf{B}^{(0)})$ denote the initial lattice basis. Observe that when $J>1$, each time a new vector $\mathbf{b}_i^*$ is inserted into the first position of $\mathbf{B}^{(i)}_{\pi[i,k]}$, the norms of the $J-1$ GS vectors $\mathbf{b}_{i+1}^*, ..., \mathbf{b}_{i+J-1}^*$, 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, in turn, makes it impossible to predict the norms of subsequent vectors through iterative calculations.

To solve the problem we mentioned above, our idea is to use the properties of HKZ-reduced lattice bases to estimate these unknown norms between adjacent blocks when $J>1$. We first present an idealized version of the Pump algorithm, denoted as Pump$'$. This version satisfies the property that a projected sublattice basis $\mathbf{B}_{\pi[i,i+\beta]}$ after the reduction by Pump$'(\mathbf{B}_{\pi[i,i+\beta]}, i, \beta, f)$, strictly satisfies the property of an HKZ-reduced basis for all $i \in [d-\beta]$, where the dimension of the entire lattice basis $\mathbf{B}$ is d.

Next, we construct a PnJBKZ simulator for PnJBKZ using Pump′. Let l_i'' represent the logarithm of each Gram-Schmidt norm after a PnJBKZ(β, J) tour. Our PnJBKZ simulator is denoted as `PnJBKZSim`, which simulates the change of a lattice basis $\mathbf{B}$ after reduction by PnJBKZ. Under GH, we simulate each l_i'' as $\mathrm{Sim}(l_i'') = \ln\left(\mathrm{GH}\left(\mathcal{L}(\mathbf{B}^{(i)}_{\pi[i,k]})\right)\right)$, where $i \in [d-\beta]$ and $k = \min\{i-(i \bmod J)+\beta, d\}$.

The key challenge lies in calculating the volume of $\mathcal{L}(\mathbf{B}^{(i)}_{\pi[i,k]})$. Similar to the BKZ reduction process, during each block reduction in PnJBKZ, the basis matrix of the projected sublattice undergoes a unimodular transformation. So the volume of the projected sublattice remains unchanged, i.e. $\mathrm{Vol}(\mathcal{L}(\mathbf{B}^{(i)}_{\pi[0,k]})) = \mathrm{Vol}(\mathcal{L}(\mathbf{B}^{(0)}_{\pi[0,k]}))$. Suppose $\mathrm{Sim}(l_j'')$ is known, $\forall j \in [i]$, we calculate $\ln\left(\mathcal{L}(\mathbf{B}^{(i)}_{\pi[i,k]})\right) := \ln\left(\mathrm{Vol}\left(\mathcal{L}(\mathbf{B}^{(0)}_{\pi[0,k]})\right)\right) - \ln\left(\mathrm{Vol}\left(\mathcal{L}(\mathbf{B}^{(i)}_{\pi[0,i]})\right)\right) = \sum_{j=0}^{k-1} l_j - \sum_{j=0}^{i-1}\mathrm{Sim}(l_j'')$, where $i \in [d]$, and $k = \min\{i-(i \bmod J)+\beta, d\}$. Under GH, for $i \in [d]$, we can iteratively compute $\mathrm{Sim}(l_i'') :=$

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

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 of the GS norms for a lattice basis after each tour of the PnJBKZ(β, J) reduction[5]. Here, l_i'' are the actual GS vector norms after a tour of PnJBKZ(β, J) reduction. A detailed description of the PnJBKZ simulator is provided in Alg. 4.

[5] When $i \equiv 1 (\bmod J)$, the index i of $\mathrm{GH}\left(\mathcal{L}\left(\mathbf{B}^{(i)}_{\pi[i,i+\beta-(i \bmod J)]}\right)\right)$ in our simulator is the same as that of $\mathrm{GH}\left(\mathcal{L}\left(\mathbf{B}^{(i)}_{\pi[i,i+\beta]}\right)\right)$ in the BKZ-2.0 simulator. The form of calculation is the same in both simulators, but the simulated volumes of projected sublattice differ between them. Because in BKZ 2.0 simulator [17] $\mathrm{Vol}(\mathcal{L}'\left(\mathbf{B}^{(i)}_{\pi[i,i+\beta]}\right)) = \prod_{j=0}^{i+\beta-1}\|\mathbf{b}_j^*\| / \prod_{j=0}^{i-1}\|\mathbf{b}_j^{*'}\|$ and it calculates $\|\mathbf{b}_j^{*'}\|$ by $\|\mathbf{b}_j^{*'}\| := \mathrm{GH}\left(\mathcal{L}'\left(\mathbf{B}^{(i)}_{\pi[j,j+\beta]}\right)\right)$, while in PnJBKZ simulator $\mathrm{Vol}(\mathcal{L}''\left(\mathbf{B}^{(i)}_{\pi[i,i+\beta]}\right)) = \prod_{j=0}^{i+\beta-1}\|\mathbf{b}_j^*\| / \prod_{j=0}^{i-1}\|\mathbf{b}_j^{*''}\|$ and $\ln(\|\mathbf{b}_j^{*''}\|)$ obtained from Eq. (3). Consequently, when $i \equiv 1 (\bmod J)$, the calculation of $\mathrm{GH}\left(\mathcal{L}\left(\mathbf{B}^{(i)}_{\pi[i,i+\beta]}\right)\right)$ differs between the two simulators.

input : rr, blocksize $\beta \in \{45, \cdots, d\}$, 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.

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

Algorithm 4: PnJBKZ Simulator

5.1.1 Consider the impact of LLL reduction in PnJBKZ. Similar to the classical BKZ simulator [17], the original version of the PnJBKZ simulator overlooked the impact of LLL reduction on the lattice basis after each Pump step. In practice, however, LLL reduction is applied after each Pump step during a PnJBKZ-(β, J) tour. This LLL reduction leads to a smooth decrease in the GS norms, as observed experimentally. To improve the prediction accuracy of the

PnJBKZ simulator, we incorporated the Size-reduced and Lovász conditions of the LLL-reduced basis into the original PnJBKZ Simulator.

5.2 Upper bound of Jump in PnJBKZ simulator

This section shows that using Pump′ to simulate the actual Pump is reasonable when the Jump value in PnJBKZ is below a specific upper bound.

To predict the reduction effect of PnJBKZ, which uses Pump′ as its SVP oracle, we set the upper bound of jump value J to be β. Specifically, the lattice basis $\mathbf{B}_{[i,i+\beta]}$ reduced by a β-dimensional Pump′ is HKZ reduced, meaning $\|\mathbf{b}_i^*\| = \lambda_1\left(\mathcal{L}_{\pi[i,k]}\right)$, $i < d$, $k = \min\{i+\beta-(i \bmod J), d\}$. When $J \leq \beta$, the PnJBKZ simulator uses $\mathrm{Sim}(l_i'') = \ln\left(\mathrm{GH}\left(\mathcal{L}(\mathbf{B}^{(i)}_{\pi[i,k]})\right)\right)$ to predict $\lambda_1\left(\mathcal{L}_{\pi[i,k]}\right)$, and GH ensures that PnJBKZ simulator is accurate.

However, when $J>\beta$, there will always be $J-\beta>0$ vectors at the beginning of each block whose norms remain unknown, rendering the PnJBKZ simulator unusable in such cases. Additionally, the verification experiment results presented in Appendix D.1.1 demonstrate the accuracy of the PnJBKZ simulator in predicting the reduction effect of the PnJBKZ, which uses the ideal version of Pump′. This verification of the ideal scenario serves to confirm the correctness of the fundamental theory behind our construction of the PnJBKZ simulator.

In practice, the d4f technique [26] is commonly used to accelerate the sieve in each Pump step of PnJBKZ. During the Pump-up stage, the Pump performs a $(\beta - \mathrm{d4f}(\beta))$-dimensional progressive sieve, and can thus perform at most $(\beta-\mathrm{d4f}(\beta))$ embeddings during the Pump-down stage. Consequently, the output basis of each β-dimensional Pump will inevitably include $\mathrm{d4f}(\beta)$-dimensional GS vectors that do not satisfy the HKZ-reduced basis properties. Thus, the upper bound of Jump value should be smaller than $\beta - \mathrm{d4f}(\beta)$.

Finally, when predicting the reduction effect of PnJBKZ used in practice, which uses Heuristic optimistic d4f value $\mathrm{d4f}_{\mathrm{op}}(\beta)$ for each Pump, we set the upper bound of jump J to be $\mathrm{d4f}_{\mathrm{slope}}(s) = -\ln(4/3)/s$. As demonstrated in Appendix D.1.2, the actual d4f depends on the current lattice basis quality, rather than relying on a fixed Heuristic optimistic d4f value used in the implementation of G6K. Further discussions and numerous experiments verifying the accuracy of the PnJBKZ simulator in predicting PnJBKZ with the optimistic d4f value can be found in Appendix D.1.2, D.1.3 and D.1.4.

6 Experiments and Application to LWE

Since PnJBKZ with hk3-sieve (resp. Pump using hk3-sieve and pump-down) is recognized as the most efficient practical lattice reduction algorithm (resp. SVP oracle) currently, we use PnJBKZ (resp. Pump) for the TwoStepSolver. This section demonstrates the efficiency of the MinTwoStepSolver in solving LWE instances in practice. Additionally, we discuss its efficiency in solving SVP in Appendix E. The MinTwoStepSolver algorithm is defined as:

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

This algorithm combines TwoStepSolver with a strategy generated by the PSSearch, all under a specific time-cost model $\mathcal{T}$, to effectively solve LWE. Sec. 6.1 presents verification experiments for the PnJBKZ simulator, which is a key 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 used in Sec. 6.2 for solving the TU Darmstadt LWE Challenges. Additionally, a simulation accuracy test is provided 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

Appendix D.1.1 shows the accuracy of the PnJBKZ simulator in predicting the reduction effect of PnJBKZ, when PnJBKZ uses the ideal version of Pump′. Additionally, to ensure that the PnJBKZ simulator accurately predicts the PnJBKZ reduction effect when using the optimistic d4f value, we apply a more refined d4f value estimation, as proposed in [43]. This refined estimation is used to adjust the optimistic d4f settings during the simulation of PnJBKZ reduction. For further details, see Appendix D.1.2.

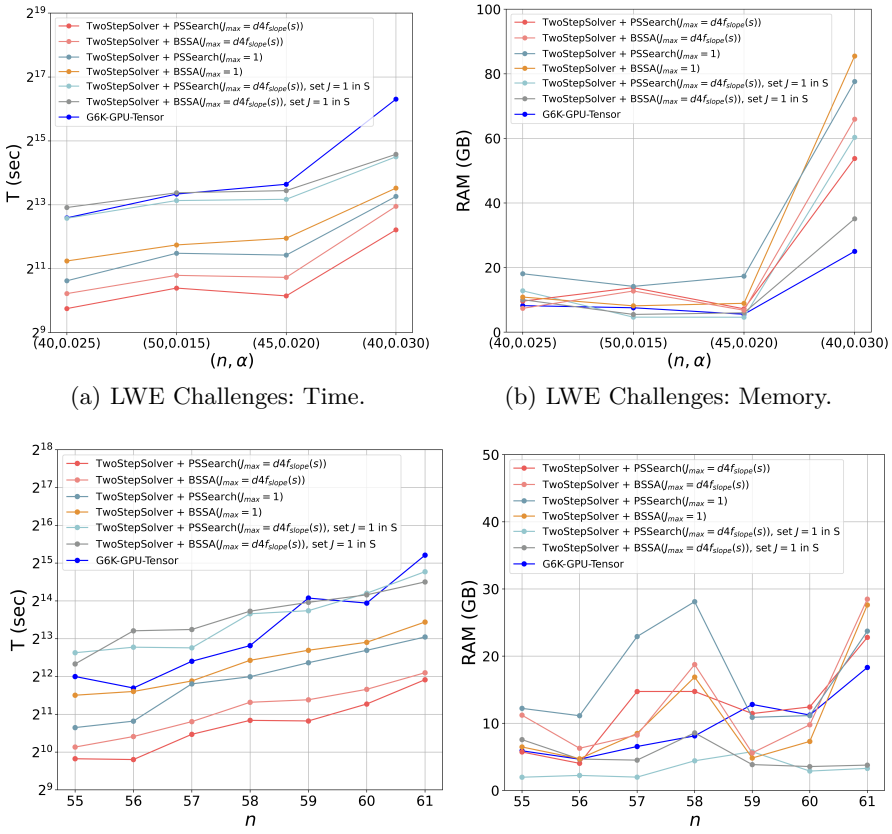
The experimental results in Appendix D.1.3 show that for various reduction parameters, such as blocksize, jump, and tours, most ratios $l_i''/\mathrm{Sim}(l_i'')$ fall within the range [0.95, 1.05]. Ratios outside this range still fall within [0.90, 1.10], as shown in Figs. 17, 18. In comparison, when the CN11 simulator is used to simulate the classic BKZ, most ratios also fall within [0.95, 1.05]. However, some extreme ratios of the CN11 simulator exceed this range, with the largest ratio reaching 1.15 and the smallest falling below 0.85. This indicates that the prediction accuracy of PnJBKZ simulator for PnJBKZ reductions using d4f technology is at least as good as the classic BKZ simulator [17] for predicting BKZ reductions. For more details on predicting PnJBKZ with the d4f technique, see Appendix D.1.2 and D.1.3.

More importantly, the primary purpose of constructing the polynomial-time PnJBKZ simulator is to search a more efficient reduction strategy for solving LWE in practice. The key feature of the simulator is that, given a reduction strategy that uses Jump, it can accurately predict the basis quality during the reduction process. Our experiments, detailed in Table 3 and Appendix D.1.5, show 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. The prediction error of the lattice basis slope does not exceed 0.03%. Therefore, the PnJBKZ simulator we have constructed is sufficiently effective for its intended purpose. Additional results of the verification experiments with different reduction parameters on various lattice bases can be found in Appendix D.1.3 and D.1.4 and as well as in a GitHub link[3].

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

6.2 Efficiency of MinTwoStepSolver for solving LWE

We use the Blocksize and Jump Strategy Selection Algorithm(BSSA, Appendix B.3) for comparison in experiments. BSSA, derived from ProBKZ [19], replaces the BKZ and enumeration algorithms with PnJBKZ and Pump, which still uses the shortest path algorithm to search for optimized reduction strategies.



(a) LWE Challenges: Time.

(b) LWE Challenges: Memory.

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

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

§ The experiment used the "dd" float type and pump/down=True, under an identical benchmarking condition (machine C). "default G6K" refers to the implementation of "lwe_challenge.py" in G6K-GPU-Tensor. 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 with jump=1.

Fig. 6: Comparison of Different LWE-solving Algorithms under same benchmark. §

From the result of Fig. 6 and Table 4, we observe that using the strategy selected by PSSearch (resp. BSSA) significantly decreased the walltime cost by

about 7.2∼23.4 (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" for calculations. The log files corresponding to Fig. 6 can be found in the folders `lwechal-test` and `lwe-instance-test`. These results can also be reproduced by running the test code `implement_lwechal_forall.sh` and `implement_lwe_instance_forall.sh` in source code[2].

Additionally, Fig. 7 demonstrates that, during the reduction step, the optimized reduction strategy found by using the PSSearch can be 4 to 36.7 times faster than the trivial progressive reduction strategy, while achieving the same or better lattice basis quality.

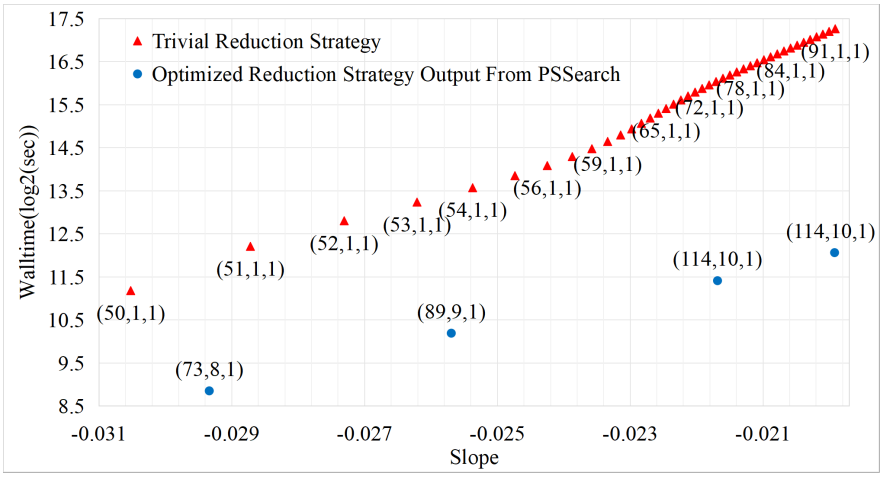


Fig. 7: Speedup in Reduction Step via the optimized strategy generated from PSSearch.Test on LWE challenge (n=60,α = 0.010), which constructs a 222-dimensional lattice basis by using the primal attack. The points are labeled by the reduction parameters (β,J,tours).

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. For more details on the LWE solving efficiency comparison experiments, see Appendix D.2.1. For more information about BSSA, refer to Appendix B.3.

6.3 Optimized strategy generated by MinTwoStepSolver

We use PSSearch (Alg. 3) with the practical cost model mentioned in Appendix B.2 to select the blocksize and jump strategy for solving some instances of TU Darmstadt LWE Challenges, and test it on machine C.

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

The selected strategies from PSSearch are listed in Table 2, which also shows that the time cost of generating the reduction strategy by PSSearch is acceptable. We provide the open source code of PSSearch in the folder "`strategy_gen`" from source code[2]. We successfully solved the TU Darmstadt LWE Challenge instances with $(n, \alpha) \in \{(40, 0.035), (40, 0.040), (50, 0.025), (55, 0.020), (90, 0.005)\}$ by the selected strategies in Appendix D.3.

6.4 Simulated Accuracy of MinTwoStepSolver for LWE

To show the accuracy of the reduction estimation and time-cost model, we compare the predicted quality of the lattice basis and walltime with the results from actual experiments at each middle node during the reduction step. Table 3 and Appendix D.1.5 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]

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

(β, J)	Simulation		Practical	
	Slope	$\log_2$(T)	Slope	$\log_2$(T)
(56, 8)	−0.0285	6.0	−0.0277	6.2
(80, 10)	−0.0250	6.3	−0.0245	6.5
(81, 10)	−0.0232	6.3	−0.0231	6.6
(102, 11)	−0.0210	7.5	−0.0212	7.8
(114, 11)	−0.0196	9.1	−0.0198	9.2
(119, 11)	−0.0190	10.0	−0.0191	10.1

Table 4: Actual running time and RAM cost for LWE Challenges.

(n,α)	Machine	CPU threads	T (h)	RAM (GB)
(80,0.005)	C	32	2.78	7.3
(40,0.035)	C	32	50.4	326
(40,0.035)[a]	C	32	1180	283
(50,0.025)	A	128	592	184
(55,0.020)	A	128	611	890
(90,0.005)	B	64	370	332
(40,0.040)	A	128	683	1120

[a] Use the default LWE solving strategy in G6K-GPU-Tensor [28].

6.5 New LWE Records

Based on MinTwoStepSolver, we solved six LWE instances in TU Darmstadt LWE Challenge website[1]. See Fig. 2 and Appendix D.2.2 for more details.

6.6 Security Estimation for NIST schemes

We re-estimate the hardness of the LWE-based NIST schemes [52] to estimate the influence of the optimized solving strategy found by PSSearch. Under the RAM

[5] The data in Table 3 is extracted from a test in Fig. 6 for comparing the quality and walltime between our simulations and actual experiments. For more results please see Appendix D.1.5.

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-Estimator[6] in [45] under gate-count model, which adopts the improved list-decoding technique proposed in [38]. The estimated results are listed in Table 5. Our new concrete hardness estimation of LWE[7] answers Q7 in Section 5.3 of Kyber [8] and narrows the security estimation error interval. For more details, refer to the citation [29].

6.7 The Acceleration Effect of the Optimized Strategy

In practice, because of the advantages of multi-threading, the parameter c in the practical sieve time-cost model with $\beta \leq 124$ is lower than the theoretical value. Define the ratio r as

$$r := \frac{T_{\mathrm{PnJBKZ}}(\beta, 1)}{T_{\mathrm{PnJBKZ}}(\beta + \Delta\beta, J)} = \frac{J}{1 - \frac{\Delta\beta}{d-\beta} \cdot 2^{c \cdot \Delta\beta}} > 1.$$

As shown in the inequality above, since r becomes larger if c decreases, the strategy generated from PSSearch accelerates the efficiency in solving LWE Challenge significantly. For more details, see Figs. 6, 7 and Table 4.

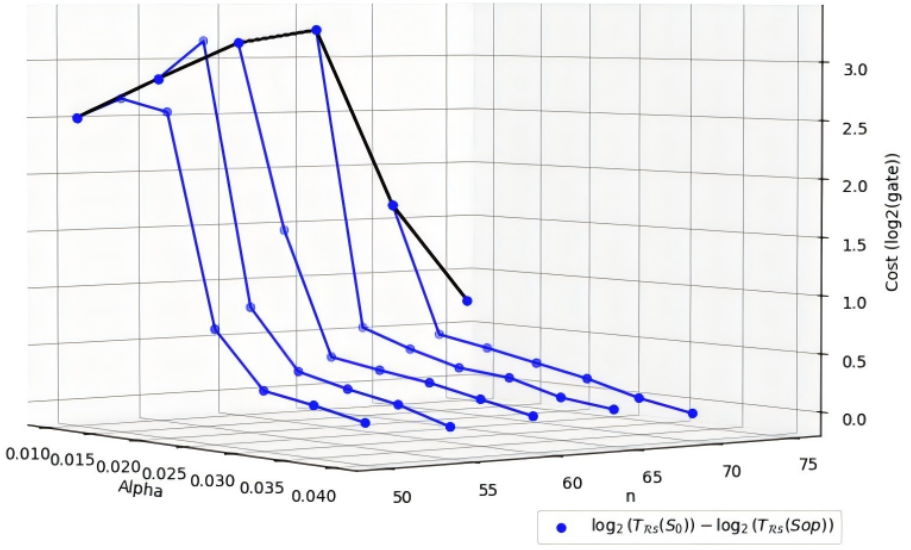


Fig. 8: Acceleration effect of the optimized strategy. LWE challenge instances with dimension and error rate parameters (n,α).

Fig. 8 shows that, (1) The black and blue lines indicate that as the LWE hardness parameters (dimension n, error rate α) become sufficiently large, the acceleration effect of multi-threading gradually diminishes;

[6] https://github.com/lducas/leaky-LWE-Estimator
[7] https://github.com/Summwer/lwe-estimator-with-PnJBKZ.git

(2) The blue lines also indicate that as α increases, the LWE instance approaches an SVP instance, then the acceleration effect of the optimized strategy diminishes. This is because most of the time cost becomes concentrated in the sieving process of the search step. Thus, in practice, the acceleration effect of the optimized strategy weakens while the LWE parameters increase.

However, even when the advantages of multi-threading diminish, 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. Three groups of security parameters (highlighted in Table 5) fail to meet the required security level for the designed standards set by NIST. Compared to the trivial Two-step solving strategy proposed in [29], the security levels decrease by 1.1 to 1.3 bits. See Table 5.

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

	$\log_2$ G/$\log_2$(gates)				$\log_2$ B/$\log_2$(bits)			$\Delta \log_2$ G	
	NIST Required [53]	Previous	Two-step		Previous	Two-step			
			S_0	S_{op}		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. "$\mathsf{S}_0 = \{(\beta_i = i + 2,\ J_i = 1) \mid i = 1, ..., \beta\}$" is a trivial progressive BKZ+Pump in Two-step mode to estimate security as [29] stated. "S_{op}" is a progressive Pn-JBKZ+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].

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, Tancrède, 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 gausssieve for speed," in *Progress in Cryptology - LATINCRYPT 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. 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.
42. Z. Zhao, J. Ding, and B.-Y. Yang, "BGJ15 revisited: Sieving with streamed memory access." Cryptology ePrint Archive, Paper 2024/739, 2024.
43. 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.
44. 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.
45. 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.
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. "National Institute of Standards and Technology, Submission require ments 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].
54. A. Leon-Garcia, *Probability, statistics, and random processes for electrical engineering*. Upper Saddle River, NJ: Pearson/Prentice Hall, 3. ed ed., 2008.
55. 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.
56. 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.
57. M. R. Albrecht, R. Player, and S. Scott, "On the concrete hardness of Learning with Errors," *Journal of Mathematical Cryptology*, vol. 9, Jan. 2015.
58. 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.
59. K. Xagawa, "Cryptography with Lattices." https://xagawa.net/pdf/2010Thesis.pdf, 2010.
60. 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

In this section, we provide detailed definitions of the various basic concepts used in this paper.

Definition 9 (The Gaussian Distribution [54]). *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 $\rho_{N(\mu,\sigma^2)} = e^{-\frac{(x-u)^2}{2\sigma^2}} / \sigma\sqrt{2\pi}$.*

Definition 10 (The discrete Gaussian Distribution [55]). *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 $f_{D(\mu,\sigma)} : \mathbb{Z} \to [0,1], x \to f_{N(\mu,\sigma)}(x)/f_{N(\mu,\sigma^2}(\mathbb{Z})$, where $f_{N(\mu,\sigma)}(\mathbb{Z}) = \sum_{x\in\mathbb{Z}} f_{N(\mu,\sigma)}(x)$.*

Definition 11 (Chi-Squared Distribution [54]). *Given n random variables $X_i \sim N(0,1)$, the random variables $X_0^2 + \cdots + X_{n-1}^2$ 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 $Y_0^2 + \cdots + Y_{n-1}^2$ 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$_\gamma$) [56]). *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_σ} Distribution [57–59]) 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 $(\mathbf{A}, \mathbf{b} = \mathbf{As} + \mathbf{e}) \in (\mathbb{Z}_q^{m\times n}, \mathbb{Z}_q^m)$ sampled as above.*

Definition 14 (Search LWE$_{m,n,q,D_\sigma}$ problem [57–59]). *Given a pair $(\mathbf{A}, \mathbf{b})$ sampled from LWE_{m,n,q,D_σ} distribution Ψ compute the pair $(\mathbf{s}, \mathbf{e})$.*

B Detailed Description of Algorithms

In this section, we provide a detailed description of some algorithms. First, Section B.1 will give a detailed description of the Refined SVPDimEst algorithm. Then, we will present the practical time-cost models for Pump and PnJBKZ in

Section B.2. Additionally, Section B.3 will describe the reduction strategy selection algorithm, called Blocksize and Jump Strategy Selection based on ProBKZ (BSSA). Finally, Section B.4 will illustrate the impact of optimizing the number of LWE samples when solving LWE.

B.1 Detailed Description of the Refined SVPDimEst

In this part, we provide a detailed description of the Refined SVPDimEst. Firstly the noise vector $\mathbf{e} \in \mathbb{Z}q^m$ of LWE follows a discrete Gaussian distribution $D0, \sigma$ with standard deviation σ. Thus, the probability distribution of the squared norm of the target vector in a β-dimensional sublattice follows $\sigma^2 \cdot \chi^2_\beta$.

The idea of treating the norm of the projected target vector on the sublattice as a random variable, rather than as an expected value, was first proposed in [45] for estimating the blocksize of each BKZ. We adapt this approach to the SVP call dimension by replacing the simulated Gram-Schmidt norms of BKZ with the Gaussian Heuristic value of the projected sublattice.

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

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

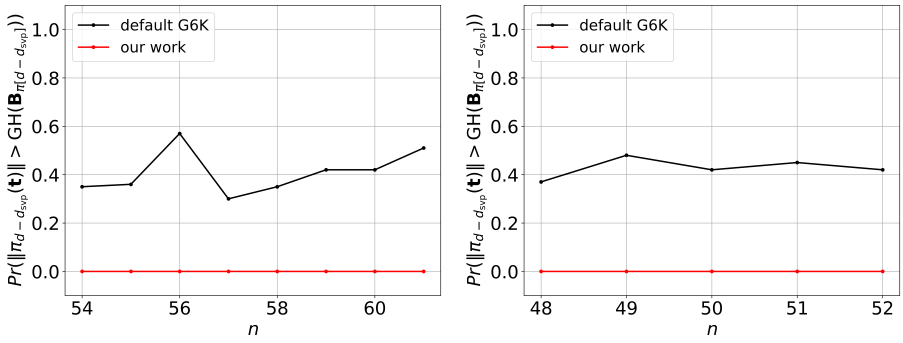
In Alg. 5, set $T_{\mathcal{C}}(d_{\text{svp}})$ as the estimated expected cost of the final SVP call. Considering the progressive SVP call, we calculate failure and success probabilities. The success probability of a β-dimensional progressive SVP call denoted as $P_{\text{suc}}(\beta)$ computed by line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

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

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

Fig. 9 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

($\geq$0.999) for recovering the target vector with a new estimated dimension in Alg. 5 to solve LWE problem.



(a) $\alpha = 0.010$ (b) $\alpha = 0.015$

Fig. 9: 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. $\|\pi_{d-d_{\mathrm{svp}}}(\mathbf{t})\| > \mathrm{GH}(\mathbf{B}_{\pi[d-d_{\mathrm{svp}},d]}) \geq \sigma\sqrt{d_{\mathrm{svp}}}$. The red line shows that using the estimated dimension computed by Alg. 5 in our work, the condition $\|\pi_{d-d_{\mathrm{svp}}}(\mathbf{t})\| \leq \mathrm{GH}(\mathbf{B}_{\pi[d-d_{\mathrm{svp}},d]})$ can always be satisfied.

Lemma 2 illustrates that, when using the chi-squared distribution to estimate the dimension of the SVP (`SVPDimEst`) needed to recover the target vector in LWE, a higher-quality lattice basis enables recovering the target vector by solving the SVP in a smaller dimension.

Lemma 2. *Under the Gaussian Heuristic, given an `SVPDimEst` as described in Alg. 5, consider two arbitrary lattice bases $\mathbf{C} \neq \mathbf{D}$ for the same d-dimensional lattice, generated from an LWE instance with standard deviation σ by the primal attack. Assume that $\mathbf{D} \geq_{\mathcal{Q}} \mathbf{C}$ and that $T_{\mathrm{SVP}}(d_{\mathrm{svp}})$ is a monotonically increasing function with respect to d_{svp}. Let $\mathbf{C}'$ (resp. $\mathbf{D}'$) be the lattice basis after calling a tour of $\mathcal{R}$-ξ on $\mathbf{C}$ (resp. $\mathbf{D}$), then $T_{\mathcal{C}}(\mathtt{SVPDimEst}(\mathsf{rr}(\mathbf{C}'), \sigma)) \geq T_{\mathcal{C}}(\mathtt{SVPDimEst}(\mathsf{rr}(\mathbf{D}'), \sigma))$.*

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

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 based on the definition of chi-squared distribution, $P(Y) := \Pr\left[x \leftarrow \sigma^2 \cdot \chi^2_{d_{\mathrm{svp}}} : x \leq Y\right]$ is an increasing function of Y. So, it follows that $\forall \beta \in [1, d]$, $P\left(\mathrm{GH}\left(\mathbf{C}'_{[d-\beta,d]}\right)^2\right) \leq P\left(\mathrm{GH}\left(\mathbf{D}'_{[d-\beta,d]}\right)^2\right)$ and there exists a minimum d_{svp} s.t. $P\left(\mathrm{GH}\left(\mathbf{C}'_{[d-d_{\mathrm{svp}},d]}\right)^2\right)$

$\leq 0.999 \leq P\left(\mathrm{GH}\left(\mathbf{D}'_{[d-d_{\mathrm{svp}},d]}\right)^2\right)$. Since $T_{\mathrm{SVP}}(d_{\mathrm{svp}})$ is monotonically increasing with respect to d_{svp}, and from Eq. (4), it follows that $T_{\mathcal{C}}(\texttt{SVPDimEst}\,(\mathsf{rr}(\mathbf{C}),\sigma)) \geq T_{\mathcal{C}}(\texttt{SVPDimEst}\,(\mathsf{rr}(\mathbf{D}),\sigma))$. □

We can still prove the correctness of PSSearch by using an adaptive version of Theorem 2 (denoted as Theorem 3). This is done by replacing the `SVPDimEst` method in Sec. 2.5 with Alg. 5 and proving Theorem 3 using Lemma 2, instead of Lemma 1.

Theorem 3. *Given the strategy space $\mathcal{S}$ (as defined in Def. 6) and `SVPDimEst` (as specified in Alg. 5), if Condition 1 holds, and $T_{\mathrm{SVP}}(d_{\mathrm{svp}})$ is monotonically increasing with respect to d_{svp}, then Alg. 3 (PSSearch) returns the reduction strategy from $\mathcal{S}$ that minimizes the time cost for solving the given LWE instance.*

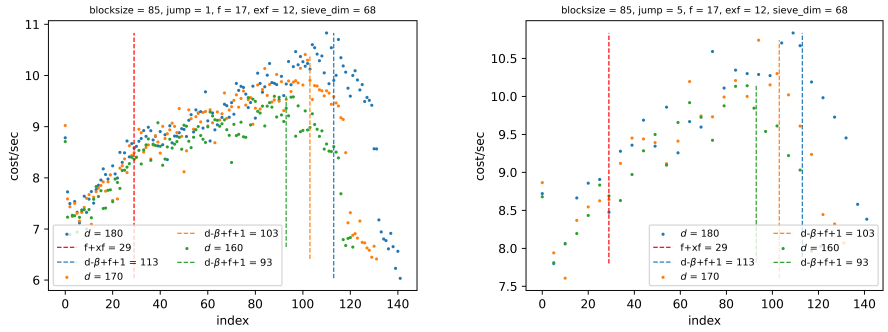
B.2 Practical Time-Cost model of Pump and PnJBKZ

In this section, we present the practical time-cost model for the Pump and PnJBKZ algorithms. An accurate time-cost model is a crucial component for determining the optimal reduction strategy that selects the progressive blocksize and jump size, minimizing the expected time cost in solving LWE.

The asymptotic complexity of the sieving does not match the actual cost well in the low dimensions (for dimensions ≤ 128) [6]. The multi-threading technology used in Pump helps balance some of the time cost increases as the dimension of sieving grows. Therefore, we construct a practical time-cost model by using the experimental method to test the running time of the Pump (described in Appendix B.2.1) on different lattice bases, to find the optimized reduction parameters of solving TU Darmstadt LWE challenges with a shorter time cost.

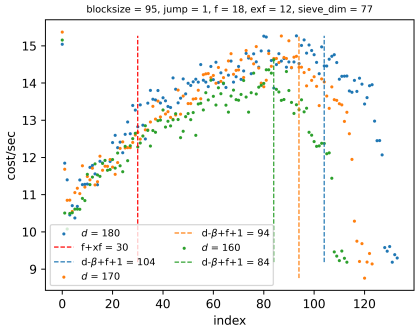
Although the time-cost model based on experimental results fits the actual cost of running PnJBKZ well, using testing machines with different configurations will inevitably cause changes in the time-cost model in low-dimensions. Therefore, we use this experimentally constructed time-cost model only when searching for the optimized progressive blocksize and jump size selection strategy for minimize the time cost of solving LWE challenges.

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

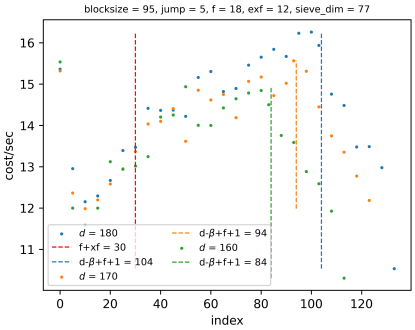


(a) PnJBKZ(85,1)

(b) PnJBKZ(85,5)



(c) PnJBKZ(95,1)



(d) PnJBKZ(95,5)

Fig. 10: 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.

Additionally, when constructing the actual time-cost model by testing PnJBKZ on a specific machine, we find that each Pump in PnJBKZ has a different time cost, as shown in Fig. 10. In particular, the time cost of the first Pump is higher than that of subsequent Pumps. It increases with the incremental index from 2^{nd} to $(d-\beta+f+1)^{\text{th}}$, then decreases after $d-\beta+f+1$ index.

This implies that for a fixed blocksize β, the average Pump cost in PnJBKZ increases as the lattice basis dimension d grows. This means that the simplified model, which treats each SVP oracle inside BKZ as having the same time cost no longer holds in the context of PnJBKZ. Therefore, in Appendix B.2.2, we propose a new time-cost model for PnJBKZ that more accurately reflects its time cost performance in practical applications.

B.2.1 Practical Cost Model of Pump We can regard T_{Pump} as the computational cost model for the d_{svp}-dimensional progressive sieve. T_{Pump} in [20]

is considered as

$$T_{\text{Pump}}(d_{\text{svp}}) = \sum_{j=\beta_0}^{d_{\text{svp}}} T_{\text{sieve}}(j) = \sum_{j=\beta_0}^{d_{\text{svp}}} 2^{c\cdot j+o(j)} = 2^{c\beta_0}\left(1+2^{c}+\cdots+2^{c\left(d_{\text{svp}}-\beta_0\right)}\right) \\ \leq 2^{c\beta_0}\cdot\frac{2^{c\left(d_{\text{svp}}+1\right)+o\left(d_{\text{svp}}+1\right)}}{1-2^{c}} = O\left(2^{cd_{\text{svp}}}\right) \approx 2^{cd_{\text{svp}}+c_1}, \tag{5}$$

where β_0 is the initial sieving dimension 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 related to the full sieve cost and sieve dimension, and $T_{\text{sieve}}(j)$ is the sieve cost with dimension j.

However, we find that the asymptotic complexity of sieving does not match the actual cost well in low dimensions. While the dimension is low, the number of threads used in the Pump increases with the dimension, balancing part of the time cost increase. In low dimensions, c might be much lower than the theoretical result.

To accurately predict the unknown coefficients c and c_1 in our computational cost model, we use an experimental method to test the running time of Pump with different sieving dimensions on the projected lattice bases of the 180-dimensional SVP Challenge[8] and with different blocksizes βs. The experimental results show that our computational cost models above can fit well with the actual cost of Pump.

Let β be the independent variable, and $\log_2(T_{\text{Pump}})$ can be obtained from the experimental test as the dependent variable. 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 in the linear regression model. The coefficient of determination (R^2 or R squared) is a statistical measure in a regression model that indicates the proportion of variance in the dependent variable explained by the independent variable. Generally, the range of R^2 is $[0, 1]$ and when R^2 is closer to 1, the better the model fits the data.

From Figure 11, we observe that R^2 is close to 1, indicating a good fitting effect. Figure 11 also shows that the logarithm of Pump's computational cost is linearly correlated to d_{svp} for 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.

B.2.1.1 Consider Cost of DualHash Generation.
Moreover, for a given sieve dimension β, the G6K-GPU-Tensor implementation requires $O(\beta)$ memory and computational cost to generate the DualHash value 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 $c_2 = 0.2075$ 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 (shown in Fig. 12 shown) through the curve fitting method.

[8] https://www.latticechallenge.org/svp-challenge

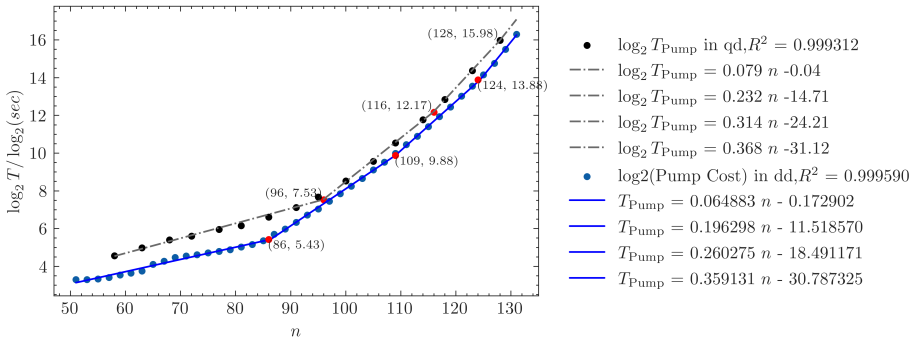


Fig. 11: Pump Cost Figure while $d = 180$, Sieve used in Pump is gpu_sieve, and it's running on Machine C with 2 GPUs and 32 threads: Relation between $\log_2(T_{\mathrm{Pump}})$ and sieve dimension n.

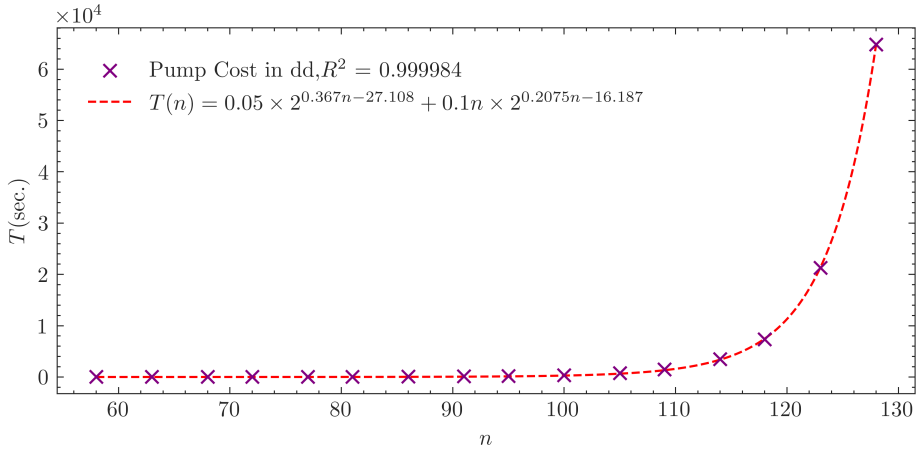


Fig. 12: Pump cost model considers the cost to generate the hash value: The vertical axis represents T_{Pump} and the horizontal axis represents the sieve dimension n.

B.2.2 Practical Cost Model of PnJBKZ PnJBKZ consists of a series of Pumps. Assuming PnJBKZ as a combination of Pumps with equal time cost, its computational cost can be calculated as the sum of $\frac{d+2f-\beta}{J}$ progressive sieves on the $(\beta - f)$-dimension projected sublattice with jump J.

However, as shown in Fig. 10, each Pump in PnJBKZ has a different cost. Specifically, the Pump cost increases from 2^{nd} to $(d-\beta+f+1)^{\text{th}}$ index and then decreases. Figure 10 shows that the growth rate in the range of $[0, f + f_{\text{extra}}]$ differs from that in $[f+f_{\text{extra}}, d-\beta+f+1]$, where f_{extra} is the extra dimension-for-free value in G6K, set to 12 in the default setting to enhance PnJBKZ's efficiency.

So we divide the PnJBKZ cost into 4 parts: the first index of Pump, the preceding indices in range $[0, f+f_{\text{extra}})$, the middle indices in range $[f+f_{\text{extra}}, d-\beta+f+1)$ and the later indices in range of $[d-\beta+f+1, d)$. Let the cost of each range be denoted as T_{first}, T_{pre}, T_{mid} and T_{former}. Let $T_{f+f_{\text{extra}}}$ and $T_{d-\beta+f+1}$ represent the Pump cost at the indices $f + f_{\text{extra}}$ and $T_{d-\beta+f+1}$ respectively. We tested T_{first}, T_{pre}, T_{later} and the coefficients A and B in the formula

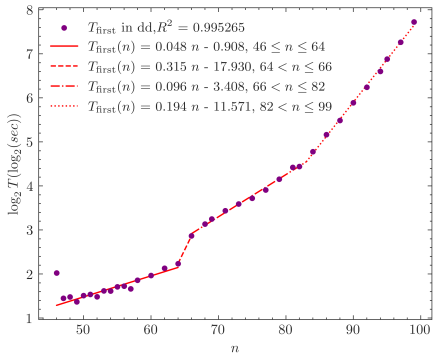
$$\begin{aligned} T_{\text{mid}}(d, \beta, J, f, f_{\text{extra}}) =& 1/2 \cdot (T_{f+f_{\text{extra}}} + T_{d-\beta+f+1}) \cdot (d - \beta - f_{\text{extra}} + 1) \\ =& 1/2 \cdot ((A \cdot (f + f_{\text{extra}}) + B) + (A \cdot (d - \beta + f) + B)) \\ & \cdot (d - \beta - f_{\text{extra}} + 1) \end{aligned}$$

in dimension $d = 180$, jump $J = 1$ and "dd" float type. The simulated cost model is shown in Fig. 13. Then, we can derive that

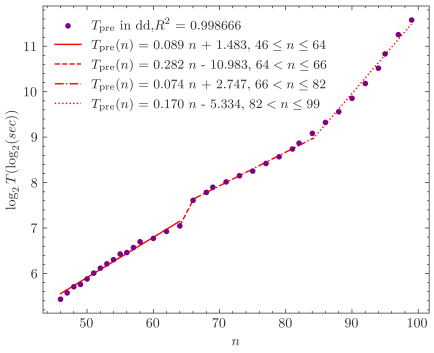
$$\begin{aligned} T_{\text{PnJBKZ}}(d, \beta, J, f, f_{\text{extra}}) = T_{\text{first}} &+ T_{\text{pre}} \cdot \frac{\lceil \frac{f+f_{\text{extra}}}{J} \rceil - 1}{f + f_{\text{extra}} - 1} \\ &+ T_{\text{mid}}(d, \beta, J, f, f_{\text{extra}}) \cdot \frac{\lceil \frac{f+f_{\text{extra}}}{J}) \rceil}{f + f_{\text{extra}}} \\ &+ T_{\text{later}} \cdot \frac{\lceil \frac{d-\beta-f_{\text{extra}}}{J} \rceil + 1}{d - \beta - f_{\text{extra}} + 1}, \end{aligned} \tag{6}$$

where f is the dimension for free value of β.

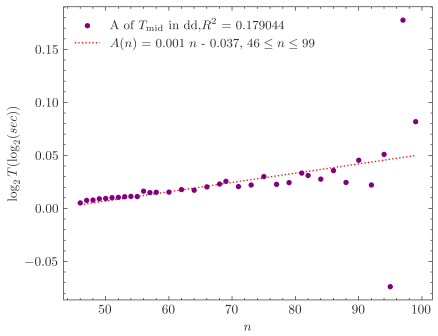
We have also used Eq. 6 to simulate the PnJBKZ cost of other dimensions (such as $d = 160$ and 170) with blocksize from 51 to 119 and jump $J \geq 1$, and found that it fits well in the simulation, as shown in Figure 14.



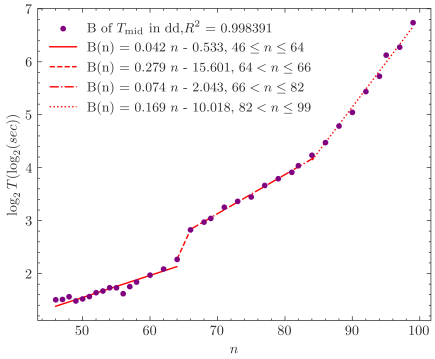
(a) T_{first}



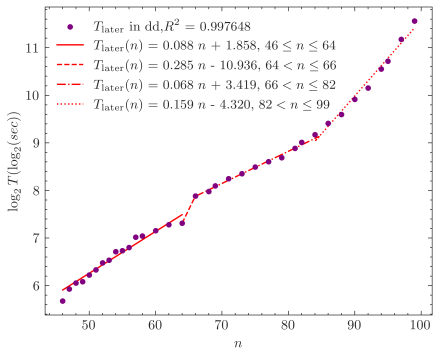
(b) T_{pre}



(c) A of T_{mid}

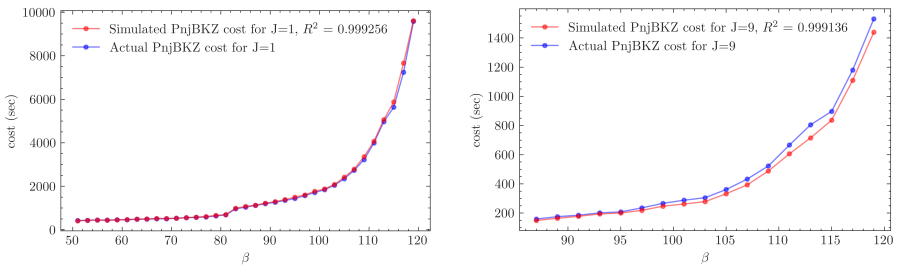


(d) B of T_{mid}

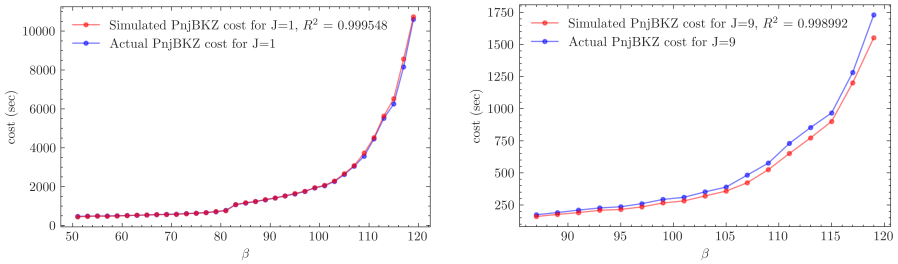


(e) T_{later}

Fig. 13: 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 GPUs $= 2$ and 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.



(a) $(d, J) = (160, 1)$ (b) $(d, J) = (160, 9)$



(c) $(d, J) = (170, 1)$ (d) $(d, J) = (170, 9)$

Fig. 14: Simulate each PnJBKZ Cost using Eq. (13) 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.

B.3 Blocksize and Jump Strategy Selection based on ProBKZ

In this section, we provide a detail description of the Blocksize and Jump Strategy Selection Algorithm based on ProBKZ (BSSA). The *blocksize and jump strategy selection algorithm based on ProBKZ [19]* (BSSA, Fig. 15) uses the *Shortest Path Algorithm* to select the optimized reduction strategy.

BSSA begins 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 intermediate nodes (such as $\beta^{\text{sstart}} = \beta_i$, for $\beta^{\text{start}} < \beta_i < \beta^{\text{goal}}$) from β^{start} to β^{goal} as measures of lattice basis quality. For the 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, $\mathsf{BS}[\beta^{\text{goal}}]$, where the key is each middle node β_i and the value is a tuple of $\mathsf{bs} = (\mathsf{rr}, \mathsf{S}, T_{\text{PnJBKZs}}, \text{PSC})$. Here rr is the length of Gram-Schmidt vector after fully BKZ-

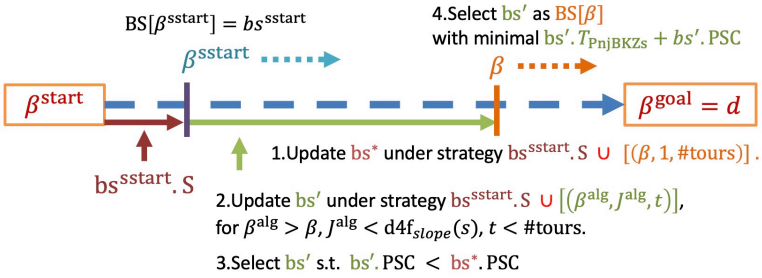


Fig. 15: BSSA Process.

β^{goal} reduction, S represents the blocksize and jump selection strategy that will improve the lattice basis quality from fully BKZ-β^{start} reduced basis to a fully BKZ-β^{goal} reduced basis.

This strategy is a combination of $(\beta^{alg}, J^{alg}, t, T_{PnJBKZ})$ stored on each edge in the shortest path from node β^{start} to β^{goal}, where the sum of the simulated BKZ cost, $T_{PnJBKZs} = \sum_{\beta^{alg}, J^{alg}, t} T_{PnJBKZ}(\beta^{alg}, J^{alg}, t)$, is minimized. The shortest path can be found using Dijkstra algorithm. PSC is one of the outputs from the Pump dimension estimation method (Alg. 5), representing the estimated time cost for solving uSVP$_\gamma$ by processing Pump on the BKZ-β^{goal} reduced basis.

By setting different final values of β^{goal}, we can obtain different reduction strategies BS that improve the lattice basis quality from β^{start} to β^{goal}, along with different sieving dimension of the last Pump corresponding to the varying quality of the fully β^{goal} reduced lattice basis. Then, by considering multiple different final β^{goal} values, we select the Two-step solving strategy with the minimum total time cost. This total time cost includes the time cost of improving the lattice basis quality via a series of PnJBKZ$(\beta, J) \in$ S and the time cost of the final Pump. For more details on BSSA, see Alg. 6 .

B.4 Choosing the number of LWE Samples

In this section, we will demonstrate the impact of optimizing the number of LWE samples and the reduction strategy on solving the LWE problem.

BKZ-only mode is the mainstream method for estimating the security of LWE-based cryptosystem currently. It employs Kannan's Embedding technique to reduce the LWE problem to the uSVP$_\gamma$ problem and utilizes the GSA assumption to simulate the change of a BKZ-β reduction. This evaluation method was first proposed by Erdem Alkim et al. in [33] and its correctness was later proven in [60], which has provided a lower bound of LWE samples and the blocksize β. We have renamed this method as "2016 Estimation from GSA for LWE" (referred to as 2016 Estimate).

To solve the LWE problem, the first step is to determine the number of LWE instances required to construct the lattice basis in the primal attack. In the 2016

Estimate [18], the strategy for selecting the number of LWE instances, denoted as m, is to find the value of m such that the following inequality holds, while minimizing the value of β. Let $d = m + 1$, n represent the dimension of LWE instance, then

$$\min_{\beta \in \mathbb{N}} \left\{ T_{\mathrm{BKZ}}(\beta) : \sigma\sqrt{\beta} \leq \delta\left(\beta\right)^{2\beta-d-1} \cdot q^{\frac{d-n-1}{d}} \right\}. \tag{7}$$

The strategy in the 2016 Estimate is to determine m such that the LWE problem can be solved with the minimum time cost, using a fixed blocksize of BKZ-β algorithm.

input : rr_0, $F(\star, \mathcal{D})$, $\beta^{\mathrm{start}} \leftarrow 50$, $J_{\max}(\star) \leftarrow \mathrm{d4f}(\star)/2$;
output: $T_{\min}$, $S_{\min}$;
Function `BSSA`(rr_0, $F(\star, \mathcal{D})$, $\beta^{\mathrm{start}} \leftarrow 50$)**:**
  $d \leftarrow \mathrm{len}(\mathsf{rr}_0)$; $\mathrm{PSC}^{(0)} \leftarrow$ `ProSieveDimEst`(rr_0, $F(\star, \mathcal{D})$);
  $\mathsf{BS}[\beta^{\mathrm{start}}] = (\mathsf{rr}_0, [\,], 0, \mathrm{PSC}^{(0)})$;
  for $\beta \leftarrow \beta^{\mathrm{start}}$ **to** d **do**
    $T^{(\min)}_{\mathrm{PnJBKZs}} \leftarrow +\infty$;
    for $\beta^{\mathrm{sstart}} \leftarrow \beta^{\mathrm{start}}$ **to** $\beta - 1$ **do**
      $\mathsf{bs}^{\mathrm{sstart}} \leftarrow \mathsf{BS}\,[\beta^{\mathrm{sstart}}]$; $\mathsf{bs} \leftarrow (\emptyset, \emptyset, +\infty, +\infty)$;
      Update bs^* under strategy $\mathsf{bs}^{\mathrm{sstart}}.\mathsf{S} \cup [(\beta, 1, \sharp\mathrm{tours}(\mathsf{bs}^{\mathrm{sstart}}.\mathsf{rr}, \mathrm{BKZ}\text{-}\beta)]$;
      for $\beta^{\mathrm{alg}} \leftarrow \beta + 1$ **to** d **do**
        for $j \leftarrow J_{\max}(\beta^{\mathrm{alg}})$ **to** 1 **do**
          $T' \leftarrow +\infty$;
          for $t \leftarrow 1$ **to** $\sharp\mathrm{tours}(\mathsf{bs}^{\mathrm{sstart}}.\mathsf{rr}, \mathrm{PnJBKZ}\text{-}(\beta^{\mathrm{alg}}, j))$ **do**
            Update bs' under strategy $\mathsf{bs}^{\mathrm{sstart}}.\mathsf{S} \cup [(\beta^{\mathrm{alg}}, j, t)]$;
            if $\mathsf{bs}'.\mathrm{PSC} < \mathsf{bs}^*.\mathrm{PSC}$ **then**
              $T' \leftarrow \mathsf{bs}'.T_{\mathrm{PnJBKZs}}$;
              break;
          if $\mathsf{bs}.T_{\mathrm{PnJBKZs}} > T'$ **then**
            $\mathsf{bs} \leftarrow \mathsf{bs}'$;
      if $T^{(\min)}_{\mathrm{PnJBKZs}} > \mathsf{bs}.T_{\mathrm{PnJBKZs}}$ **then**
        $T^{(\min)}_{\mathrm{PnJBKZs}} \leftarrow \mathsf{bs}.T_{\mathrm{PnJBKZs}}$; $\mathsf{BS}\,[\beta] \leftarrow \mathsf{bs}$;
  $\mathsf{bs}_{\min} \leftarrow \min_{\mathsf{bs}.T_{\mathrm{PnJBKZs}} + \mathsf{bs}.\mathrm{PSC}} \mathsf{BS}$;
  return $T_{\min} \leftarrow \mathsf{bs}.T_{\mathrm{PnJBKZs}} + \mathsf{bs}.\mathrm{PSC}$, $\mathsf{S}_{\min} \leftarrow \mathsf{bs}_{\min}.\mathsf{S}$;

Algorithm 6: BSSA

In G6K, the estimation method simulates a two-stage strategy. The main difference from ours is that its two-stage strategy involves two tours of PnJBKZ with a fixed blocksize β simulated from GSA assumption, and a progressive sieve algorithm in dimension d_{svp}. It simulates this scenario and aims to find the minimal cost for the pair $(\beta, d_{\mathrm{svp}})$ from

$$\min_{\beta, d_{\mathrm{svp}} \in \mathbb{N}} \left\{ 2 \cdot T_{\mathrm{BKZ}}(\beta) + \mathrm{PSC}(d_{\mathrm{svp}}) : \|\pi_{d-d_{\mathrm{svp}}}(\mathbf{v})\| \leq \mathrm{GH}(\mathcal{L}_{\pi[d-d_{\mathrm{svp}}]}) \right\}, \tag{8}$$

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

However, as explained in Sec. 4.3, the 2016 Estimate still carries a probability of failure to find the target vector through its estimation. Therefore, our strategy for solving the LWE problem involves 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, PnJBKZ simulator is used to simulate the lattice basis after a series of PnJBKZ steps. In the second stage, the goal is to find the unique shortest vector using by the Pump algorithm. 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. Additionally, we employ the new Pump estimation scheme to simulate the norm of the target vector.

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

$$\min_{\beta, d_{\text{svp}} \in \mathbb{N}} \left\{T_{\text{PnJBKZs}}\left(\mathbf{B}\right) + \text{PSC}\left(d_{\text{svp}}\right) : P(d_{\text{svp}}) \geq P_{\text{success}}\right\}, \tag{9}$$

where δ is the basis quality after PnJBKZs. $T_{\text{PnJBKZs}}\left(\mathbf{B}\right)$ will respectively call PSSearch 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 that minimizes both β and d_{svp} satisfying the inequality (9). The concrete process is described in the Algorithm 7.

By using the optimization strategy for selecting the number of LWE instances, we can solve LWE challenges faster than the G6K default strategy. However, the efficiency improvement is not significant (at most 2.2% in the test). See the Table 6.

C Proofs of Theorems and Lemmas

In this section, we will give the proofs of **Theorem 1** and **Lemma 1**.

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, there exists a minimum value d_{svp} such that $d_{\text{svp}} \in$

input: n, q, α, m_{all}, β_{bound}, $d^{(\text{svp})}_{\text{bound}}$, τ, $\mathbf{A}^{m_{all}\times n}$, $\mathbf{b}^{m_{all}\times 1}$;
output: $S_{\min}$, $T_{\min}$, m;
$\sigma, T_{\min}, \mathsf{mRange} \leftarrow \alpha q, +\infty, \{\}$;
$m_0 \leftarrow$ LWE samples estimation in G6K as formula (8);
$m_{\min} \leftarrow \min\{m$ **satisfies** equation (9)$\}$; $S_{\min}$, $T_{\min} \leftarrow$ None, None;
while $\tau > 0$ **do**
  Construct $\mathbf{B}$ by $\left(\mathbf{A}^{m_0\times n}, \mathbf{b}^{m_0\times 1}, q\right)$;
  $m_1 \leftarrow m_0$;
  for $m \in \{\max\{m_{\min}, m_0 - \tau\}, m_0, \min\{m_{all}, m_0 + \tau\}\}$ **do**
    $d \leftarrow m + 1$, $M \leftarrow \sigma^2 m + 1$;
    Construct $\mathbf{B}$ by $\left(\mathbf{A}^{m\times n}, \mathbf{b}^{m\times 1}, q\right)$;
    T_{total}, $\mathsf{S} \leftarrow \mathtt{PSSearch(rr(}\mathbf{B}\mathtt{)}, \sigma^2\chi^2_\star\mathtt{)}$;
    if $T_{\min}$ is None or $T_{\min} < T_{\text{total}}$ **then**
      $\mathsf{S}_{\min}$, $T_{\min}$, $m_1 \leftarrow \mathsf{S}$, T_{total}, m;
  if $m_1 = m_0$ **then**
    $\tau \leftarrow \lfloor \frac{\tau}{2} \rfloor$;
  $m_0 \leftarrow m_1$;
return $S_{\min}$, $T_{\min}$, m_0;

Algorithm 7: Our LWE Samples Number Selection Algorithm

Table 6: LWE samples improvement simulated result generated by PSSearch 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%

$[1, d+1]$, and the target vector $\mathbf{v}$ can be found under the ξ-reduced basis. This is 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, the each set S is finite. Thus, $\mathcal{S}$ is finite. □

C.2 Order Preservation for SVPDimEst

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

Proof. From Property 1 and the condition $\mathbf{D} \geq_{\mathcal{Q}} \mathbf{C}$, it follows that $\mathbf{D}' \geq_{\mathcal{Q}} \mathbf{C}'$. Since $|\mathbf{C}| = |\mathbf{D}|$, we obtain $\mathrm{GH}(\mathbf{C}'_{\pi[k,d]}) \leq \mathrm{GH}(\mathbf{D}'_{\pi[k,d]})$, $\forall k \in [d]$. From the description of `SVPDimEst` in Sec. 2.5, $d_{\text{svp}} := \texttt{SVPDimEst}(\mathbf{D})$ is the smallest integer satisfying $\sigma\sqrt{d_{\text{svp}}} \leq \mathrm{GH}(\mathbf{D}_{\pi[d-d_{\text{svp}},d]})$. Then there are only two possible cases: (1) $\mathrm{GH}(\mathbf{C}'_{\pi[d-d_{\text{svp}},d]}) < \sigma\sqrt{d_{\text{svp}}} \leq \mathrm{GH}(\mathbf{D}'_{\pi[d-d_{\text{svp}},d]})$, we have $\texttt{SVPDimEst}(\mathsf{rr}(\mathbf{C}'), \sigma) > \texttt{SVPDimEst}(\mathsf{rr}(\mathbf{D}'), \sigma)$; (2) $\sigma\sqrt{d_{\text{svp}}} \leq \mathrm{GH}(\mathbf{C}'_{\pi[d-d_{\text{svp}},d]}) \leq \mathrm{GH}(\mathbf{D}'_{\pi[d-d_{\text{svp}},d]})$, we have $\texttt{SVPDimEst}(\mathsf{rr}(\mathbf{C}'), \sigma) = \texttt{SVPDimEst}(\mathsf{rr}(\mathbf{D}'), \sigma)$. Thus, we get $\texttt{SVPDimEst}(\mathsf{rr}(\mathbf{C}'), \sigma) \geq \texttt{SVPDimEst}(\mathsf{rr}(\mathbf{D}'), \sigma)$. □

D Experiments

In this section, we present all the verification experiments and provide further details about them. First, Section D.1 showcases the accuracy verification experiments for the PnJBKZ simulator. Additionally, Section D.2 demonstrates the improvement in solving LWE after applying our optimized reduction strategy, which is computed using either PSSearch (Algorithm 3) or BSSA. Furthermore, we will present the optimized reduction strategies for solving LWE challenges in Section D.3.

D.1 PnJBKZ Simulator Accuracy Verification Experiments

In this section, we first demonstrate in D.1.1 that the PnJBKZ Simulator can accurately predict the reduction effect of PnJBKZ in the ideal case, where each pump used by PnJBKZ outputs a HKZ-reduced lattice basis. Then, D.1.2 and D.1.3 illustrate how to simulate PnJBKZ using d4f technology in practice. Finally, D.1.4 and D.1.5 present additional accuracy verification experiments for the PnJBKZ simulator when employing d4f technology in practice.

D.1.1 Ideal version of 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

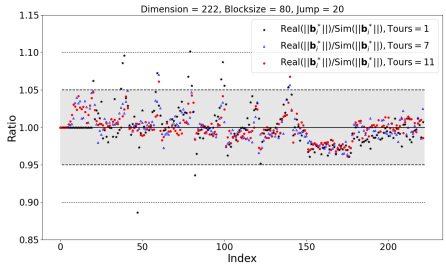
behind our construction of the PnJBKZ simulator is correct. Specifically, we demonstrate that when the jump parameter J of PnJBKZ is smaller than the blocksize β, the PnJBKZ simulator, constructed using the properties of HKZ-reduced bases and Gaussian heuristics, is reasonable and can accurately predict the actual reduction effects of PnJBKZ.

To practically ensure that each Pump outputs an HKZ-reduced basis, it is necessary to forgo the d4f technique. Without this adjustment, a Pump will only perform $\beta - \mathrm{d4f}(\beta)$ dimensional progressive sieving during the Pump-up stage, and can thus perform at most $\beta - \mathrm{d4f}(\beta)$ embeddings during the Pump-down stage. Consequently, the output basis of each β-dimensional Pump will inevitably include $\mathrm{d4f}(\beta)$-dimensional GS vectors that do not meet the HKZ-reduced basis properties. Thus, in this subsection D.1.1, 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 resieving during the Pump-down stage.

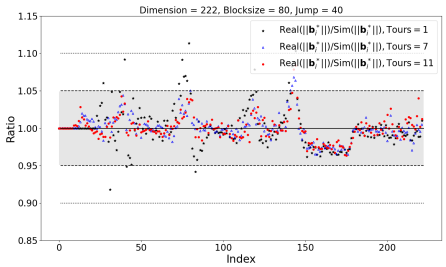
Additionally, since the LLL reduction is applied after each pump step during one tour of reduction in PnJBKZ-(β, J), the Gram-Schmidt (GS) values obtained from the experiments exhibit a smooth downward trend. To improve the prediction accuracy of the PnJBKZ simulator, we incorporated the size-reduced and Lovász conditions of the LLL-reduced basis into the original version of the PnJBKZ simulator.

We calculate $\mathrm{Sim}(l_i'')$ based strictly on the properties of the HKZ-reduced basis. Similar to classical BKZ simulators [17, 36], which assess the accuracy of BKZ simulators, we use the ratio $l_i''/\mathrm{Sim}(l_i'')$ for $i \in [0, d-1]$ in each tour of the PnJBKZ reduction as a criterion for measuring the accuracy of the PnJBKZ simulator. Refer to Fig. 16, 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 $(\beta, J, \mathrm{tours})$, and $\mathrm{Sim}(l_i'')$ is the simulated logarithm of these Gram-Schmidt vector lengths, calculated using Eq. (3).

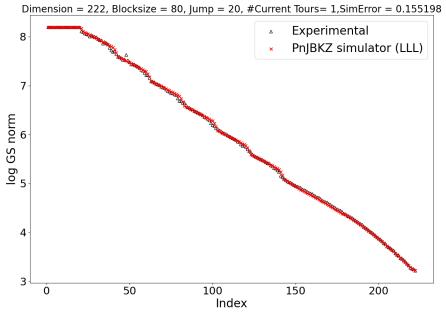
When using the classic Chen-Nguyen simulator to predict the actual reduction effect of BKZ, which employs pruned enumeration as the SVP oracle, most ratios fall within the range [0.95, 1.05]. Our simulation results are similar to those of the classic Chen-Nguyen simulator, with the remaining ratios falling within the range of [0.85, 1.15]. For compare, refer to Fig. 1 in Section 4.3 of the BKZ 2.0 paper [17] with Fig. 16 in our paper. Thus, 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 presented in Fig. 16.



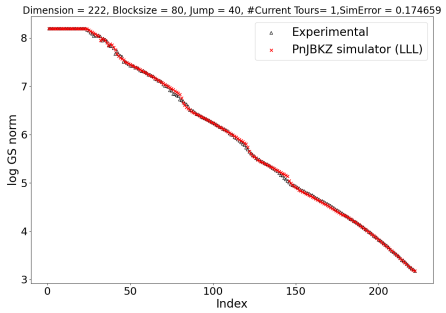
(a) β=80, jump=20, Average Error Ratio=1.8%,



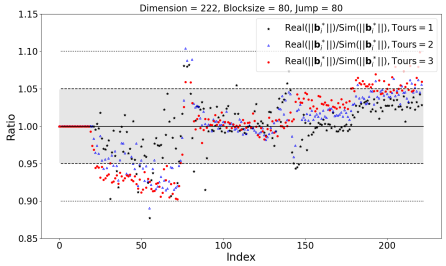
(b) β=80, jump=40, Average Error Ratio=1.94%,



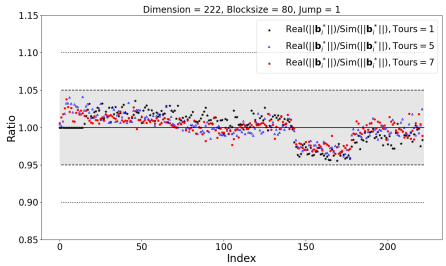
(c) β=80, jump=20



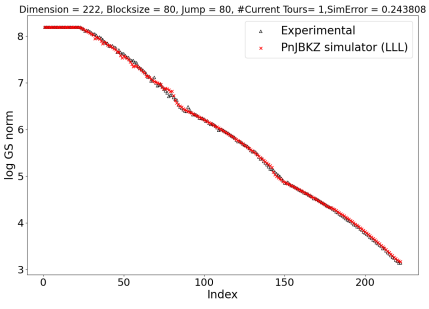
(d) β=80, jump=40



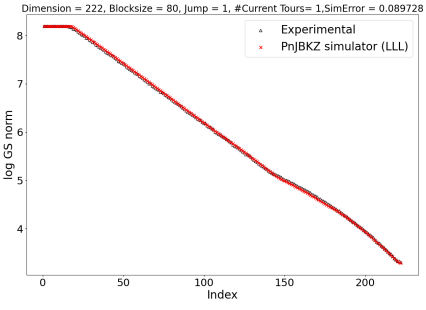
(e) β = 80, jump=80, Average Error Ratio=2.29%,



(f) β=80, jump=1



(g) β=80, jump=80



(h) β=80, jump=1

Fig. 16: Ratio $l_i''/\mathrm{Sim}(l_i'')$ and corresponding overall prediction effect of PnJBKZ simulator. Run PnJBKZ(β, J) reduction on a 222-dimension LWE lattice basis (n=60,α=0.010) and record the ratio values. We test 20 times for each reduction parameter.

D.1.2 Predicating PnJBKZ using d4f technology The ideal PnJBKZ simulation discussed in Section 5.1 did not consider the influence of using the dimension-for-free (d4f) technology. In practice, the implementation of PnJBKZ used in [20] and [28] default to using 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, [26] provides two theoretical d4f estimations for solving β-dimension SVP as $\mathrm{d4f}(\beta) = \beta \ln(4/3)/\ln(\beta/2\pi)$ and $\mathrm{d4f}(\beta) = \beta \ln(4/3)/\ln(\beta/2\pi e)$, while in the implementation of G6K [20], it gives a more relaxed value and we called it "optimistic d4f":

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

After using d4f to accelerate PnJBKZ, each Pump in the algorithm performs progressive sieving only in a $(\beta - d4f)$-dimensional space during the Pump-up stage. As a result, it can 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 using d4f technology, including optimistic d4f values, we employ a refined d4f value estimation proposed in [43]. Additionally, we modify our simulation strategy to improve the accuracy of the simulator in predicting the behavior of PnJBKZ reductions that utilize d4f technology and optimistic d4f settings. In this Appendix D.1.2, along with the subsequent verification experiments, we provide a detailed analysis of these adjustments and their impact on the overall performance. Appendix D.1.3, and D.1.4, all PnJBKZ reduction experiments use the "optimistic d4f" settings applied in the G6K implementation [20].

Wang et al. in [43] proposed a more refined d4f value estimation function based on the quality of the current lattice basis. Specifically, we first revisit the estimation of the upper bound of the d4f value under the pessimistic condition outlined in [26]. The formula for this pessimistic estimation is given by:

$$\mathrm{GH}(\mathcal{L}) \leq \mathrm{GH}\left(\mathcal{L}_{\pi[f,d]}\right) \sqrt{4/3} \tag{10}$$

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:

$$f \approx \frac{d \ln(4/3)}{\ln(d/2\pi)} \tag{11}$$

The optimistic condition in [26]:

$$\mathrm{GH}\left(\mathcal{L}\right)\sqrt{\frac{d-f}{d}} \leq \mathrm{GH}\left(\mathcal{L}_{\pi[f,d]}\right)\sqrt{\frac{4}{3}}. \tag{12}$$

Using the GSA and the optimistic asymptotic value of d4f, we can rewrite f as:

$$f \approx \frac{d\ln\left(4/3\right)}{\ln\left(d/2\pi e\right)} \tag{13}$$

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 lattice reduction process, the reduction quality of the lattice basis improves gradually rather than remaining constant. Especially in the initial reduction stage, when the lattice basis quality is far from being fully-BKZ-$d/2$ reduced. The effect of this gradual improvement in lattice reduction quality on d4f is not considered in Eq.(12) and Eq.(13).

Thus, the two different maximum values of d4f analyzed in [26] depend solely on the lattice dimension and serve as asymptotic upper bounds rather than accurate estimations. Our goal is to provide a refined estimation of the maximum value of d4f, based on the GSA coefficient.

Since under the Gaussian Heuristic, $\lambda_1\left(\mathcal{L}_{\pi[f+1,d]}\right) = \mathrm{GH}\left(\mathcal{L}_{\pi[f+1,d]}\right) \approx \sqrt{\frac{d-f}{2\pi e}}\cdot\left(\prod_{i=f+1}^{d}\|b_i^*\|\right)^{\frac{1}{d-f}}$. By applying the GSA, we obtain:

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

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

$$\lambda_1\left(\mathcal{L}_{\pi[f+1,d]}\right) = \sqrt{\frac{d-f}{2\pi e}}\frac{\left|\det\left(\mathcal{L}\right)\right|^{\frac{1}{d}}}{\delta^f} \tag{14}$$

Based on the pessimistic condition in [26]: $\mathrm{GH}\left(\mathcal{L}_{\pi[f+1,d]}\right)\sqrt{4/3} \geq \mathrm{GH}\left(\mathcal{L}\right)$ and Eq. (14), we can get a conservative d4f estimation based on GSA:

$$f \leq \log_\delta\sqrt{\frac{4\left(d-f\right)}{3d}} \tag{15}$$

Based on the optimistic condition in [26]: $\mathrm{GH}\left(\mathcal{L}_{\pi[f+1,d]}\right)\sqrt{4/3} \geq \pi_f(\mathrm{GH}\left(\mathcal{L}\right))$ and Eq. (14), we can get an optimistic d4f estimation based on GSA:

$$f \leq \log_\delta\sqrt{\frac{4}{3}} \tag{16}$$

It illustrates that, under the current reduction quality of the lattice basis, the maximum d4f value should be $\mathrm{d4f}_\delta = \ln_\delta\sqrt{4/3} \approx \ln\left(4/3\right)/(-\mathrm{slope})$. Here,

slope represents the slope of the logarithm of the Gram-Schmidt norms l_i for $\forall i \in \{0, ..., d-1\}$.

This leads to a problem: the actual d4f value that can be achieved under a certain reduction quality of the lattice basis should, in theory, be $\mathrm{d4f}_\delta$. If the Jump size is not limited, there may be a situation where $\mathrm{d4f}_\delta <$ Jump $<$ the optimistic heuristic d4f value. In this case, the first (Jump $- \mathrm{d4f}_\delta$) GS values of each Pump can no longer be guaranteed to be the shortest vector on the corresponding projection sublattice. As a result, the prediction of the first (Jump $- \mathrm{d4f}_\delta$) GS values of each Pump based on the HKZ property is no longer accurate.

To ensure PnJBKZ simulator can still 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 2. Later, our experiments will demonstrate that, in practice, when using optimistic d4f value during PnJBKZ reduction, Heuristic 2 indeed holds under the appropriate reduction parameter.

Heuristic 2 (Pump outputs HKZ reduced basis) *Given a d-dimensional lattice basis $\mathbf{B}$ with a reduction quality whose slope is equal to s. Under the reduction parameter $J \leq \mathrm{d4f}_{\mathrm{slope}}(s) \ll \beta \leq d$, for $\kappa < d-1$, the projected sublattice basis $\mathbf{B}_{\pi[\kappa,\min\{\kappa+\beta,d\}]}$ reduced by a Pump$(\mathbf{B}_{\pi[\kappa,\min\{\kappa+\beta,d\}]}, \kappa, \beta, f \leq \mathrm{d4f}_{\mathrm{slope}}(s))$, 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 $\|\mathbf{b}_i^*\| \approx \mathrm{GH}(\mathcal{L}(\mathbf{B}_{\pi[i,\min\{\kappa+\beta,d\}]}))$.*

Here $\mathrm{d4f}_{\mathrm{slope}}(s)$ is an upper bound of the d4f value estimation function based on the quality of the current lattice basis proposed in [43]. $\mathrm{d4f}_{\mathrm{slope}}(s) := \ln_\delta \sqrt{4/3} \approx \ln(4/3)/(-s)$.

As mentioned above, the default d4f function used in the implementation of PnJBKZ (both [20] and [28]) is an optimistic heuristic setting. This optimistic d4f setting in the implementation of G6K results in the actual reduction effect of a PnJBKZ(β, J) with the optimistic d4f setting being closer to that of a PnJBKZ(β', J) with the theory d4f estimation value, rather than a PnJBKZ(β, J) with the theoretical d4f estimation value. Here $\beta' \leq \beta$.

Therefore, to more accurately predict the behavior of the PnJBKZ using the optimistic d4f function, we propose the following simulation strategy. Specifically, using the information about the quality of the current lattice basis, such as the slope value s, [43] calculates the refined d4f value estimation as $\mathrm{d4f}_{\mathrm{slope}}(s) = \ln(4/3)/(-s)$. In this case $J \leq \mathrm{d4f}_{\mathrm{slope}}(s)$, we calculate $d4fgap(\beta, s) := d4f_{\mathrm{optimistic}}(\beta, s) - \mathrm{d4f}_{\mathrm{slope}}(s)$. If $\beta < 40$, $d4fgap(\beta, s) = 0$; $d4fgap(\beta, s) = \lfloor \beta - 40/2 - \mathrm{d4f}_{\mathrm{slope}}(s) \rfloor$, if $40 \leq \beta \leq 75$; $d4fgap(\beta, s) = \lfloor 11.5 + 0.075\beta - \mathrm{d4f}_{\mathrm{slope}}(s) \rfloor$, 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 used in the G6K implementation and the actual d4f value under current lattice reduction quality (slope s value). This simulation strategy, based

on the current lattice reduction quality information (slope value), provides a more refined d4f estimation to adjust the over-optimistic d4f used in the default implementation of PnJBKZ [20].

Finally, in Appendix D.1.3 and D.1.4, we show the verification experiments of Heuristic 2 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 found in Section 3.2 of [43]. The comparison results indicate that using the simulation strategy we mention above leads to more accurate predictions of the behavior of PnJBKZ which uses the optimistic d4f function. In this paper, we default to using the above simulation strategies to predict the practical reduction effect of the PnJBKZ(β, J) which uses the optimistic d4f setting in practice.

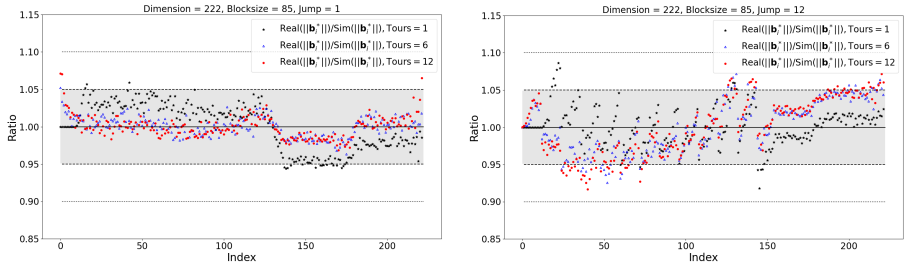
D.1.3 Verification Experiments of Heuristic 2 and the PnJBKZ Simulator for Predicting PnJBKZ Reduction with Optimistic d4f Settings

In this part, we show that Heuristic 2 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 were tested on the TU Darmstadt LWE Challenge lattice basis with parameter (n=60, α=0.010). Before running the PnJBKZ simulator, we performed a small block reduction to remove the influence of q-ary vectors in the LWE Challenge lattice basis. After this pre-processing, we obtained a 222-dimension lattice basis, which contains a few q-ary vectors at the beginning of the lattice basis. This lattice basis has a slope value equal to -0.0248 (with the pre-processing taking only a few minutes the walltime).

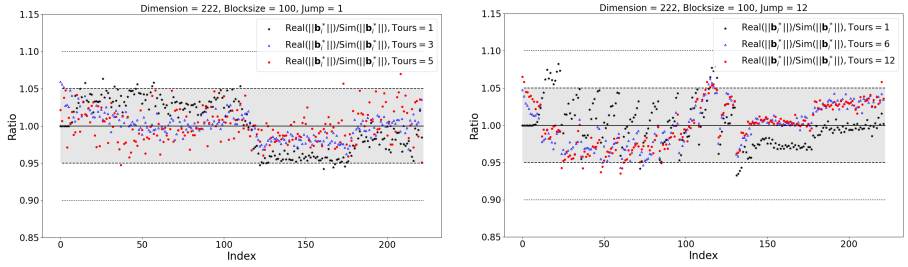
Next, we calculate the ratio $l_i''/\mathrm{Sim}(l_i'')$ for $i \in [0, d-1]$ in each tour of PnJBKZ's reduction, see Fig. 17. Here, l_i'' represents the average logarithms of these Gram-Schmidt vector lengths, which were obtained from 20 independent reduction experiments using the same reduction parameter $(\beta, J, tours)$. The PnJBKZ reduction was performed 20 times with these parameters respectively. Meanwhile, $\mathrm{Sim}(l_i'')$ denotes the simulated logarithms of the Gram-Schmidt vector lengths, calculated using Eq. (3).

We calculate $\mathrm{Sim}(l_i'')$ strictly based on the properties of the HKZ-reduced basis and the Gaussian Heuristic. Therefore, in addition to serving as a criterion for evaluating the accuracy of the PnJBKZ simulator, the ratio $l_i''/\mathrm{Sim}(l_i'')$ can also be used to assess whether Heuristic 2 holds. In particular, as shown in Fig. 17, for values of β ranging from 85 to 100 and jump from 1 to 12, we observe the minimum theoretical upper bound value for the current lattice basis quality: $\mathrm{d4f}_{\mathrm{slope}}(s = -0.0248) \approx 11.6 \leq 12$. When $\mathsf{jump} \leq \lceil \mathrm{d4f}_{\mathrm{slope}}(s = -0.0248) \rceil = 12$, even with the number of tours increasing to 12, all most of the ratios $l_i''/\mathrm{Sim}(l_i'')$ are within range: [0.95,1.05] (the rest ratios are also within range [0.90,1.10]).



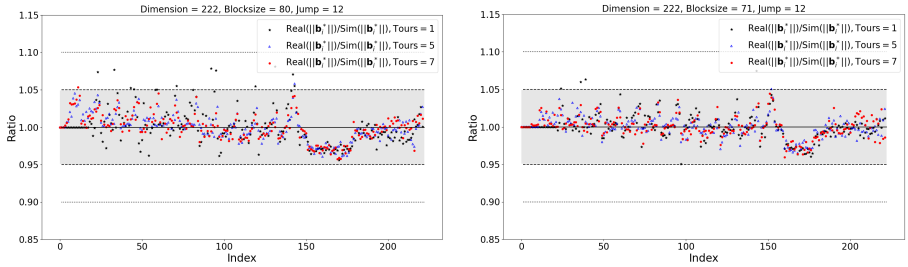
(a) β=85, jump=1

(b) β=85, jump=12



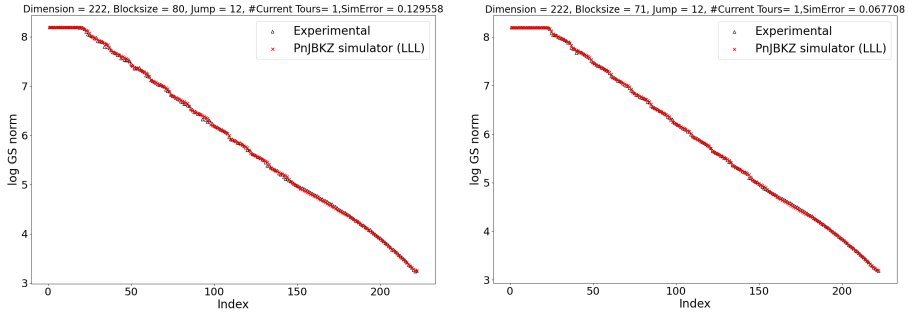
(c) β=100, jump=1

(d) β=100, jump=12



(e) β=80, jump=12

(f) β=71, jump=12



(g) β=80, jump=12

(h) β=71, jump=12

Fig. 17: Ratio $l_i''/\mathrm{Sim}(l_i'')$. Run PnJBKZ($\beta, J$) reduction on a 222-dimension LWE lattice basis (n=60,α=0.010) and record the ratio values. We test 20 times for each reduction parameter.

Fig. 17 demonstrates that, based on the refined d4f estimation, our PnJBKZ simulator accurately predicts the reduction effect of PnJBKZ. Furthermore, Heuristic 2 holds when $J \leq \mathrm{d4f}_{\mathrm{slope}}(s)$.

In comparison, when simulating classic BKZ, most ratios also fall within [0.95, 1.05], although some extreme ratios exceed this range. The largest ratio reaches 1.15, while the smallest falls below 0.85. This indicates that the prediction accuracy of the PnJBKZ simulator, when using d4f technology for PnJBKZ reductions, is at least as good as that of the classic BKZ simulator [17] for predicting BKZ reductions.

Furthermore, to improve the prediction accuracy of the PnJBKZ simulator when predicting PnJBKZ reduction with optimistic d4f settings, the PnJBKZ simulator, similar to the ideal version, will by default account for the impact of LLL reduction in its calculations. Specifically, in this paper, all experiments using the PnJBKZ simulator incorporate the Size-reduced and Lovász conditions of the LLL-reduced basis into the original version. As a result, in the subsequent experiments, we will no longer explicitly mention whether the influence of LLL reduction has been considered.

Moreover, this is further verified by (e) and (f) in Fig. 18 when the tours increase to 13. Meanwhile, the PnJBKZ simulator uses Eq. (3) as an approximate estimate for the actual value l_i''. This estimate effectively reflects how the average norms of Gram-Schmidt vectors change during each tour's reduction of PnJBKZ(β, J), which uses the optimistic d4f setting in practice.

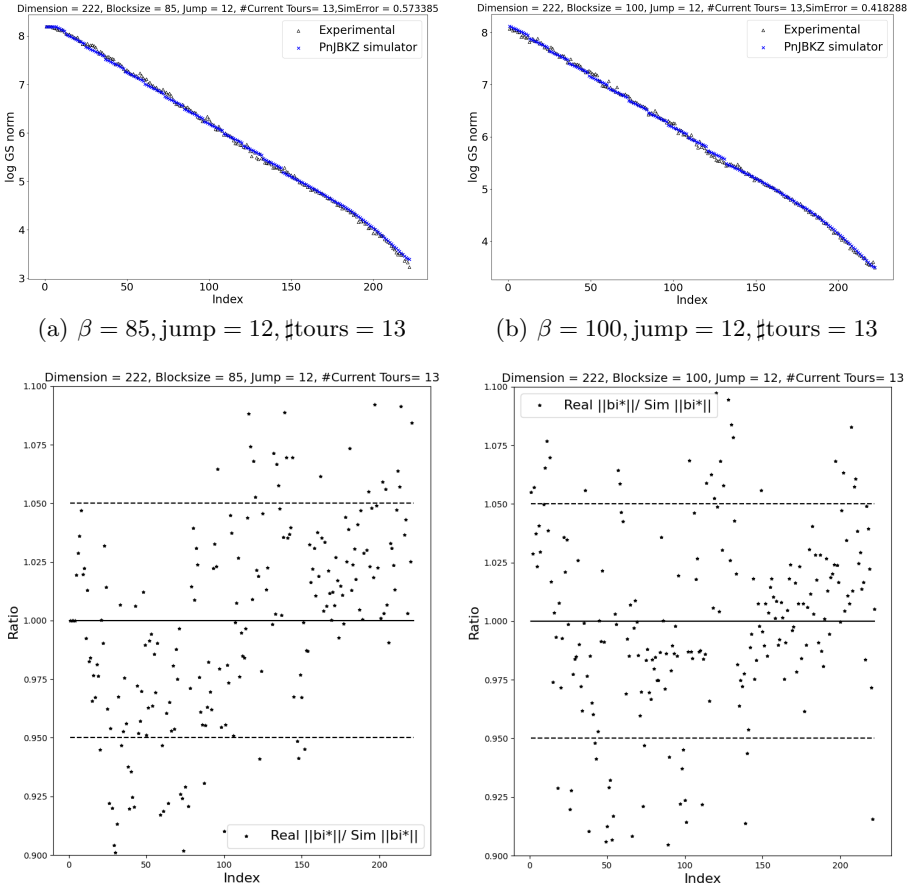
We define the prediction error as

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

where $\sharp$tours represents the number of current tours and $\mathrm{Sim}(\|\mathbf{b}_i^*\|)_{(\sharp\mathrm{tours})}$ are the lengths of Gram-Schmidt vectors predicted by PnJBKZ simulator with $\sharp$tours. Fig. 18 illustrates that the overall prediction error of the PnJBKZ simulator for different jumps is similar to that observed for jump=1.

More verification experiment results with different reduction parameters on various lattice bases can be found in Appendix D.1.4.

In addition, our experiments, detailed in Table 3 and Appendix D.1.5, demonstrate that for the LWE challenge lattice basis with various reduction parameters (e.g. different blocksizes and jump sizes), the predicted slope values from our PnJBKZ simulator closely align with those obtained from actual reductions. These results further demonstrate that, based on the 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.



(a) $\beta = 85, \text{jump} = 12, \sharp\text{tours} = 13$ (b) $\beta = 100, \text{jump} = 12, \sharp\text{tours} = 13$

(c) $\beta = 85, \text{jump} = 12, \sharp\text{tours} = 13$ (d) $\beta = 100, \text{jump} = 12, \sharp\text{tours} = 13$

Fig. 18: 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,α=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.

D.1.4 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 effect of the PnJBKZ algorithm, which uses optimistic d4f setting. These experiments were tested on various LWE challenge lattice bases with different reduction parameters. Specifically, we varied the blocksize 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 performed 20 independent experiments to calculate

the practical average length of the Gram-Schmidt vectors after applying the PnJBKZ reduction.

First, to eliminate the influence of q-ary vectors in the initial LWE challenge lattice basis, we perform pre-processing on all LWE challenge lattice basis using small blocksize reduction, which can be completed within a few minutes of wall time. For example, for $(n = 70, \alpha = 0.005)$ and $(n = 75, \alpha = 0.005)$, after pre-processing, we obtain LWE challenge lattice basis $(n = 70, \alpha = 0.005)$ having a slope of $-0.04921/2$. The corresponding $\mathrm{d4f}_{\mathrm{slope}}(s)$ values range from 11.7 to 13.7. Since we need the maximum jump value $J \leq \mathrm{d4f}_{\mathrm{slope}}(s)$ to ensure the accuracy of the PnJBKZ simulator (see Section 5 for details), we set the maximum jump value $J \leq 12$ in this test experiments.

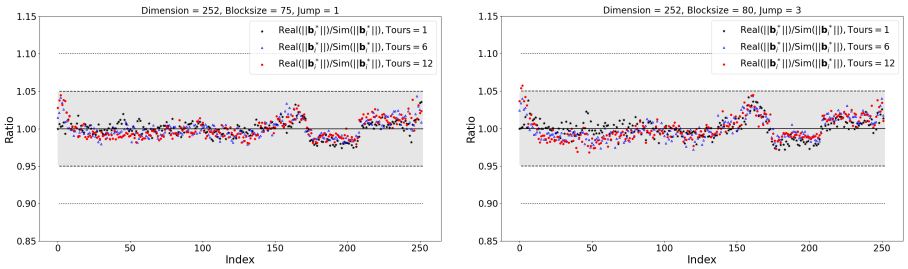
Then we present the results of verification experiments on four different lattice bases, with $\beta \in [50, 70]$ and jump values within $J \in [1, 12]$: $(n = 70, \alpha = 0.005)$, $(n = 75, \alpha = 0.005)$, $(n = 60, \alpha = 0.010)$ and $(n = 50, \alpha = 0.015)$. These results are shown in Figures 21 $\sim$ 29, respectively. The Verification experiment results indicate that our PnJBKZ simulator performs well in predicting the behavior of PnJBKZ for blocksize 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 19 $\sim$ 29 show that, for different lattice basis with different reduction parameters, as long as the jump $\leq \mathrm{d4f}_{\mathrm{slope}}(s)$, even when the number of tours increase to 13, overall, the simulation values are closed match the actual values. Most of the ratios $\frac{l_i''}{\mathrm{Sim}(l_i'')}$ remain within the range $[0.95, 1.05]$. Furthermore, while the progressive reduction strategy will not run the same reduction parameter (β, J) for more than 10 tours, our simulator remains accurate even when the number of tours increases to 13.

These results indicate that when $J \leq \mathrm{d4f}_{\mathrm{slope}}(s)$, we can ensure that Heuristic 2 holds, and the PnJBKZ simulator's estimation of the actual value $\|\mathbf{b}_i''^*\|$ (calculated 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. Similar to the classical BKZ simulator [17], the prediction error of our PnJBKZ simulator does not exceed 5%, with most ratios $\frac{l_i''}{\mathrm{Sim}(l_i'')}$ staying within the range $[0.95, 1.05]$.

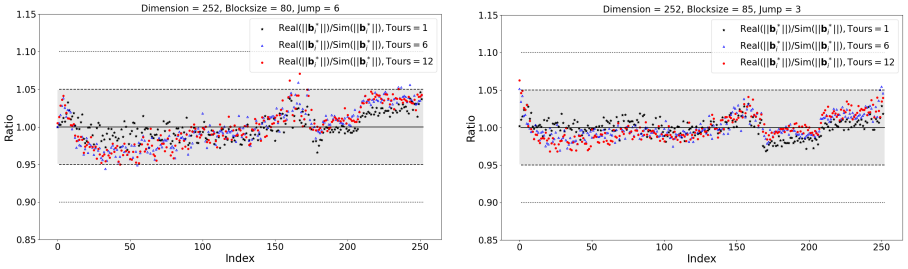
Additionally, Table 3 and Appendix D.1.5 show that the practical slope of lattice Gram-Schmidt basis after each tour of the PnJBKZ reduction, with different blocksizes and different jump values, closely matches our simulation results. The prediction error of lattice basis slope does not exceed 0.03%. For more details, see Table 3. This further validates the accuracy of our PnJBKZ simulator. For selecting the optimized reduction strategy, our PnJBKZ simulator provides sufficiently accurate predictions.

D.1.4.1 LWE challenge lattice basis ($n = 75, \alpha = 0.005$).
Figure 19 $\sim$ Figure 20.



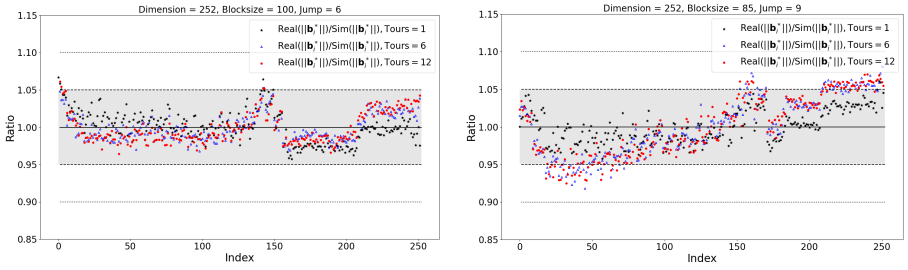
(a) β=75, jump=1

(b) β=80, jump=3



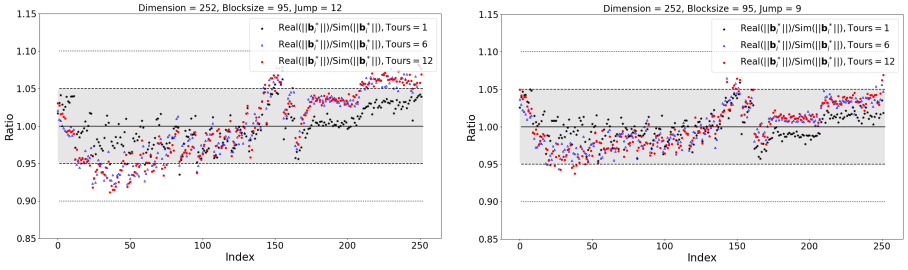
(c) β=80, jump=6

(d) β=85, jump=3



(e) β=100, jump=6

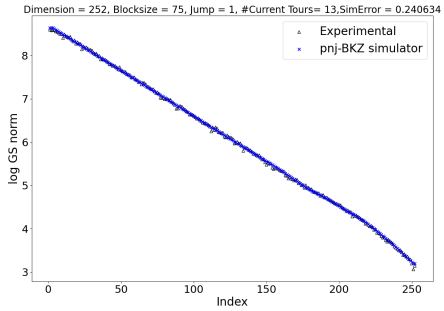
(f) β=85, jump=9



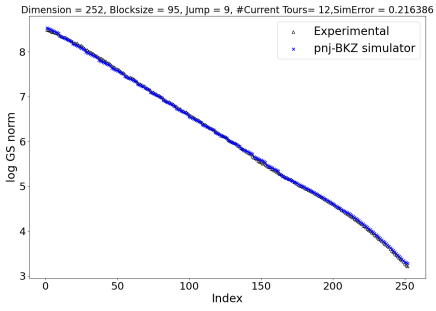
(g) β=95, jump=12

(h) β=95, jump=9

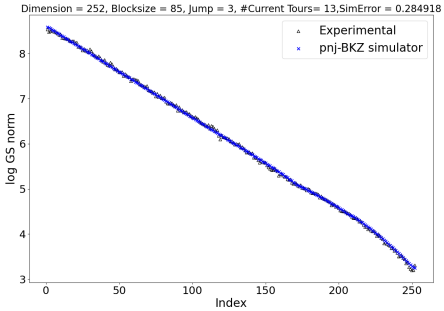
Fig. 19: Ratio $l_i''/\mathrm{Sim}(l_i'')$. Run 12 tours of PnJBKZ(β, J) reduction on a 252-dimension LWE lattice basis ($n = 75, \alpha = 0.005$), and record the ratio values. We test 20 times for each reduction parameters.



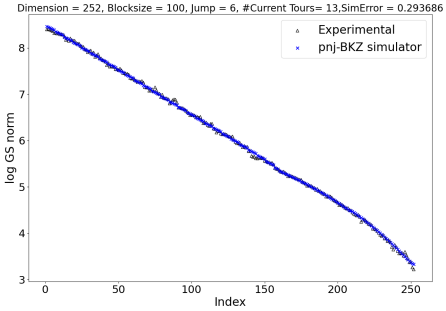
(a) $\beta = 75, \text{jump} = 1, \sharp\text{tours} = 13$



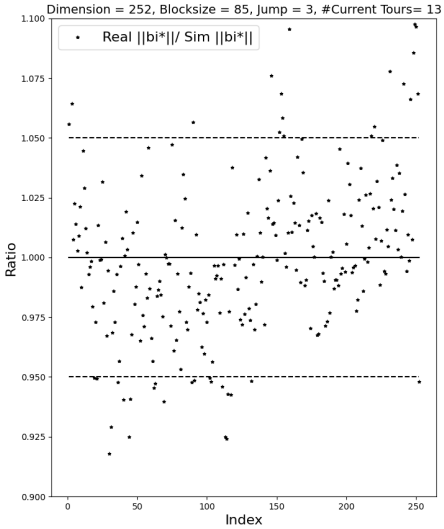
(b) $\beta = 95, \text{jump} = 9, \sharp\text{tours} = 12$



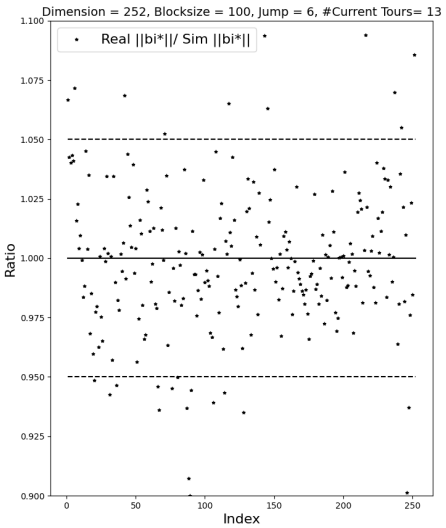
(c) $\beta = 85, \text{jump} = 3, \sharp\text{tours} = 13$



(d) $\beta = 100, \text{jump} = 6, \sharp\text{tours} = 13$



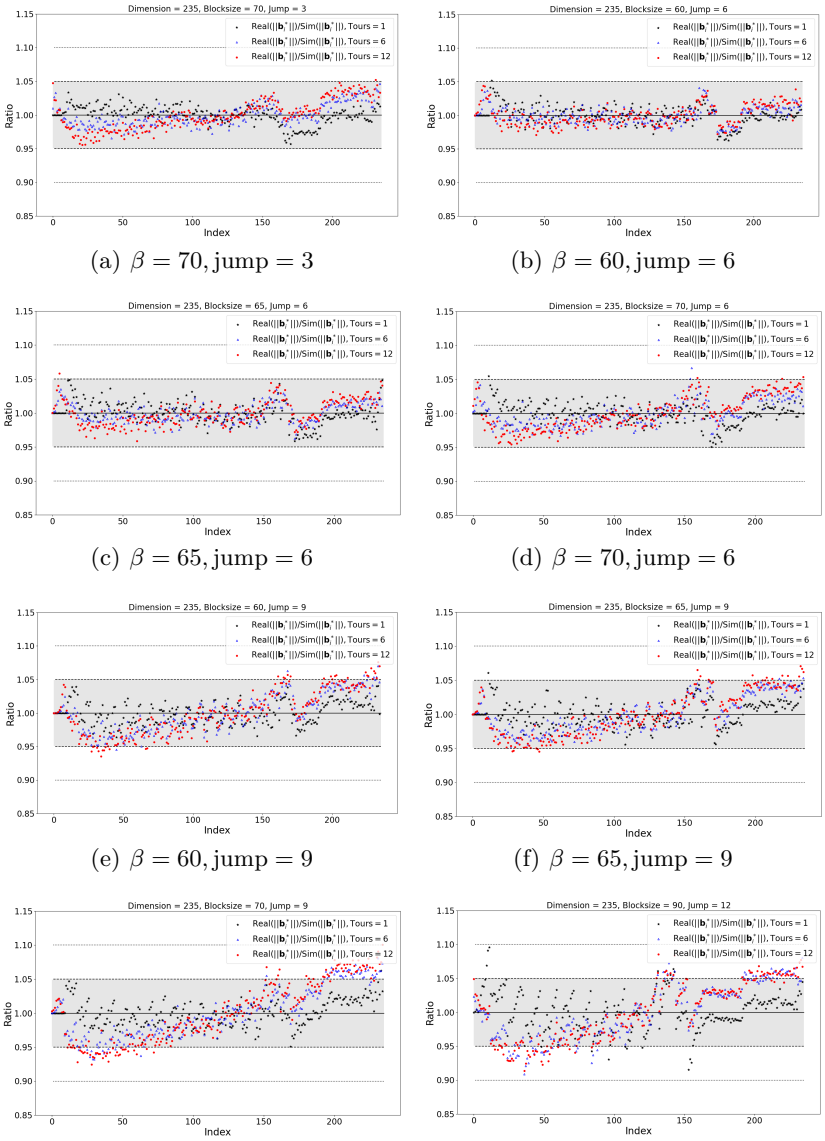
(e) $\beta = 85, \text{jump} = 3, \sharp\text{tours} = 13$



(f) $\beta = 100, \text{jump} = 6, \sharp\text{tours} = 13$

Fig. 20: 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.

D.1.4.2 LWE challenge lattice basis ($n = 70, \alpha = 0.005$).
Figure 21 $\sim$ Figure 23.



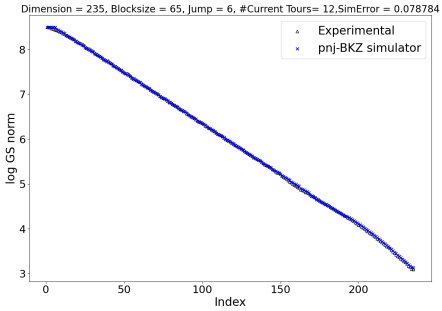
(a) $\beta = 70, \text{jump} = 3$ (b) $\beta = 60, \text{jump} = 6$

(c) $\beta = 65, \text{jump} = 6$ (d) $\beta = 70, \text{jump} = 6$

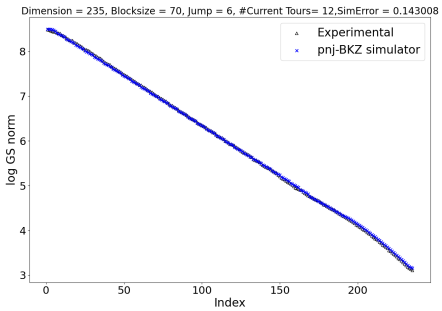
(e) $\beta = 60, \text{jump} = 9$ (f) $\beta = 65, \text{jump} = 9$

(g) $\beta = 70, \text{jump} = 9$ (h) $\beta = 90, \text{jump} = 12$

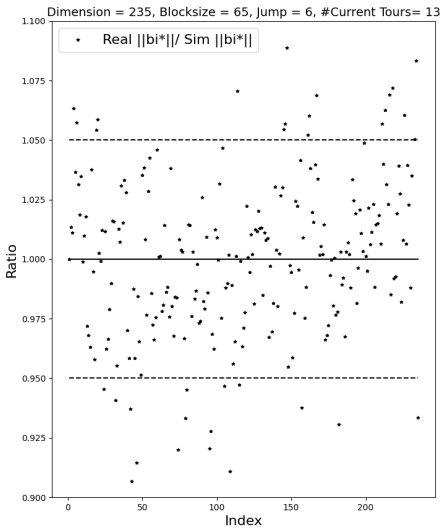
Fig. 21: Ratio $l_i''/\text{Sim}(l_i'')$. Run 12 tours of PnJBKZ(β, J) reduction on a 235-dimension LWE lattice basis ($n = 70, \alpha = 0.005$), and record the ratio values. We test 20 times for each reduction parameter.



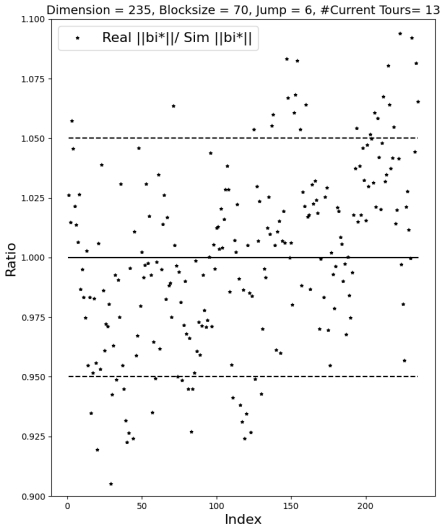
(a) $\beta = 65, \text{jump} = 6, \sharp\text{tours} = 12$



(b) $\beta = 70, \text{jump} = 6, \sharp\text{tours} = 12$

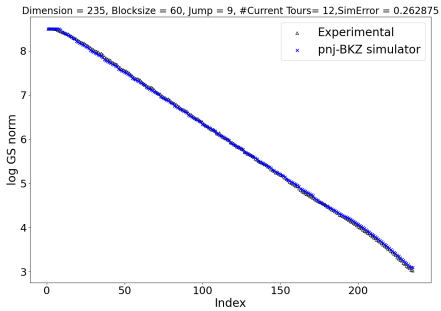


(c) $\beta = 65, \text{jump} = 6$

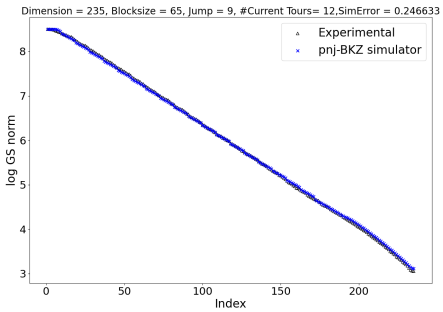


(d) $\beta = 70, \text{jump} = 6$

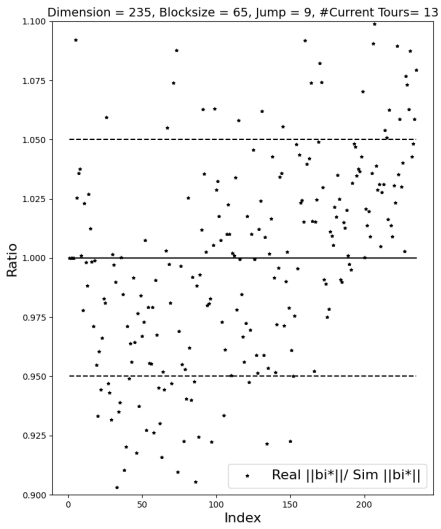
Fig. 22: Ratio $l_i''/\text{Sim}(l_i'')$. Run 13 tours of PnJBKZ(β, J) reduction on a 235-dimension LWE lattice basis ($n = 70, \alpha = 0.005$), different β with $J = 6$, and record the ratio values. We test 20 times for each reduction parameter.



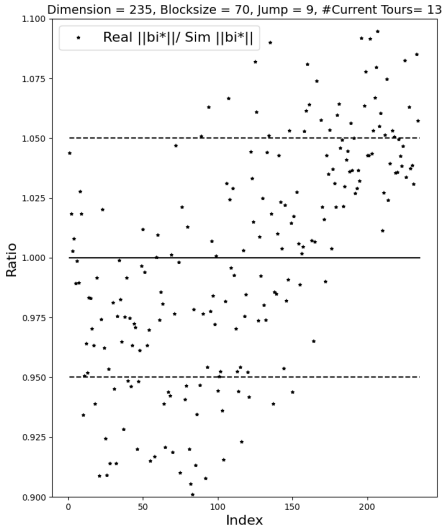
(a) $\beta = 60, \text{jump} = 9, \sharp\text{tours} = 12$



(b) $\beta = 65, \text{jump} = 9, \sharp\text{tours} = 12$



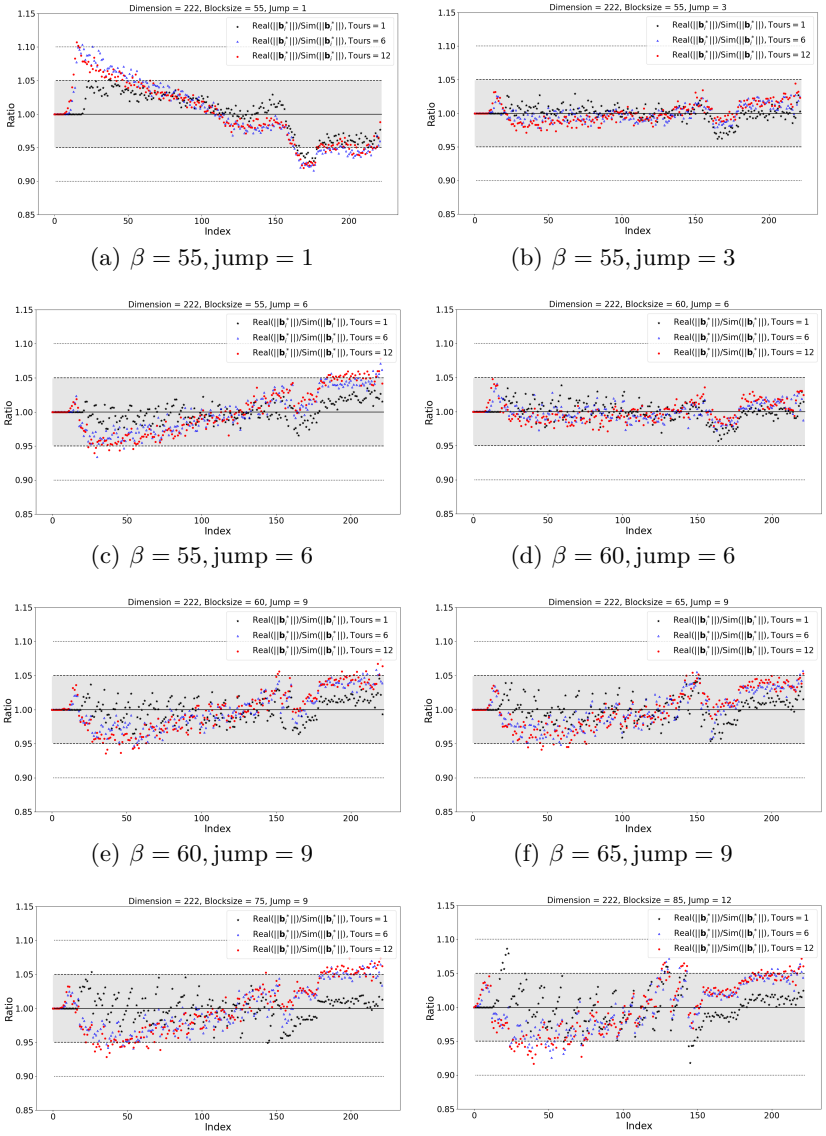
(c) $\beta = 65, \text{jump} = 9$



(d) $\beta = 70, \text{jump} = 9$

Fig. 23: 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.

D.1.4.3 LWE challenge lattice basis ($n = 60, \alpha = 0.010$).
Figure 24 $\sim$ Figure 26.



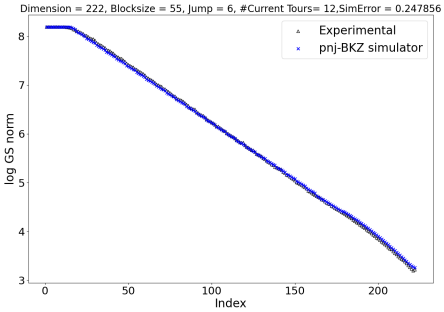
(a) $\beta = 55, \text{jump} = 1$ (b) $\beta = 55, \text{jump} = 3$

(c) $\beta = 55, \text{jump} = 6$ (d) $\beta = 60, \text{jump} = 6$

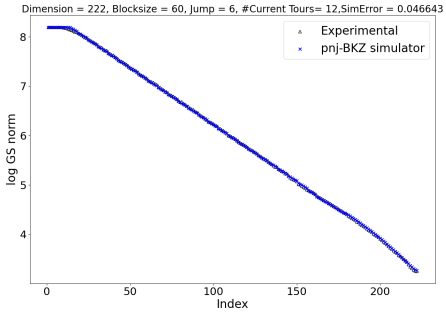
(e) $\beta = 60, \text{jump} = 9$ (f) $\beta = 65, \text{jump} = 9$

(g) $\beta = 75, \text{jump} = 9$ (h) $\beta = 85, \text{jump} = 12$

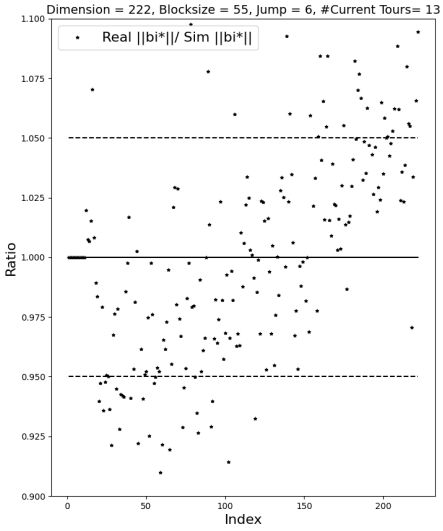
Fig. 24: Ratio $l_i''/\text{Sim}(l_i'')$. Run 12 tours of PnJBKZ(β, J) reduction on a 222-dimension LWE lattice basis ($n = 60, \alpha = 0.010$), and record the ratio values. We test 20 times for each reduction parameter.



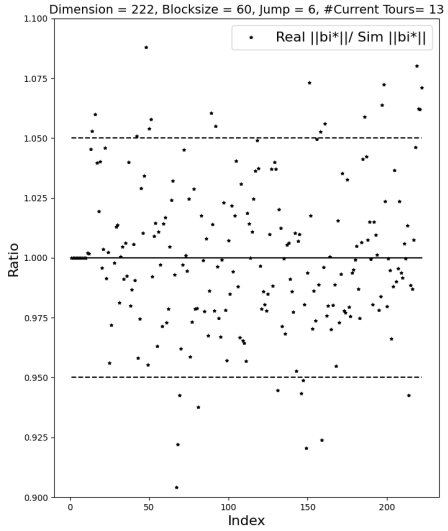
(a) $\beta = 55, \text{jump} = 6, \sharp\text{tours} = 12$



(b) $\beta = 60, \text{jump} = 6, \sharp\text{tours} = 12$

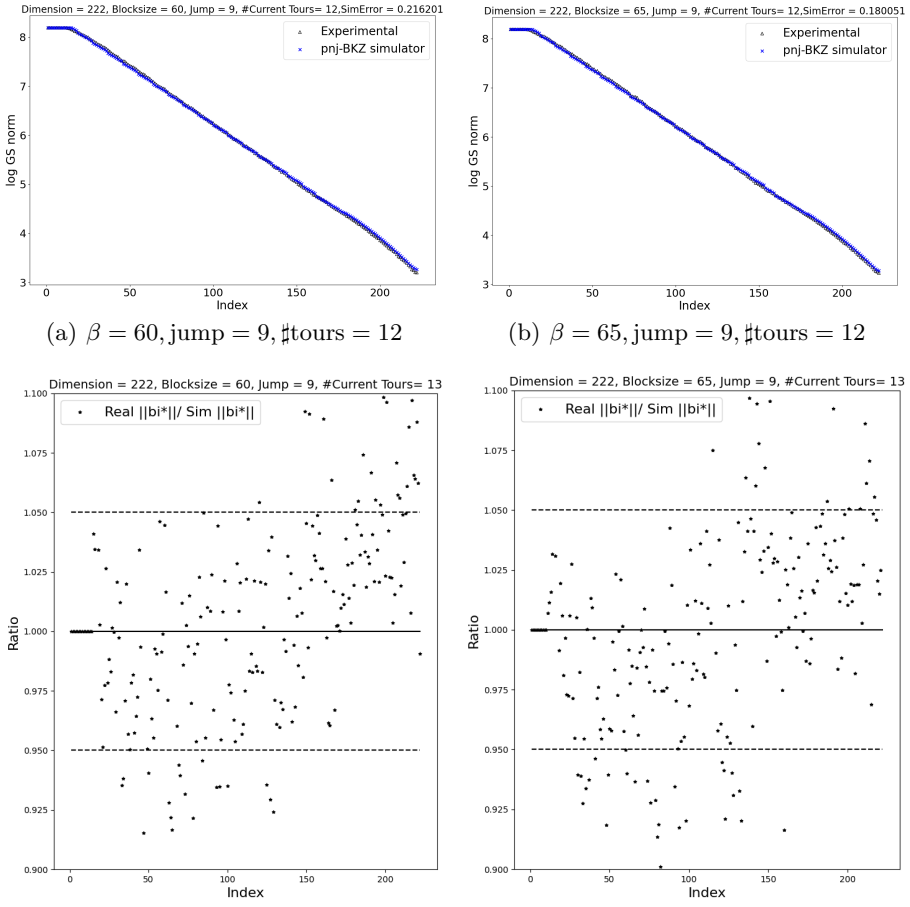


(c) $\beta = 55, \text{jump} = 6$



(d) $\beta = 60, \text{jump} = 6$

Fig. 25: 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) $\beta = 60, \text{jump} = 9, \sharp\text{tours} = 12$

(b) $\beta = 65, \text{jump} = 9, \sharp\text{tours} = 12$

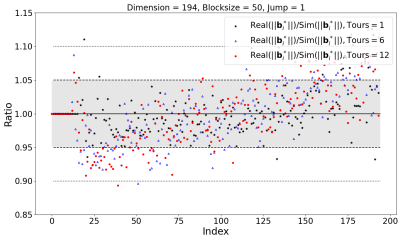
(c) $\beta = 60, \text{jump} = 9$

(d) $\beta = 65, \text{jump} = 9$

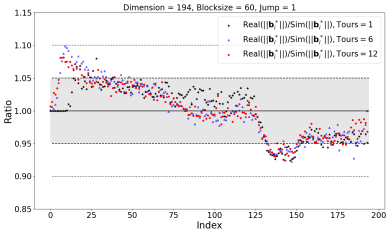
Fig. 26: Ratio $l_i''/\text{Sim}(l_i'')$. Run 13 tours of PnJBKZ(β, J) reduction on a 222-dimension LWE lattice basis ($n = 60, \alpha = 0.010$), different β with $J = 9$, and record the ratio values. We test 20 times for each reduction parameter.

D.1.4.4 LWE challenge lattice basis ($n = 50, \alpha = 0.015$).

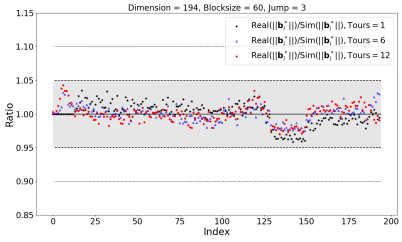
Figure 27 $\sim$ Figure 29.



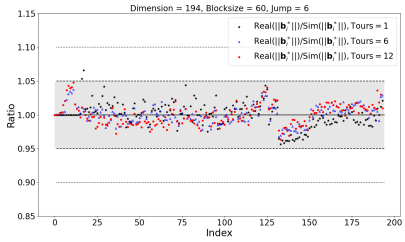
(a) $\beta = 50, \text{jump} = 1$



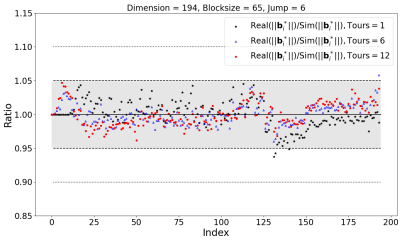
(b) $\beta = 60, \text{jump} = 1$



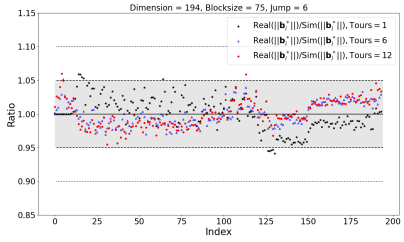
(c) $\beta = 60, \text{jump} = 3$



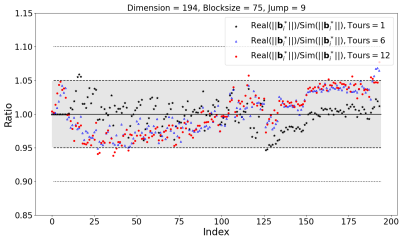
(d) $\beta = 60, \text{jump} = 6$



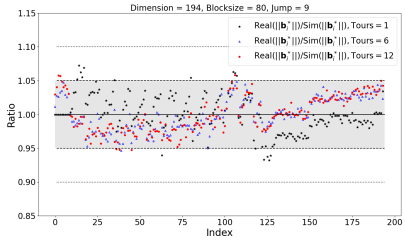
(e) $\beta = 65, \text{jump} = 6$



(f) $\beta = 75, \text{jump} = 6$

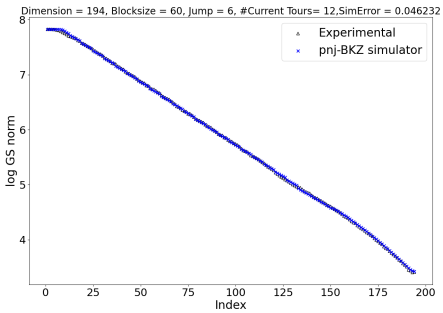


(g) $\beta = 75, \text{jump} = 9$

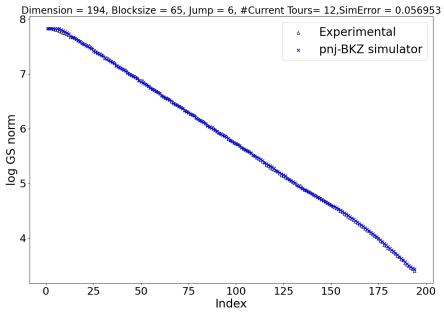


(h) $\beta = 80, \text{jump} = 9$

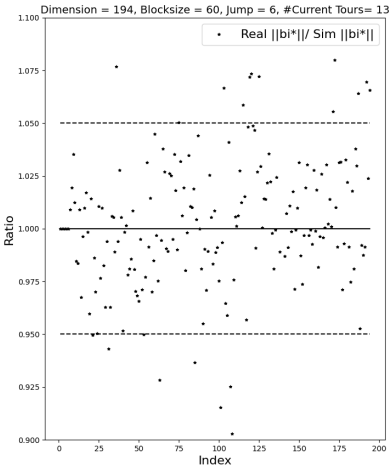
Fig. 27: Ratio $l_i''/\text{Sim}(l_i'')$. Run 12 tours of PnJBKZ(β, J) reduction on a 194-dimension LWE lattice basis $(n = 50, \alpha = 0.015)$, and record the ratio values. We test 20 times for each reduction parameter.



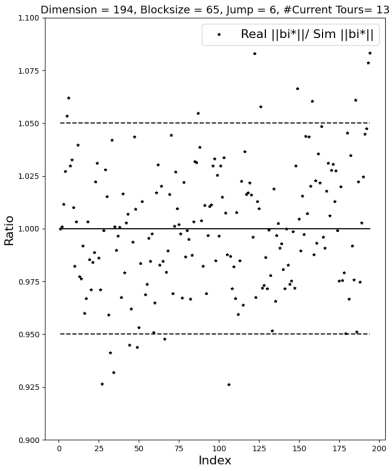
(a) $\beta = 60, \text{jump} = 6, \sharp\text{tours} = 12$



(b) $\beta = 65, \text{jump} = 6, \sharp\text{tours} = 12$

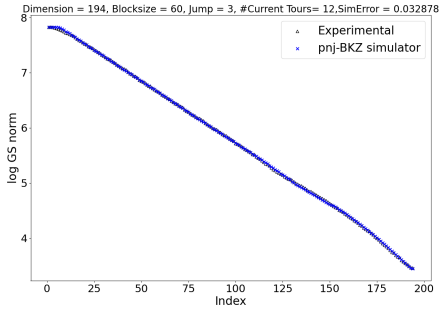


(c) $\beta = 60, \text{jump} = 6$

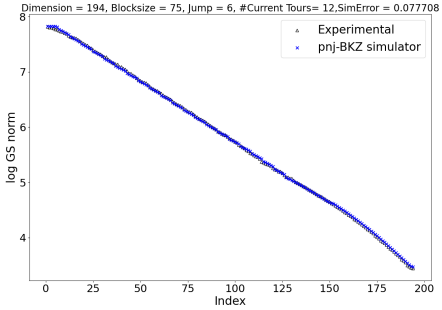


(d) $\beta = 65, \text{jump} = 6$

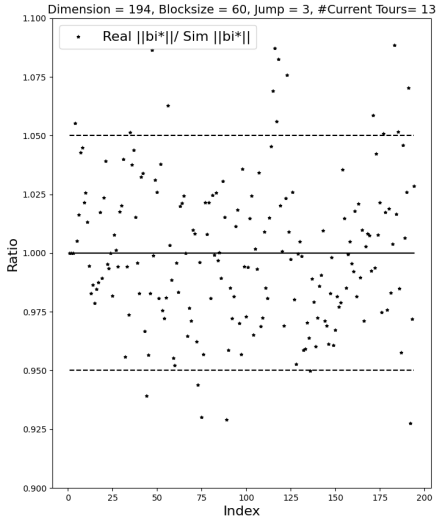
Fig. 28: Ratio $l_i''/\text{Sim}(l_i'')$. Run 13 tours of PnJBKZ(β, J) reduction on a 194-dimension LWE lattice basis ($n = 50, \alpha = 0.015$), different β with $J = 6$, and record the ratio values. We test 20 times for each reduction parameter.



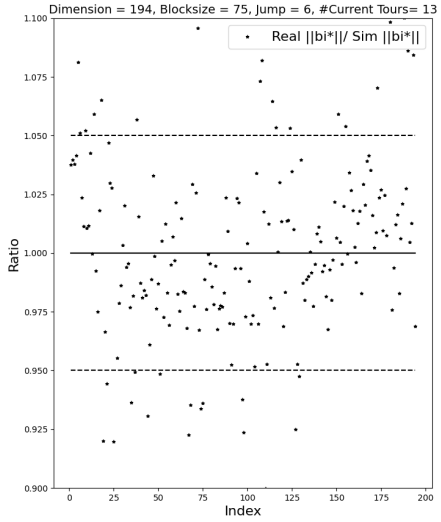
(a) $\beta = 60, \text{jump} = 3, \sharp\text{tours} = 12$



(b) $\beta = 75, \text{jump} = 6, \sharp\text{tours} = 12$



(c) $\beta = 60, \text{jump} = 3$



(d) $\beta = 75, \text{jump} = 6$

Fig. 29: 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.

D.1.5 Comparison between simulated slope (cost) and real slope (cost) during reduction In this part, we will demonstrate the accuracy of our PnJBKZ simulator by showing that the simulation slope of lattice basis closely matches the real slope of lattice basis reduced by the PnJBKZ algorithm.

We present a comparison of the slope and time cost for two LWE Challenges using qd float type in Tables 7, 8 and Table 9. These tables show that the simulated slope and cost closely match the real slope and time cost. They also demonstrate that our PnJBKZ simulator accurately reflects how the average of the norms of Gram-Schmidt vectors change during the reduction of PnJBKZ(β, J) on different LWE lattice bases.

Tables 4, 7, 8 and Table 9 all show that, although there is a slight gap between the slope of the simulated Gram-Schmidt (GS) norms and the slope of the real reduced GS norms in the first round of reduction due to the influence of the q-ary vector in the initial LWE lattice basis, this gap decreases as the reduction progresses. Before finally entering the Pump stage, the difference between the simulated and real reduction slopes becomes sufficiently small. This demonstrates that for selecting the optimized blocksize and jump strategy, our PnJBKZ simulator is accurate enough.

(β, J)	Simulation		Practical	
	Slope	log(T)	Slope	log(T)
(70,8)	-0.0288	6.4	-0.0278	6.6
(80,10)	-0.0256	6.4	-0.0249	6.6
(102,11)	-0.0221	7.7	-0.0218	8.0
(102,11)	-0.0207	7.7	-0.0208	8.0
(103,11)	-0.0202	7.8	-0.0205	8.1

Table 7: Quality and wall time (T in seconds) during reduction of LWE Challenge $(n, \alpha) = (45, 0.020)$.

(β, J)	Simulation		Practical	
	Slope	log(T)	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 8: Quality and log(walltime) (log(T) in seconds) during reduction of LWE Challenge $(n, \alpha) = (50, 0.015)$.

Table 9: Quality and log(walltime) (log(T) in seconds) during reduction of LWE Challenge $(n, \alpha) = (40, 0.025)$.

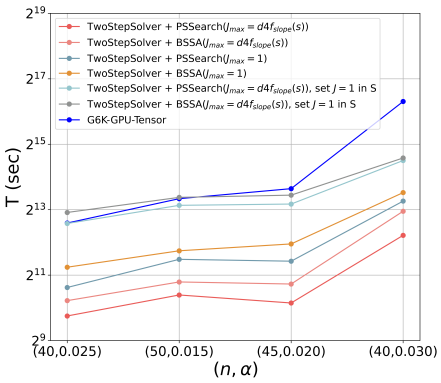
(β, J)	Simulation		Practical	
	Slope	log(T)	Slope	log(T)
(77,8)	-0.0281	6.5	-0.0265	6.6
(81,10)	-0.0249	6.2	-0.0241	6.6
(102,11)	-0.0217	7.5	-0.0215	7.8
(102,11)	-0.0205	7.5	-0.0207	7.8

D.2 Experiments and Application to LWE

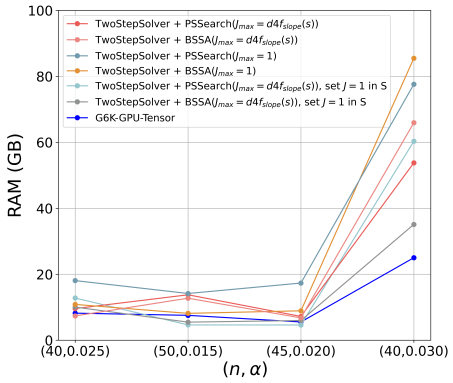
In this section, we evaluate the walltime of different LWE solving strategies through experiments to demonstrate that our approach is more efficient in solving LWE compared to the heuristic LWE solving strategy used in G6K. Specifically, we employ an optimized reduction strategy for solving LWE, which is derived from either PSSearch (Alg. 3) or BSSA.

D.2.1 Efficiency of MinTwoStepSolver for solving LWE The default LWE solving algorithm in G6K is the script `lwe_challenge.py` in the imple-

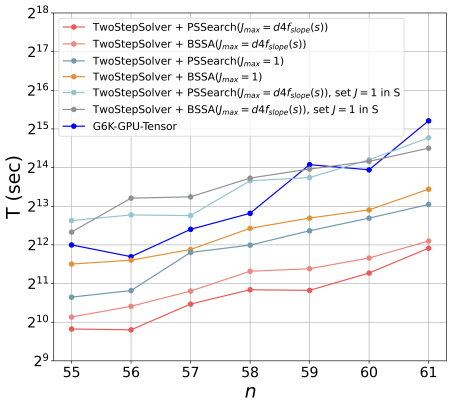
mentation of G6K-GPU-Tensor [28], which uses a heuristic reduction strategy. For more detail about the default LWE solving algorithm in G6K-GPU-Tensor, refer to the script at [4]. Fig. 30 presents the experimental result of different LWE-solving algorithms. We use the practical cost model proposed in Appendix B.2 as $\mathcal{T}$.



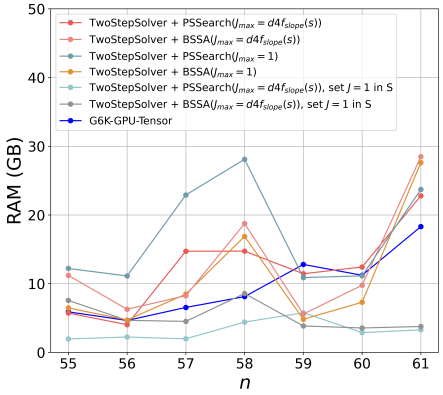
(a) LWE Challenges: Time.



(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. 30: Comparison of Different LWE-solving Algorithms under same benchmark. §

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

The blue lines and scatter points in Fig. 30 represent the experimental time or memory cost of the default strategy in G6K. The remaining lines and scatter points in Fig. 30 indicate the experimental time or memory cost of MinTwoStepSolver, using the strategies generated by PSSearch(Alg. 3) (BSSA(Alg. 6)). We modify the progressive blocksize selection strategy from ProBKZ [19] to adapt it to 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 B.3 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.

From the result in Fig. 30, we can observe that using the strategy selected by PSSearch (BSSA) significantly decreased the walltime cost by approximately 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. The log files of Fig. 30 for these results can be found in the `lwechal-test` and `lwe-instance-test`. The results can also be reproduced by running the test code `implement_lwechal_forall.sh` and `implement_lwe_instance_forall.sh` in source code[2].

As shown 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. 30(a) and Fig. 30(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. 30(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 $(n, \alpha) \in \{(n, 0.010) \mid n = 55, \cdots, 61\}$. This indicates that the progressive reduction strategy generated by ProBKZ [19] is not optimal in terms of time cost, especially for large n, with an acceptable memory cost, as demonstrated in Fig. 30(d).

Additionally, experiments shown in Fig. 30 indicate that the flexible use of a solving strategy with $\mathsf{jump}>1$ can solve LWE 4.88 to 7.85 times faster than the $J_{\max}=1$ solving strategy. Specifically, in Fig. 30(c), we define S_1 as the strategy generated by PSSearch with $J_{\max}=\mathrm{d4f}_{\mathrm{slope}}(s)$, 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 $\mathrm{d4f}_{\mathrm{slope}}(s)$ represents the simulated slope of the lattice basis during the reduction process. This demonstrates a substantial improvement in reduction efficiency when the maximum jump exceeds 1, potentially by skipping some sufficiently reduced lattice bases, since each Pump turns on sieving at the Pump-down stage, and there is significant overlap between each Pump.

Furthermore, one can also limit the maximum memory usage of solving LWE by adjusting parameter "`--max_RAM`" to control the generated solving strategy. Additionally, we also designed an LWE sample optimized selection algorithm (Alg. 7) to optimize the number of chosen LWE samples in Appendix B.4. However, the efficiency improvement from this optimization is not significant, with a maximum gain of 2.2% in our test.

D.2.2 New LWE Records The TU Darmstadt LWE Challenge website presents Challenges designed to test the efficiency of solving LWE, helping to estimate the hardness of LWE in practice.

Using our new algorithm, i.e. MinTwoStepSolver, we have successfully 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)$\}$ from the TU Darmstadt LWE Challenge website[1]. For more details, refer to Fig. 2 for more details. Specifically, we used the following machines for our experiments: Machine A, with AMD EPYC$^{\text{TM}}$ 7002 Series 128@2.6GHz, NVIDIA 3090 * 8, 1.5T RAM; Machine B, with AMD EPYC$^{\text{TM}}$ 7002 Series 64@2.6GHz, a100 * 4, 512 GB RAM; and Machine C, a workstation with Intel Xeon 5128 16c 32@2.3GHz, 1.48T RAM and NVIDIA RTX 3090 * 2 GPUs.

Then we present the walltime and RAM cost for solving the above LWE Challenges in Table 4. The units for time (T) in Tables 3 and 4 are seconds and hours, respectively. In Table 4, we observe that the time required to solve the LWE Challenge with parameters $(n, \alpha) = (40, 0.0035)$ using the G6K-GPU [28] is 23.4 times longer than the time of our method.

D.3 The Optimized Strategy for the LWE Challenge

In Table 10, we present the optimized blocksize and jump strategy generated by PSSearch for successfully solving TU Darmstadt LWE Challenge with the parameters

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

. These results were obtained by running by running "implement_unsolved_lwechal.sh" in source code [2].

E 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 lies in replacing the success condition for solving u-SVP from Section B.1 with the success condition for solving SVP proposed in [26]. This section focuses on demonstrating the practical efficiency of the MinTwoStepSolver in solving SVP instances. The The algorithm

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

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

(n, α)	RAM limit	Strategy (β, jump)	PSSearchGen/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

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, a core component of PSSearch. The executable code of TwoStepSolver for solving SVP based on PSSearch is publicly available on GitHub[3].

[3] https://github.com/Summwer/lwe-estimator-with-pnjbkz