Optimizing Two-Step Primal LWE Attacks with PnJBKZ Simulation

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

[1] Shanghai Jiao Tong University, Shanghai, 200240, China
{xww_summer, wanggxx, dwgu}@sjtu.edu.cn
[2] State Key Laboratory of Integrated Service Networks, Xidian University, Xi'an, 710071, China
lzwang_2@stu.xidian.edu.cn, bcwang@xidian.edu.cn,
[3] School of Cyber Engineering, Xidian University, Xi'an, 710071, China
[4] Univ Bordeaux, CNRS, Inria, Bordeaux INP, IMB, UMR 5251, Talence, France

Abstract. Learning with Errors (LWE) and its variants underlie many lattice-based cryptosystems, making accurate concrete attack-cost estimates important for parameter selection. Practical primal attacks combine lattice-basis reduction with a final shortest-vector search, and their performance depends strongly on how these components are scheduled. We formalize the two-step strategy space considered in this work and propose PSSearch, a pruning-based method that selects a minimum estimated cost strategy under a specified reduction simulator and cost model. We instantiate the framework with a Pump-and-Jump BKZ (PnJBKZ) simulator supporting jump parameters $J > 1$, practical dimension-for-free settings, and LLL effects, together with a refined estimator for the dimension of the final SVP call. We integrate these components into a practical solver in G6K-GPU-Tensor and compare the PSSearch-selected strategies with its default heuristic strategy under comparable settings. On six TU Darmstadt LWE Challenge instances, the selected strategies achieve speedups of 7.2×–23.4×. Under the stated primal-attack and classical gate-count assumptions, the optimized two-step primal estimates for selected ML-KEM and ML-DSA parameter sets are 3.4–4.6 bits below the corresponding previous estimates; relative to the prescribed two-step baseline S_0, PSSearch provides an additional 1.2–1.3-bit reduction. Along the tested relation $\log_2 q = 0.198n^{0.832}$, the empirically fitted advantage satisfies $\Delta(\log_2 T) = 0.01887n - 0.98$, where T denotes the estimated attack cost. These figures concern the primal-attack branch; the overall attack-cost estimate is determined by the minimum cost among all applicable attack families.

Keywords: Learning with Errors · lattice cryptanalysis · primal attack · lattice reduction · PnJBKZ · lattice sieving · strategy optimization · concrete security estimation

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

1 Introduction

Learning with Errors (LWE) [Reg09] plays a crucial role in lattice-based cryptography, as the security of many lattice-based cryptographic schemes [DLP14, Boy13, MKMS22, CKKS17] depends on the hardness of LWE or its variants [Reg09, LPR10, BG14, ABD+21], including the NIST post-quantum standards ML-KEM and ML-DSA [ADM+24, DLL+24]. 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.

The main attack families against LWE are primal, dual, and hybrid attacks [KKNM25]. Under the native success-probability conventions of the two analyses, the dual attack of Carrier et al. [CMHST25], which combines lattice reduction with polar-code-based lossy source coding and FFT processing, reports lower costs for Kyber than the optimized primal-attack estimates obtained in this work under the common classical circuit model.

In contrast, primal attacks admit mature implementations based on G6K, PnJBKZ, and lattice sieving, and therefore provide a natural setting for studying practical reduction-schedule optimization. Accordingly, this article focuses on optimizing the primal branch. All security figures reported below should be interpreted as primal-attack cost estimates rather than overall security levels of the corresponding schemes. The overall security of a scheme is bounded by the minimum cost among all applicable primal, dual, and hybrid attacks. Fundamentally, all these attack strategies rely on the most essential lattice cryptanalysis tools: lattice reduction algorithms (such as the BKZ family [CN11, TKH18]) and their core subroutines, namely SVP solvers (divided into enumeration [GNR10, ABF+20] and sieving algorithms [MV10, FBB+15, HK17, HKL18, BGJ15, BDGL16, ADH+19, DSvW21, ZD26]).

Indeed, the choice of lattice reduction and SVP solving algorithms is crucial across all LWE attack strategies: the primal, dual, and hybrid attacks. The total computational cost for solving LWE is heavily dependent on the specific algorithmic configurations employed in practice. This involves critical design choices, such as applying lattice reduction to the full lattice basis, executing SVP calls on specific projected sublattices, and determining how these components are optimally integrated. We refer to this overarching configuration as the *LWE-solving strategy*.

The concrete security of lattice-based cryptography, particularly NIST post-quantum standards ML-KEM [ADM+24] and ML-DSA [DLL+24], is tightly bound to the precise computational cost of LWE attacks. The Kyber Round-3 specification [ABD+21], which preceded the standardization of ML-KEM, identifies refined BKZ strategies as Open Problem Q7. It considers a single higher-dimensional SVP call after BKZ preprocessing and notes that a precise analysis of the gains in large dimensions is still missing. The designers anticipate that resolving this could potentially decrease the security estimation for Kyber by 2 to 8 bits. Based on our current knowledge, the current LWE-solving strategy is heuristic. However, contemporary security estimates still rely on sub-optimal,

heuristic LWE-solving strategies. To systematically uncover these potential 2 to 8 bits of security loss, we must bridge the gap between heuristic practices and time-optimal configurations. In what follows, we first formalize the limitations of the existing heuristic baseline, and then systematically deconstruct Kyber's open question into four fundamental research questions **Q1**∼**Q4**.

1.1 Prior Works and Research Questions

LWE heuristic solving Strategy. In most prior work on solving LWE, lattice reduction and SVP algorithms have been invoked in heuristic strategies, for example, a fixed blocksize β_{ADPS16} is called iteratively until a short vector is found by BKZ-β_{ADPS16} in BKZ 2.0 [CN11] and ADPS16 [ADPS16], with no additional SVP call after the BKZ procedure. The specific block size β_{ADPS16} is chosen as the minimal value required to ensure that the projected error vector is shorter than the corresponding Gram-Schmidt vector of the reduced basis, thereby successfully recovering the LWE secret.

To provide an alternative to the fixed-blocksize approach, other frameworks employ a progressive strategy. Specifically, the "lattice-estimator" [AYH] utilizes a Progressive BKZ approach where the blocksize is incrementally increased by 1, concluding with a final sieving call. This framework determines the final sieving dimension β_{search} by balancing the cumulative reduction time cost $(\sum_{i=2}^{\beta_{\text{end}}} 2^{c\cdot i})$ with the final SVP time cost $(2^{c\cdot\beta_{\text{search}}})$, where c is a constant that depends on the specific SVP algorithm being used. The resulting execution sequence progresses as follows:

$$\text{BKZ-2},\ \text{BKZ-3},\ \ldots,\ \text{BKZ-}\beta_{\text{end}},\ \text{SVP-}\beta_{\text{search}}.$$

This naturally raises several critical questions: Does there exist an LWE-solving strategy that strictly minimizes the computational cost of lattice attacks? How can we find such a strategy? Furthermore, how would such an optimized strategy affect the concrete security estimates of lattice-based schemes?

In Section 3.1, we review existing LWE-solving strategies and analyze how their specific configurations, particularly the choice of block sizes during lattice reduction, impact the overall computational cost. This comparative analysis naturally leads us to formulate our central research question:

Question 1 (Q1) *Given an LWE instance* $(\mathbf{A}, \mathbf{b})$*, how can one select an admissible lattice-reduction strategy that minimizes the estimated attack cost under a specified reduction simulator,final-SVP dimension estimator and cost model?*

While we use the primal attack as the main instantiation because of its practical efficiency and mature implementations, the reduction-strategy search component is not inherently primal-specific. It can potentially be adapted to dual and hybrid attacks by replacing the terminal algorithm and cost model with attack-specific counterparts. The complete instantiation and evaluation in this article, however, concern the primal uSVP attack. For this setting, we show that, under the Geometric Series Assumption (GSA) and the specified cost model, the two-step mode has strictly lower estimated total cost than the corresponding

BKZ-only mode. A preliminary version of this analysis appeared in our PKC 2024 paper [XWW+24, Theorem 1].

Two-step mode. The two-step mode denotes a standard technique for solving lattice problems in practice [LN13, AWHT16, XWW+24, CMHST25]. It has two sequential phases: first, a reduction step, in which lattice reduction is performed on the basis to progressively improve its quality; followed by a search step, which calls a search algorithm (e.g., an SVP solver) to find the target vector on this newly reduced basis. However, there is an inherent trade-off between these two steps. A stronger lattice reduction drastically decreases the time required for the subsequent search step, but incurs a significantly higher computational cost itself, and vice versa. Therefore, finding the optimal configuration to minimize the overall cost is highly non-trivial. This naturally leads to our second research question:

Question 2 (Q2) *How should the computational effort be divided between the Reduction and Search steps of a two-step strategy so as to minimize the total estimated cost?*

Furthermore, the continuous emergence of advanced lattice cryptanalysis techniques in recent years naturally raises a more comprehensive question. How do we resolve the trade-offs identified in **Q1** and **Q2** when taking into account a series of state-of-the-art methods, such as progressive sieving [LM18], dimensions-for-free (`d4f`) [Duc18], PnJBKZ [ADH+19, DSvW21], and refined success probability estimation [PV21, XWW+24]? This culminates in our third research question:

Question 3 (Q3) *How can we construct a model-optimal two-step strategy within the specified candidate space that integrates these state-of-the-art lattice analysis techniques to minimize the overall time cost?*

While **Q1**∼**Q3** focus on the attacker's perspective of minimizing computational cost, a critical question remains for the defensive side. For standardization and scheme designers, it is imperative to quantify the practical implications of these threats. Therefore, the last question arose:

Question 4 (Q4) *What is the precise impact (measured in bits of security) of these optimized two-step primal attack strategies on practical lattice-based cryptosystems, particularly the NIST-standardized algorithms?*

In this paper, we aim to comprehensively answer the above questions by presenting a minimum estimated-cost strategy within the defined candidate space and model for solving LWE via the primal uSVP framework.

1.2 Organization and Technical Overview

In Section 3.1, we begin by reviewing, comparing, and analyzing the existing methodologies for invoking lattice reduction and SVP algorithms. We then outline our approach to addressing the aforementioned questions and present the organization of this paper.

To address **Q3** and **Q4**, we decompose these questions into several subproblems, solve them step by step, and finally present a comprehensive solution to **Q3** and **Q4**. Specifically, we will address **Q3** and **Q4** by proceeding through the following steps:

Objective 1: Precisely and formally define the LWE solving strategy space, which is composed of all admissible combinations within the explicitly specified finite parameter ranges of lattice basis reduction steps and SVP-solving algorithms.

Objective 2: For a given LWE instance, demonstrate that the search for an optimal strategy can be restricted to a finite set under a cost bound, ensuring the existence of a minimum-cost strategy.

Objective 3: Develop an efficient search algorithm to identify the LWE solving strategy with minimum estimated cost in strategy space.

Objective 4: Analyze the impact of such optimization of solving strategy on concrete security strength of NIST-standardized lattice-based schemes.

First, Section 2 presents the notations and preliminaries of this paper. Then we will accomplish **Objective 1** in Section 3.2 and **Objective 2** in Section 4.

Objective 3 is more complex and requires the development of several technical components:

Part I: The framework of a fast **strategy search algorithm**, which is designed to explore the LWE solving strategy space using pruning techniques, aiming to identify the strategy with the minimum estimated cost.

Part II: An accurate **SVP dimension estimator** that is proposed for the SVP dimension in the Search step, which accurately calculates the success probability of solving LWE by accounting for both the SVP dimension and the randomness in the target vector's norm.

Part III: An accurate **PnJBKZ simulator** that can accurately predict the practical reduction effort of the PnJBKZ algorithm with jump strategy and LLL algorithm in polynomial time.

The classical BKZ 2.0 simulator [CN11] processes consecutive blocks and therefore does not directly model PnJBKZ with a nontrivial jump parameter $J > 1$. After a Pump call starting at index i, the log-GS lengths at the $J - 1$ intervening indices are modified but are not determined by the standard BKZ recurrence before the next block is processed. We estimate these missing values by modelling the block processed by Pump as approximately HKZ-reduced and combining the Gaussian Heuristic with preservation of the relevant projected volume. This yields a polynomial-time simulator for PnJBKZ with nontrivial jump parameters. The complete recurrence and its refinements for practical `d4f` settings and LLL reduction are presented in Section 5.

Part IV: An accurate **`d4f` estimator**, which can accurately calculate the achievable `d4f` value by accounting for the gradual improvement of lattice basis quality during the reduction process.

Under GSA, the asymptotic value of `d4f` can be estimated f as $f \approx \frac{d\ln(4/3)}{\ln(d/2\pi e)}$. This estimation of `d4f` given in [Duc18] is 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.

Our idea is to consider the impact of current lattice basis quality, then we can give a more refined estimation of `d4f`, based on the information of current reduction quality of the lattice basis, like $\delta(\mathbf{B})$.

$$f \leq \log_\delta \sqrt{\frac{4}{3}} \approx \ln(4/3)/(-\text{slope}).$$

Slope represents the slope of the logarithm of the Gram-Schmidt norms $\|\mathbf{b}_i^*\|$.

Part I is detailed in Section 4; **Part II** and **III** are presented in Sections 6.2 and 5, with the latter's practical estimation method further discussed in Section 5.1; Finally, **Part IV** and its theoretical analysis are covered in Section 5.2.

A refined LWE estimator operating in a two-step mode is introduced in Section 6 to accurately assess the impact of the solving strategy optimization on the concrete security strength of lattice-based schemes. Building on this, Section 7 presents final security estimations for NIST-standardized schemes, fully accounting for the effects of strategy optimization. Thus, **Objective 4** has been completed in Section 7.

Finally, the performance of **Parts I** $\sim$ **IV** has been extensively validated through experiments, with the results summarized in Section 8.

We outline the organization of this paper in Figure 1 and provide a technical overview in Figure 2.

Figure 3 summarizes the development of this research line. An early integrated version of this work appeared as IACR ePrint 2022/1343. An improved version of the PnJBKZ simulator was subsequently developed in our AsiaCCS 2023 paper [WWW23], while the two-step LWE estimator was developed in our PKC 2024 paper [XWW+24]. The present version reunifies these components and integrates them with PSSearch, practical cost modelling, and the end-to-end experimental evaluation.

1.3 Contributions

A more accurate estimation of security strength facilitates a better trade-off for designers of lattice-based cryptographic schemes: it enables the optimization of both scheme size and efficiency while still guaranteeing adequate security. The most important contribution of this work lies in determining precisely how much the optimization of LWE-solving strategies affects the bit security estimation of lattice-based schemes. To accomplish this, we introduce a series of new concepts and approaches. In the following, we outline the key contributions of this work.

1. We formalize the two-step LWE-solving strategy space in Section 3.2. By establishing a positive lower bound on the single-step cost in Theorem 1, we demonstrate that under a finite parameter set and cost budget T_0, the strategy space explored by PSSearch is strictly finite. In Section 4.2.2, we then prove

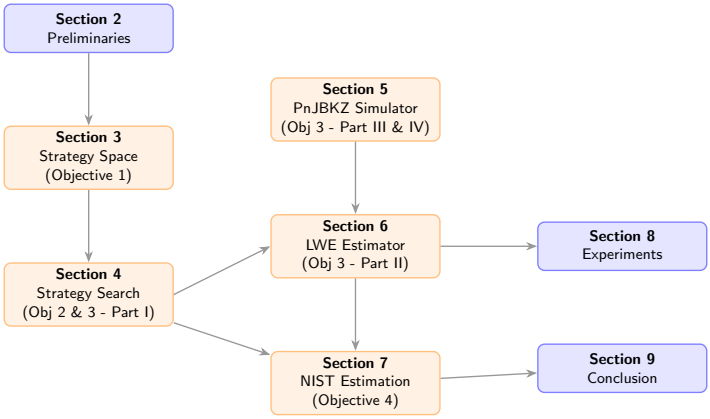


Fig. 1: Organization of this paper.

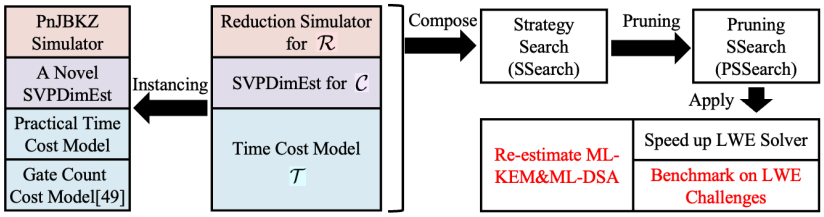


Fig. 2: Technical overview

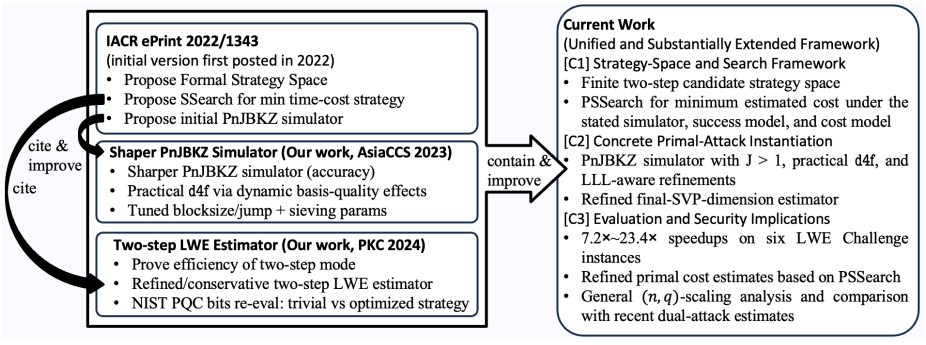


Fig. 3: Evolution of this work and its relationship to the preliminary ePrint and conference versions. The AsiaCCS 2023 and PKC 2024 papers contain two components of the early integrated research program, while the present journal article reunifies and further extends the complete framework.

that PSSearch returns a minimum modeled-cost strategy within this candidate set under the specified simulator, dimension estimator, and cost model.

2. We implement PSSearch using the most efficient lattice reduction and SVP algorithms available to date, namely PnJBKZ and Pump in G6K.

(1) Specifically, in Section 5 we design the first simulator for PnJBKZ that can simulate the reduction effect of PnJBKZ with a nontrivial jump parameter $J > 1$, enabling the search for more flexible and efficient reduction strategies. We further improve the simulator by accounting for the effect of LLL reduction, as described in Section 5.1.1.

(2) Additionally, we propose a refined dimension estimation method for the SVP oracle, taking into account the distribution of the LWE noise vector in Section 6.2.

3. The most significant conclusion is that the optimization of solving strategies significantly impacts both the practical solving of LWE and the concrete security estimation of lattice-based schemes. We summarize these impacts as follows:

(1) TU Darmstadt LWE Challenges. In particular, using the minimum time-cost Two-step strategy for solving LWE generated by PSSearch, we successfully cracked six previously unsolved TU Darmstadt LWE Challenges[1], achieving a speedup of 23.4 times compared to the previously most efficient LWE solver, G6K [DSvW21]. For details, see Figs. 4, 9, and Table 4. In the reduction step, the time cost needed to achieve the same basis quality is 4 to 36.7 times faster than the trivial reduction strategy. For more details, see Figure 10. The executable code of TwoStepSolver based on PSSearch is publicly available on GitHub[2]. Furthermore, a recent work [ZD26] introduces a sieving implementation that extends memory to the hard drive, enabling the efficient execution of higher-dimensional sieving. Adapting our strategy optimization algorithm to build upon this expanded memory framework represents a promising direction for future work.

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

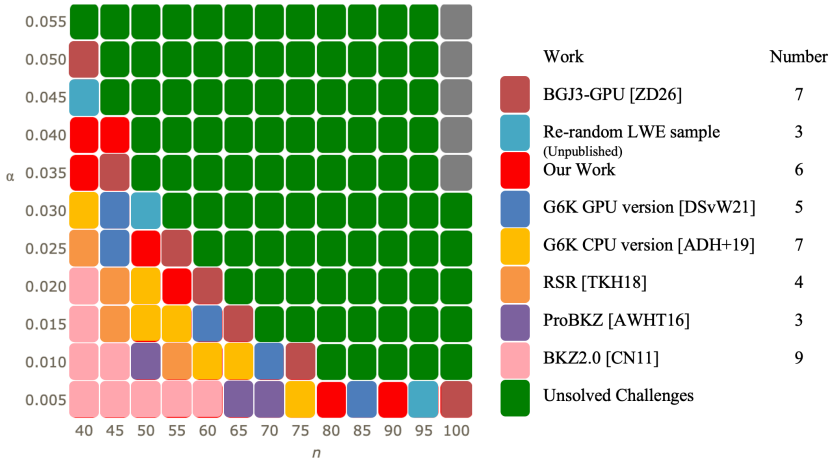


Fig. 4: LWE Challenges and the Algorithms to Solve it

(2) Impact on NIST standards schemes. Our new hardness estimation for the NIST standards ML-KEM and ML-DSA shows that, when using the minimum time-cost strategy, the modeled costs of the primal-attack branch decrease by $3.4 \sim 4.6$ bits, rather than 2 to 8 bits indicated in the Kyber documentation.

(3) More General Conclusions. To quantify the impact of solving-strategy optimization on LWE, we evaluated its efficiency across randomly generated LWE instances with varying parameters rather than relying solely on fixed relationships found in NIST standards. Specifically, we fixed the error standard deviation σ and varied the lattice dimension n and the modulus q, measuring the time cost for security evaluation using the gate-count metric. Our empirical evaluation reveals that the time cost advantage of our optimized strategy S_{op} over the trivial strategy S_0 expands significantly when n and q increase concurrently. As illustrated by the base-2 logarithmic time complexity difference, denoted as $\Delta(\log_2 T) = \log_2(T(\mathsf{S}_0)) - \log_2(T(\mathsf{S}_{\mathrm{op}}))$, the acceleration effect grows substantially across the parameter space. By extracting the maximum advantage points, we empirically fitted a trend line demonstrating that the maximum advantage follows the relation $\Delta(\log_2 T) = 0.01887n - 0.98$ under the specific scaling relationship $\log_2 q = 0.198n^{0.832}$. This empirical fit indicates that, along the tested scaling relation, the speedup factor of S_{op} over S_0 grows exponentially with n. See Figure 13 in detail.

Additionally, we note that several works have been published on accelerating sieving implementations, such as [ZDY25] and [DSvW21], which have effectively increased the practical solving speed of hard lattice problems like SVP and LWE. We wish to emphasize that our work is orthogonal to those research focused on accelerating SVP-solving algorithms. Our study investigates how the optimization of solving strategies influences the total time cost of solving LWE. Regardless of whether an accelerated SVP-solving algorithm is adopted—including

more advanced solvers that may emerge in future research—our method can be integrated with any such advanced algorithm. This is achieved by adjusting the corresponding time-cost model in the solving strategy according to the new SVP solver or lattice reduction algorithm. By doing so, based on our algorithm the fastest solving strategy can be identified, thereby further accelerating the speed of solving LWE.

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 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. For a full-column-rank basis matrix $\mathbf{B} \in \mathbb{R}^{m \times d}$, define $\mathrm{vol}(\mathcal{L}(\mathbf{B})) := \sqrt{\det(\mathbf{B}^\intercal \mathbf{B})} = \prod_{i=0}^{d-1} \|\mathbf{b}_i^*\|$. When $m = d$, this equals $|\det(\mathbf{B})|$. $A := [\![a_0, ..., a_{n-1}]\!]$ represents an ordered list of size $\sharp A = n$ with $A[i] = a_i$. Additionally, we have $[\![a_0, ..., a_{n-1}]\!] \| [\![a_n]\!] = [\![a_0, ..., a_{n-1}, a_n]\!]$. We define $[a : b) := \{a, a+1, ..., b-1\}$ and $[d) = [0 : d)$.

Definition 1 (The Gaussian Distribution [LG08]). *Let $\sigma > 0$ and $\mu \in \mathbb{R}$ be the standard deviation and the mean value respectively, a continuous Gaussian Distribution denoted as $\mathcal{N}_{\mu,\sigma}$. If $\mu = 0$, then denote $\mathcal{N}_\sigma = \mathcal{N}_{0,\sigma}$. Its probabilistic density function $\rho_{\mathcal{N}_{\mu,\sigma}}(x) = e^{-\frac{(x-\mu)^2}{2\sigma^2}} / \sigma\sqrt{2\pi}$.*

Definition 2 (The discrete Gaussian Distribution [PV21]). *Let $\sigma > 0$ and $\mu \in \mathbb{R}$ be the standard deviation and the mean value respectively, a discrete Gaussian Distribution denoted as $\mathcal{D}_{\mu,\sigma}$. If $\mu = 0$, then denote $\mathcal{D}_\sigma = \mathcal{D}_{0,\sigma}$. For $x \in \mathbb{Z}$, its probability mass function is $f_{\mathcal{D}_{\mu,\sigma}}(x) = \frac{\rho_{\mathcal{N}_{\mu,\sigma}}(x)}{\sum_{z \in \mathbb{Z}} \rho_{\mathcal{N}_{\mu,\sigma}}(z)}$.*

Definition 3 (Chi-squared distribution [LG08]). *Let $Z_0, \cdots, Z_{k-1}$ be independent standard normal random variables. Then $\sum_{i=0}^{k-1} Z_i^2$ follows the chi-squared distribution with k degrees of freedom, denoted by χ_k^2. It has mean k and variance $2k$.*

If $Y_0, \cdots, Y_{k-1}$ are independent random variables distributed as $\mathcal{N}(0, \sigma^2)$, then $\sum_{i=0}^{k-1} Y_i^2 \sim \sigma^2 \chi_k^2$, which has mean $k\sigma^2$ and variance $2k\sigma^4$. We denote by $F_{\chi_k^2}$ the cumulative distribution function of χ_k^2, namely $F_{\chi_k^2}(y) := \Pr_{Z \sim \chi_k^2}[Z \le y]$.

2.1 Lattices and Lattice Hard Problems

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 $i \in [d)$. Let $\mathsf{rr}_{\mathbf{B}} := (l_0, \cdots, l_{d-1})$, where $l_i = \ln \|\mathbf{b}_i^*\|$, denote the log-GS profile of $\mathbf{B}$. When the basis is clear from the context, we abbreviate $\mathsf{rr}_{\mathbf{B}}$ as rr, and write $\mathsf{rr}_{[i:j)} := (l_i, \cdots, l_{j-1})$. For $i \in [d)$, we denote by π_i the orthogonal projection onto $(\mathbf{b}_0, ..., \mathbf{b}_{i-1})^{\perp}$. For $0 \leq j < k \leq d$, 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 $\lambda_1(\mathcal{L}) \approx \mathrm{GH}(\mathbf{B}) = \sqrt{\frac{d}{2\pi e}} |\mathbf{B}|^{1/d}$. For a log-GS subprofile $\mathsf{rr}[a : b)$, suppose $\mathrm{GH}\big(\mathsf{rr}[a : b)\big) := \sqrt{\frac{b-a}{2\pi e}} \exp\left(\frac{1}{b-a} \sum_{j=a}^{b-1} \mathsf{rr}[j]\right)$.

Definition 4 (unique Shortest Vector Problem(uSVP$_\gamma$) [LM09]). *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 5 ($\mathrm{LWE}_{n,m,q,\mathcal{D}_\sigma}$ Distribution [APS15, Pei16, Xag10]). *Given some samples $m \in \mathbb{Z}$, a secret vector dimension $n \in \mathbb{Z}$, a modulo $q \in \mathbb{Z}$, a probability distribution $\mathcal{D}_\sigma$. 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 $\mathcal{D}_\sigma$ 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 6 (Search $\mathrm{LWE}_{n,m,q,\mathcal{D}_\sigma}$ problem [APS15,Pei16,Xag10]). *Given a pair $(\mathbf{A}, \mathbf{b})$ sampled from $\mathrm{LWE}_{n,m,q,\mathcal{D}_\sigma}$ distribution Ψ compute the pair $(\mathbf{s}, \mathbf{e})$.*

2.2 Lattice Reduction

Definition 7 (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 8 (Hermite-Korkine-Zolotarev and Block-Korkine-Zolotarev reductions [Ngu09]). *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 9 (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* $\delta(\beta)^{2(\beta-1)} = \frac{\beta}{2\pi e}(\beta\pi)^{1/\beta}$ *[Che13].*

$\delta(\mathbf{B})$ 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 [ADH+19]) *Let* $\mathbf{B}$ *be a lattice basis after the lattice reduction, then* $\|\mathbf{b}_i^*\| \approx \rho \cdot \|\mathbf{b}_{i-1}^*\|$, $0 < \rho < 1$.

By Combining the GSA with the root Hermite factor (Definition 9) and noting that $\mathrm{Vol}(\mathcal{L}) = \prod_{i=0}^{d-1}\|\mathbf{b}_i^*\|$, we infer that $\rho = \delta^{-\frac{2d}{d-1}} \approx \delta^{-2}$. Let s be the slope value of the logarithm of GS norms l_i, $\forall i \in [d)$, $s \approx \ln\rho$ 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 [ADH+19], 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 [NV08]. Later improvements, focusing on nearest neighbor search techniques, reduced the time complexity to $O(2^{0.292d})$ [Laa15, BGJ15, ZDY25]. 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 $O(2^{0.3588d})$ [HKL18].

Progressive sieve [LM18] 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 $\mathbf{B}_{[d_{\mathrm{svp}})}$ to $\mathbf{B}_{[d_{\mathrm{svp}}+1)}$, gradually increasing the dimension d_{svp}. In contrast, the left-progressive sieve [ADH+19, DSvW21] first performs sieving in the projected sublattice basis $\mathbf{B}_{\pi[d-d_{\mathrm{svp}}:d)}$, then lifts the vectors using Babai's nearest plane algorithm [Bab86] to $\mathbf{B}_{\pi[d-d_{\mathrm{svp}}-1:d)}$ 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.

Lattice Solving Algorithms. BKZ [SE91] is the most popular lattice reduction algorithm for solving LWE. It combines the LLL [LLL82] algorithm with an SVP solver to balance the time cost and the success probability using block-size β. Many cryptanalysts have improved the BKZ algorithm, such as through the extreme pruning [GNR10] to speed up enumeration, BKZ 2.0 [CN11] based

on [GNR10], approximate enumeration oracle [ABLR21], and progressive reduction parameters optimization in BKZ, like *Improved Progressive BKZ* (ProBKZ) [AWHT16]. [AGVW17] 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 [ADPS16].

In 2019, Albrecht et al. [ADH+19] designed the *General Sieve Kernel* (G6K) and implemented the progressive sieve [LM18], named Pump, which selectively employs various sieving algorithms: the Gauss sieve [MV10, FBB+15], the NV sieve [NV08], the k-list sieve [HK17, HKL18] or the BGJ1 sieve [BGJ15]. Pump utilizes the progressive sieve and dimension-for-free (`d4f`) technique [Duc18] 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. [DSvW21] further improved G6K with GPU acceleration (named G6K-GPU-Tensor) and implemented the fastest sieving algorithm, BDGL16 [BDGL16], 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 [CN11] and uses the Gaussian heuristic to predict BKZ-β. In 2016, Aono et al. introduced a new BKZ simulator in [AWHT16] as part of their proposal for Progressive BKZ, aimed at predicting a fully BKZ-β-reduced basis. In 2017, Yu and Ducas [YD18] 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 [BSW18] proposed an even more accurate BKZ simulator by considering the distribution of short vectors in random lattices. The simulator developed by Bai et al. [BSW18] 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. [BSW18] 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 [GN08, CN11, ADH+19, DSvW21], only one tour of

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

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.

2.4 Techniques in G6K

G6K [ADH+19] is an abstract machine designed to run sieve and reduction algorithms, built by generalizing and extending the previous sieving algorithms. G6K-GPU-Tensor [DSvW21], 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. The `d4f` technique [Duc18] can provide sub-exponential time speedup and a reduction in memory for sieving algorithms. [Duc18] provides two theoretical `d4f` estimations for solving the β-dimension SVP: $\texttt{d4f}(\beta) = \beta\ln(4/3)/\ln(\beta/2\pi)$ and $\texttt{d4f}(\beta) = \beta\ln(4/3)/\ln(\beta/2\pi e)$, while in the implementation of G6K [ADH+19], a more relaxed value, referred to as the "optimistic `d4f`," is used. $\beta<40$, $\texttt{d4f}_{\text{op}}(\beta) = 0$; $40 \leq \beta \leq 75$, $\texttt{d4f}_{\text{op}}(\beta) = \lfloor(\beta-40)/2\rfloor$; $\beta>75$, $\text{d4f}_{\text{op}}(\beta) = \lfloor 11.5 + 0.075\beta\rfloor$.

However, the `d4f` estimations in G6K are only related to β, which is considered an overestimate. [WWW23] 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 $\texttt{d4f}_\delta = \ln_\delta\sqrt{4/3} \approx -\ln(4/3)/s$. Here, s represents the slope value of the logarithm of the Gram-Schmidt norms. More details about $\texttt{d4f}_{\text{slope}}(s)$ can be found in Eq. (5) in [WWW23].

Pump in G6K. Albrecht et al. proposed a sieving style algorithm, Pump, in [ADH+19], which combines *Progressive Sieve* [LM18] with the `d4f` technique [Duc18] 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 $\texttt{d4f}(d_{\text{svp}})$. Here, $\kappa+d_{\text{svp}}=d$ and the upper bound of sieve dimension is $d_{\text{svp}} - \texttt{d4f}(d_{\text{svp}})$. The Pump will insert up to $d_{\text{svp}} - \texttt{d4f}(d_{\text{svp}})$ short vectors into the basis index from κ to d, choosing the shortest vectors from the sieved set on the projected sublattice $\mathcal{L}_{\pi[\kappa+i:d)}$ for i from $d_{\text{svp}}-1$ to $\texttt{d4f}(d_{\text{svp}})$. In this way, it performs a partial HKZ reduction and outputs a nearly HKZ-reduced basis, as described in the G6K paper [ADH+19].

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* [AFG14] transforms LWE into a uSVP by constructing a special embedding lattice basis $\mathbf{B}$ using Kannan's embedding technique [Kan83], where $\mathbf{B} = \begin{pmatrix} \bar{\mathbf{A}} & \mathbf{b} \\ \mathbf{0} & 1 \end{pmatrix}$. The resulting embedding lattice has dimension $d = m + 1$. 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 a 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, a standalone SVP oracle is applied to a projected sublattice of the reduced basis. Given a reduced d-dimensional basis $\mathbf{B}$, existing LWE solvers commonly replace the norm of the projected target vector by the root-mean-square estimate $\sigma\sqrt{k}$ for a k-dimensional projection. Accordingly, the classical estimate of the required SVP dimension is $d_{\text{svp}}^{\text{cl}} := \min\left\{k \in [1 : d+1) : \sigma\sqrt{k} \leq \text{GH}\left(\mathbf{B}_{\pi[d-k:d)}\right)\right\}$. This expected-squared-norm criterion is used in several practical LWE solvers [ADH+19, DSvW21].

In this paper, we build upon the probabilistic model using BKZ simulation proposed in [DSDGR20]. Specifically, we adapt the BKZ-based estimator to fit the SVP model, as detailed in Section 6.2, 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 log-GS profile 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 [ABF+20, ABLR21]. 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: $T_{\text{Pump}}(\beta) = 2\sum_{j=\beta_0}^{\beta} T_{\text{sieve}}(j) \approx 2 \cdot \frac{2^{c(\beta+1)} - 2^{c\beta_0}}{2^c - 1} \cdot 2^{o(\beta)} \approx 2^{c\cdot\beta + c_1}$, 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 Figure 7 in [DSvW21], 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 Figure 17. More details on the practical Pump cost model can be found in Appendix A.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 $T_{\text{BKZ}}(\beta, J) = (d-\beta)/J \cdot T_{\text{Pump}}(\beta)$. 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 A.2.

3 A Unified Strategy Space for Primal LWE Attacks

Primal attacks reduce an LWE instance to a uSVP instance on a q-ary lattice basis constructed through an embedding technique [AFG14, AGVW17]. Solving the resulting uSVP instance generally involves an ordered sequence of lattice basis reductions and, when needed, a standalone SVP search on the reduced basis or one of its projected sublattices [LN13, AWHT16, XWW+24]. In a BKZ-only strategy, the target vector may be recovered during the reduction process itself. In a Two-step strategy, the reduction process is terminated at an intermediate basis quality and is followed by a separate Search step using enumeration or sieving.

The concrete performance of such an attack depends not only on the reduction and SVP-solving algorithms being used, but also on their parameters, invocation order, stopping point, and the dimension of the final SVP call. Different choices of these components may lead to substantially different time costs, even when they are applied to the same LWE instance.

Despite their different algorithmic configurations, existing LWE-solving methods share a common structure. At the algorithmic level, a solving strategy specifies

- an ordered reduction strategy that determines how the lattice basis is progressively preprocessed; and
- when a standalone Search step is used, an SVP oracle together with the dimension of the projected sublattice on which it is invoked.

A time-cost model is then used to evaluate a given strategy. In particular, these models determine the probability that the target vector is recovered during the Reduction or Search step and the corresponding expected solving cost. They are therefore evaluation components associated with the strategy space, rather than algorithmic choices constituting an individual strategy.

In this section, we formalize the common algorithmic structure above as a unified strategy space for primal attacks through uSVP. This abstraction separates the specification of a solving strategy from the models used to evaluate it, thereby allowing different reduction algorithms, SVP oracles, simulators and time-cost models to be combined in a modular manner.

We begin in Section 3.1 by reviewing representative LWE-solving strategies from prior works [ADPS16, AWHT16, ADH+19, DSvW21] and expressing their reduction procedures using a common notation. These examples motivate the formal definitions in Section 3.2, where we introduce Generic Reduction, Generic Basis Quality, and the corresponding LWE-solving strategy space. Finally, Section 3.3 discusses how the same framework can be adapted to other lattice attacks that employ lattice basis preprocessing.

3.1 Representative LWE-Solving Strategies in Prior Work

The primal attack described in Section 2.5 transforms an LWE instance into a d-dimensional uSVP instance on a lattice basis $\mathbf{B}$. The resulting uSVP instance is then solved by combining lattice basis reduction with SVP-solving algorithms [AFG14, AGVW17]. We refer to the concrete choice and scheduling of these algorithms as an *LWE-solving strategy*.

To compare representative strategies from prior works, we use the notation that will be formalized in Section 3.2. Let $\mathbf{B}_0 = \mathrm{LLL}(\mathbf{B})$ be the initial LLL-reduced basis. As Definition 10, let $\mathcal{R}(\mathbf{B}, \xi)$ (or simply $\mathcal{R}$-ξ) be one tour of a lattice reduction algorithm $\mathcal{R}$ whose elementary SVP-insertion operations are specified by the ordered blocksize list ξ.

For a tour of BKZ-β and PnJBKZ-(β, J), we denote the corresponding generic blocksize lists by $\xi_{\mathrm{BKZ}}(\beta)$ and $\xi_{\mathrm{PnJBKZ}}(\beta, J)$, respectively. Their formal instantiations are given in Table 1.

Following Definition 12, for a given lattice reduction algorithm $\mathcal{R}$ and an SVP oracle $\mathcal{C}$, an ordered sequence of reduction tours is written as $\mathsf{S} = [\![\xi_0, \cdots, \xi_{r-1}]\!]$, where the basis is iteratively updated as $\mathbf{B}_{j+1} = \mathcal{R}(\mathbf{B}_j, \xi_j)$ for $j \in [r)$. For simplicity, when the choice of the reduction algorithm $\mathcal{R}$ is clear from the context, we often abbreviate the strategy representation by directly listing its parameters. For instance, if $\mathcal{R}$ is BKZ, let $\mathsf{S} = [\![\beta_0, \cdots, \beta_{r-1}]\!]$ instead of $[\![\xi_{\mathrm{BKZ}}(\beta_0), \cdots, \xi_{\mathrm{BKZ}}(\beta_{r-1})]\!]$; if $\mathcal{R}$ is PnJBKZ, we write $\mathsf{S} = [\![(\beta_0, J_0), \cdots, (\beta_{r-1}, J_{r-1})]\!]$. After executing the reduction sequence S, the Search step is initiated by calling the standalone SVP oracle $\mathcal{C}$ on the projected sublattice $(\mathbf{B}_r)_{\pi[d-d_{\mathrm{svp}}:d)}$ to obtain the projection of the target vector, which is then lifted to the full-dimensional lattice. We use $(\mathsf{S}, d_{\mathrm{svp}}) \in \mathcal{S}(\mathcal{R}, \mathcal{C})$ to denote the resulting Two-step solving procedure within this strategy space. By convention, $d_{\mathrm{svp}} = 0$ means that the standalone SVP oracle $\mathcal{C}$ is not called, and the target vector must be recovered purely during the reduction process. The precise definitions of these objects are given in Section 3.2.

3.1.1 Fixed-Blocksize BKZ in ADPS16. The LWE-solving strategy considered in ADPS16 [ADPS16] applies BKZ with a fixed blocksize β_{ADPS}. For

an LWE instance with parameters (n, q, σ), the blocksize β_{ADPS} is selected as the smallest β satisfying $\sigma\sqrt{\beta} \leq \delta(\beta)^{2\beta-d-1} \cdot q^{m/d}$, where $\delta(\beta)$ denotes the root Hermite factor associated with BKZ-β reduction [ADPS16].

Following the above notation, one tour of this reduction is described by $\xi_{\mathrm{ADPS}} = \xi_{\mathrm{BKZ}}(\beta_{\mathrm{ADPS}})$. Thus, except for the standard truncation near the end of the basis, the elementary SVP calls use the same blocksize β_{ADPS} at every index.

If the target vector is recovered after r tours, the corresponding reduction strategy is $\mathsf{S}^{(r)}_{\mathrm{ADPS}} = [\![\underbrace{\xi_{\mathrm{ADPS}}, \cdots, \xi_{\mathrm{ADPS}}}_{r \text{ tours}}]\!]$, and the complete solving strategy is represented by $(S^{(r)}_{\mathrm{ADPS}}, 0)$. In particular, no separate Search step is performed; the target vector is expected to be recovered during one of the BKZ tours [CN11, ADPS16].

The success condition above describes the quality expected from a sufficiently reduced BKZ-β_{ADPS} basis. However, the initial LLL-reduced basis generally does not already have this reduction quality. Consequently, several tours with the same relatively large blocksize may be required before the target vector can be recovered [CN11, ADPS16]. This strategy therefore commits to the final blocksize from the beginning of the reduction process.

3.1.2 Heuristic Progressive BKZ. A common alternative is to increase the BKZ blocksize progressively rather than repeatedly applying the same large blocksize [GN08, CN11, HP13, SS12]. Let $\beta_0 \leq \beta_1 \leq \cdots \leq \beta_{r-1}$ be a sequence of blocksizes, let the strategy be $\mathsf{S}_{\mathrm{prog}} = [\![\xi_{\mathrm{BKZ}}(\beta_0), \xi_{\mathrm{BKZ}}(\beta_1), \cdots, \xi_{\mathrm{BKZ}}(\beta_{r-1})]\!]$.

Different progressive BKZ strategies prescribe different rules for selecting the sequence $\mathsf{S}_{\mathrm{prog}}$. For example, the blocksize may be increased by a fixed additive step $\Delta\beta$ (e.g. $\Delta\beta = 1$ in [GN08], $\Delta\beta = 2$ in [HP13], and $\Delta\beta = 10$ in [CN11]), or according to a doubling-type schedule (e.g. $\Delta\beta = -\beta + 2^i$ for $i = 1, 2, \ldots$ as in [SS12]). Repeated blocksizes may also be included when several tours are performed at the same reduction strength.

The motivation for progressive reduction is that inexpensive small-block tours first improve the initially poor-quality basis, thereby reducing the cost of the SVP subroutines invoked in subsequent large-block tours. This effect has been studied for BKZ and slide-reduction strategies in [SS12], for enumeration-based BKZ in [AWHT16], and for sieving-based reduction with the dimension-for-free technique in [WWW23].

Nevertheless, the blocksize sequence and its stopping point are typically determined by a fixed heuristic rule rather than by minimizing the total cost for the particular LWE instance. For the BKZ-only variants considered here, no standalone SVP oracle is called, and the resulting strategy is represented as $(\mathsf{S}_{\mathrm{prog}}, 0)$. Thus, even when the reduction algorithm is fixed to BKZ, different choices of the sequence $\mathsf{S}_{\mathrm{prog}}$ may result in substantially different solving costs.

3.1.3 ProBKZ. Aono et al. [AWHT16] proposed Improved Progressive BKZ, referred to here as ProBKZ, to select a progressive reduction sequence through

a model-based optimization procedure. ProBKZ constructs a graph whose nodes represent prescribed intermediate basis-quality levels, usually modeled by fully BKZ-β-reduced basis profiles. An edge represents the reduction procedure required to transform one basis-quality profile into another, and its weight is the estimated cost of this transformation. Dijkstra's algorithm is then used to find a low-cost path in this graph [AWHT16].

Using the common notation, the reduction sequence selected by ProBKZ has the form $\mathsf{S}_{\mathrm{ProBKZ}} = [\![\xi_{\mathrm{BKZ}}(\beta_0), \cdots, \xi_{\mathrm{BKZ}}(\beta_{r-1})]\!]$, where the blocksize sequence is obtained from the shortest path rather than from a predetermined additive or geometric rule.

Unlike the preceding BKZ-only strategies, ProBKZ does not require the target vector to be recovered during the reduction process. After executing $\mathsf{S}_{\mathrm{ProBKZ}}$, a standalone enumeration algorithm is applied to the full-dimensional preprocessed basis [AWHT16]. Therefore, the complete strategy is represented by $(\mathsf{S}_{\mathrm{ProBKZ}}, d)$, with $\mathcal{R} = \mathrm{BKZ}$, and $\mathcal{C} = \mathrm{Enumeration}$.

This separation between lattice reduction and a final standalone search is commonly referred to as the *Two-step mode*. It consists of a Reduction step, which progressively improves the basis, followed by a Search step, which invokes a dedicated algorithm to recover the target vector from the reduced basis [LN13, AWHT16, XWW+24]. Stronger preprocessing increases the reduction cost but decreases the cost of the final search, whereas weaker preprocessing has the opposite effect. The effectiveness of this trade-off for solving LWE was further analyzed in [XWW+24].

ProBKZ provides a systematic method for balancing these two costs. However, the optimization is performed only over the prescribed graph of intermediate profiles. In particular, the nodes are tied to selected fully BKZ-reduced profiles, and the final Search step is fixed to full-dimensional enumeration. Hence, ProBKZ identifies a minimum-cost path within this restricted graph rather than over all possible reduction sequences and final SVP dimensions [AWHT16].

3.1.4 The PnJBKZ–Pump Strategy in G6K. The LWE solvers implemented in G6K and G6K-GPU-Tensor also follow the Two-step mode, but use PnJBKZ as the reduction algorithm and Pump as the standalone SVP oracle [ADH+19, DSvW21]. Let $(\beta_0, J_0), \cdots, (\beta_{r-1}, J_{r-1})$ be the sequence of PnJBKZ parameters used in the Reduction step. The corresponding reduction strategy is $\mathsf{S}_{\mathrm{G6K}} = [\![\xi_{\mathrm{PnJBKZ}}(\beta_0, J_0), \cdots, \xi_{\mathrm{PnJBKZ}}(\beta_{r-1}, J_{r-1})]\!]$. Here, each $\xi_{\mathrm{PnJBKZ}}(\beta_j, J_j)$ is the generic blocksize list associated with one PnJBKZ-(β_j, J_j) tour, as specified in Table 1. Pump is used internally by PnJBKZ to perform local reductions, while the symbol $\mathcal{C} = \mathrm{Pump}$ below refers only to the standalone Pump invoked after the reduction sequence.

After executing $\mathsf{S}_{\mathrm{G6K}}$, G6K applies a standalone Pump to a projected suffix of the reduced basis. Pump combines progressive sieving with the dimension-for-free technique [ADH+19, Duc18]. Once the projection of the target vector has been recovered, it is lifted to the full-dimensional lattice using Babai's nearest-plane algorithm [Bab86]. The complete strategy therefore has the form

$(\mathsf{S}_{\mathrm{G6K}}, d_{\mathrm{svp}})$ with $\mathcal{R} = \mathrm{PnJBKZ}$ and $\mathcal{C} = \mathrm{Pump}$, where d_{svp} is the dimension of the projected sublattice processed by the standalone Pump.

In G6K and its GPU version, d_{svp} is commonly estimated by finding the smallest dimension satisfying an expected-norm condition of the form $\sigma\sqrt{d_{\mathrm{svp}}} \leq \mathrm{GH}\left((\mathbf{B}_r)_{\pi[d-d_{\mathrm{svp}}:d)}\right)$ [AWHT16,AGVW17,ADH+19,DSvW21]. However, the projected target norm is a random variable rather than a fixed expected value. Consequently, the expected-norm condition alone does not precisely determine the success probability of the final Search step [BMW19, DSDGR20, PV21].

Although G6K employs efficient sieving-based reduction and search algorithms, its reduction sequence $\mathsf{S}_{\mathrm{G6K}}$ and the transition point from PnJBKZ reduction to the standalone Pump are still selected heuristically [ADH+19, DSvW21]. Optimizing this strategy therefore requires jointly considering the reduction effect and cost of each PnJBKZ tour, the quality of the intermediate basis, and the dimension and success probability of the final Pump.

3.1.5 From existing strategies to a unified strategy space. The strategies reviewed above differ primarily in the admissible form of the reduction sequence and in the use of the final SVP oracle. ADPS16 repeatedly applies the same fixed-blocksize list ξ_{ADPS} [ADPS16]. Heuristic progressive BKZ uses a predetermined sequence of BKZ blocksize lists [GN08, CN11, HP13, SS12]. ProBKZ searches over a restricted graph of BKZ reduction sequences and follows the selected sequence with full-dimensional enumeration [AWHT16]. G6K uses a heuristic sequence of PnJBKZ blocksize lists followed by a standalone Pump on a projected sublattice [ADH+19, DSvW21].

Despite these differences, all of these methods can be represented by the common procedure

$$\mathbf{B}_0 \xrightarrow{\mathcal{R}\text{-}\xi_0} \mathbf{B}_1 \xrightarrow{\mathcal{R}\text{-}\xi_1} \cdots \xrightarrow{\mathcal{R}\text{-}\xi_{r-1}} \mathbf{B}_r \xrightarrow{\mathcal{C},\, d_{\mathrm{svp}}} \mathbf{v},$$

where $\mathbf{v}$ denotes the target unique shortest vector corresponding to the LWE solution, and the last arrow is omitted when $d_{\mathrm{svp}} = 0$.

In this notation, selecting an efficient progressive reduction strategy amounts to selecting the sequence $\mathsf{S} = [\![\xi_0, \cdots, \xi_{r-1}]\!]$, while optimizing the Two-step mode requires jointly selecting S and the final search dimension d_{svp}. To formalize and optimize these choices, we need an algorithm-independent description of a reduction tour, a common order for comparing the quality of intermediate lattice bases, and a precise definition of the set of admissible solving strategies. These notions are introduced in the following subsection through Generic Reduction, Generic Basis Quality, and the LWE-solving strategy space.

3.2 Strategy Space

3.2.1 Generic Reduction. 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 10 (Generic Reduction ($\mathcal{R}(\mathbf{B}, \xi)$ or $\mathcal{R}$-ξ)). *Let $\mathbf{B}$ be a lattice basis with dimension d, and let $\xi = [\![(\kappa_0, \beta_0), \ldots, (\kappa_{\ell-1}, \beta_{\ell-1})]\!]$ be a finite ordered list of reduction parameters, where $\ell \in \mathbb{N}$ denotes the number of reduction steps, $\kappa_i \in [d-1)$ denotes the starting index, and $\beta_i \geq 2$ denotes the target blocksize. The generic reduction algorithm, denoted as $\mathcal{R}(\mathbf{B}, \xi)$, modifies the basis $\mathbf{B}$ by sequentially applying the following procedure for each step $i \in [\ell)$:*

1. *Define the effective blocksize as $\beta_i' = \min\{\beta_i, d - \kappa_i\}$. Find a shortest non-zero vector $\mathbf{v}$ by executing a β_i'-dimensional SVP solver on the sublattice basis $\mathbf{B}_{[\kappa_i:\kappa_i+\beta_i')}$.*
2. *Insert the vector $\mathbf{v}$ into the basis $\mathbf{B}$ at index κ_i, and remove the resulting linear dependency to maintain a valid basis by applying LLL reduction to the sequence $(\mathbf{b}_0, \ldots, \mathbf{b}_{\kappa_i-1}, \mathbf{v}, \mathbf{b}_{\kappa_i}, \ldots, \mathbf{b}_{d-1})$.*

Instantiations of Generic Reduction. Table 1 summarizes how the native parameters of several commonly used lattice reduction algorithms determine the ordered blocksize list ξ in Definition 10.

For a local HKZ reduction of a b-dimensional projected block starting at index κ, define $\xi_{\mathrm{HKZ}}(\kappa, b) := [\![(\kappa + r, b - r)]\!]_{r \in [b-1)}$. Thus, $\xi_{\mathrm{HKZ}}(\kappa, b)$ represents the successive elementary SVP-insertion operations associated with an HKZ reduction of the block $\mathbf{B}_{\pi[\kappa:\kappa+b)}$ [Kan83, NV09].

For one tour of PnJBKZ-(β, J), let $[\![(p_t, b_t)]\!]_{t \in [N)}$ denote the ordered schedule of N Pump calls described in Section 2.4, where p_t and b_t are the starting index and block dimension of the t-th Pump call, respectively. Under the idealized Pump-as-HKZ abstraction,[5] the t-th Pump call is represented by $\xi_{\mathrm{HKZ}}(p_t, b_t)$. Thus, the generic reduction parameter is the ordered list of elementary SVP-insertion operations obtained by expanding all Pump calls:

$$\xi_{\mathrm{PnJBKZ}}(\beta, J) := \Big\Vert_{t \in [N)} \xi_{\mathrm{HKZ}}(p_t, b_t),$$

where $\Vert$ denotes ordered-list concatenation.

3.2.2 Generic 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 $(\ln(\|\mathbf{b}_0^*\|), \ldots, \ln(\|\mathbf{b}_{d-1}^*\|))$. However, when evaluating the quality of a lattice basis after a tour reduction of PnJBKZ or another reduction algorithm, the GSA may no longer hold. In this paper, we provide a general definition of lattice basis quality that encompasses the reductions described in Definition 10.

[5] A practical Pump performs a partial HKZ reduction and typically outputs a nearly HKZ-reduced local basis [ADH+19, DSvW21]. The effects of practical sieving and the dimension-for-free technique are considered in Section 5.

Table 1: Instantiations of ξ for representative lattice reduction algorithms.

Reduction Algorithm	Reduction Parameter	Form of the blocksize list ξ
BKZ [SE91]	β	$\xi_{\mathrm{BKZ}}(\beta) = [\![(i,\beta)]\!]_{i\in[d-1)}$
HKZ [Kan83, NV09]	–	$\xi_{\mathrm{HKZ}}(0,d) = [\![(i,d-i)]\!]_{i\in[d-1)}$
PnJBKZ [ADH+19, DSvW21]	(β, J)	$\xi_{\mathrm{PnJBKZ}}(\beta,J) = \Vert_{t\in[N)} \xi_{\mathrm{HKZ}}(p_t,b_t)$

Definition 11 (Generic Basis Quality). *For a d-dimensional lattice basis $\mathbf{B}$ with log-GS profile $\mathsf{rr}_{\mathbf{B}}$, let $(|\mathbf{B}_{[1)}| = \|\mathbf{b}_0^*\|, |\mathbf{B}_{[2)}| = \|\mathbf{b}_0^*\| \cdot \|\mathbf{b}_1^*\|, ..., |\mathbf{B}_{[d-1)}| = \prod_{i=0}^{d-2} \|\mathbf{b}_i^*\|)$ 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 $|\mathbf{C}_{[k+1)}| \geq |\mathbf{D}_{[k+1)}|$ for all $k \in [d-1)$. This is denoted as $\mathbf{D} \geq_{\mathcal{Q}} \mathbf{C}$ or $\mathsf{rr}_{\mathbf{D}} \geq_{\mathcal{Q}} \mathsf{rr}_{\mathbf{C}}$.

We now consider a left-to-right sweep with $\ell = d-1$ and $\kappa_i = i$ for $i \in [d-1)$, so that the insertion positions are $0, \cdots, d-2$. This includes a BKZ tour with $\beta_i = \beta$ and an HKZ reduction with $\beta_i = d-i$, as specified in Table 1. Under the Gaussian Heuristic, the following property describes the simulated Gram–Schmidt norms for this sweep:

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, $\mathsf{rr}_{\mathbf{B}'} \geq_{\mathcal{Q}} \mathsf{rr}_{\mathbf{B}}$.*
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 $\mathsf{rr}_{\mathbf{D}} \geq_{\mathcal{Q}} \mathsf{rr}_{\mathbf{C}}$, then $\mathsf{rr}_{\mathbf{D}'} \geq_{\mathcal{Q}} \mathsf{rr}_{\mathbf{C}'}$.*

Proof. Item 1. (Proof by induction.) Suppose the Gram-Schmidt norms of $\mathbf{B}$ (resp. $\mathbf{B}'$) are $(x_0, \cdots, x_{d-1})$ (resp. $(x'_0, \cdots, x'_{d-1})$). 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 $k = 0$, $|\mathbf{B}'_{[1)}| = x'_0 \approx \min\{\mathrm{GH}(\mathbf{B}_{[\beta_0)}), x_0\} \leq x_0$ obviously. Assume that $|\mathbf{B}'_{[n)}| \leq |\mathbf{B}_{[n)}|$ holds, while $k = n$. When $k = n+1$, if $x_n < \sqrt{\frac{\beta'}{2\pi e}} \cdot \left(\frac{|\mathbf{B}_{[n+\beta')}|}{|\mathbf{B}'_{[n)}|}\right)^{1/\beta'}$, it implies that the norm of short vector found in $\mathbf{B}_{\pi[n:n+\beta')}$ is greater than x_n, the norm of vector in the current lattice basis. In this case, x_n will not be replaced, and $x_n = x'_n$.

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

We divide the argument into two cases: (1) $x'_n = x_n$, then $|\mathbf{B}'_{[n+1)}| = |\mathbf{B}'_{[n)}| \cdot x_n \leq |\mathbf{B}_{[n+1)}|$; (2) $x'_n = \sqrt{\frac{\beta'}{2\pi e}} \cdot \left(\frac{|\mathbf{B}_{[n+\beta')}|}{|\mathbf{B}'_{[n)}|}\right)^{1/\beta'} \leq x_n$, then

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

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

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 $\mathbf{y} = (y_0, \cdots, y_{d-1})$. $|\mathbf{C}_{[k+1)}| \geq |\mathbf{D}_{[k+1)}|$ 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 $k = 0$,

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

Assume $|\mathbf{C}'_{[n)}| \geq |\mathbf{D}'_{[n)}|$ holds while $k{=}n$. When $k{=}n+1$, since $|\mathbf{C}_{[n+\beta')}| = |\mathbf{C}'_{[n+\beta')}|$ and $|\mathbf{D}_{[n+\beta')}| = |\mathbf{D}'_{[n+\beta')}|$, we have

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

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

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

3.2.3 Strategy Space.

Definition 12 (Strategy Space). *Given a lattice basis $\mathbf{B}$ of a 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 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 (Section 2.4) with hk3-sieve [HKL18] 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.

3.3 Adapting Strategy Selection for Different Lattice Attacks

In addition to primal attacks, several other lattice-based attack algorithms also rely on BKZ preprocessing, including dual attacks [CMHST25], BDD attacks [KKNM25], and approximate CVP attacks [EK20]. In these scenarios, the search step in the strategy space can be substituted with alternative algorithms. The key challenge lies in balancing the time cost of BKZ preprocessing with the time complexity of the subsequent algorithms. Our strategy selection approach is designed to be highly adaptable, allowing for the optimization of the overall attack pipeline by selecting the most efficient strategy based on this balance.

4 Strategy Search Algorithm

Section 4.1 introduces the components and construction of the strategy search algorithm. The refined SVP-dimension estimator used by this framework is presented in Section 6.2, with algorithmic details in Appendix A.1.

The primary benefit of the strategy search algorithm (Algorithm 2) lies in its generality. It functions as a drop-in component for any lattice-based attack that employs lattice basis preprocessing (including primal, dual, and hybrid attacks), permitting both minimal time cost and a precise quantification of the speed-up achieved by strategy optimization.

Finally, we introduce a fast, pruning-based implementation of the algorithm and provide its proof of correctness in Section 4.2.

4.1 Strategy Search.

Given a reduction simulator, an SVP-dimension estimator, and a time-cost model, strategy search compares the resulting two-step strategies. Section 4.1.1 describes the construction and correctness of the search algorithm. The concrete success-probability model and refined dimension estimator are presented in Section 6.

4.1.1 Construction of the Strategy Search Algorithm.

We formalize Q7 in Kyber Round-3 specification [ABD+21] as the following optimization problem.

Definition 13 (The Problem of Searching a Refined Strategy for Solving LWE). *Given an LWE instance* $(\mathbf{A}, \mathbf{b})$*, let* rr_0 *be the initial log-GS profile of its primal uSVP lattice basis. Fix a reduction algorithm* $\mathcal{R}$*, an SVP oracle* $\mathcal{C}$*, a reduction simulator* `ReductionSim`*, a dimension estimator* `SVPDimEst`*, a time-cost model* $\mathcal{T}$*, and a nonempty finite set* Ξ *of allowed parameter lists for one reduction call. Let* $\mathcal{S}_{\mathrm{est}} \subseteq \mathcal{S}(\mathcal{R}, \mathcal{C})$ *consist of the admissible complete strategies whose reduction calls use parameters in* Ξ *and whose terminal dimension is the feasible output*

$$d_{\mathrm{svp}} = \texttt{SVPDimEst}\big(\texttt{ReductionSim}(\mathsf{rr}_0, \mathsf{S}), \sigma\big).$$

The objective is to minimize $T_{\mathrm{total}}(\mathsf{S}, d_{\mathrm{svp}})$ *over* $\mathcal{S}_{\mathrm{est}}$ *under these fixed components.*

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 $(\mathsf{S}, d_{\mathrm{svp}}) \in \mathcal{S}(\mathcal{R}, \mathcal{C})$;
output: The unique shortest vector $\mathbf{t}$;

Function `TwoStepSolver`$((\mathbf{A}, \mathbf{b}), \mathcal{R}, \mathcal{C}, (\mathsf{S}, d_{\mathrm{svp}}))$**:**
  Construct lattice basis $\mathbf{B}$ using primal attack to solve LWE instance $(\mathbf{A}, \mathbf{b})$;
  $\mathbf{B} = \mathtt{LLL}(\mathbf{B})$;
  for $\xi \in \mathsf{S}$ **do**
    $\mathbf{B} \leftarrow \mathcal{R}(\mathbf{B}, \xi)$;
  if $d_{\mathrm{svp}} > 0$ **then**
    $\pi_{d-d_{\mathrm{svp}}}(\mathbf{t}) \leftarrow$ run an SVP oracle $\mathcal{C}$ on $\mathbf{B}_{\pi[d-d_{\mathrm{svp}}:d)}$;
    $\mathbf{t} \in \mathcal{L} \leftarrow$ Lift $\pi_{d-d_{\mathrm{svp}}}(\mathbf{t})$ by Babai's Nearest Plane Algorithm [Bab86];
  else
    $\mathbf{t} \leftarrow$ the first vector of $\mathbf{B}$;
  return $\mathbf{t}$;

Algorithm 1: Two-step Solver

Let TwoStepSolver (Algorithm 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 $(\mathsf{S}, d_{\mathrm{svp}})$. This approach first applies a series of reductions and, at the appropriate point, calls an SVP oracle to search for the target vector.

We propose *Strategy Search* (SSearch) to identify a minimum-cost strategy $(\mathsf{S}_{\min}, d_{\mathrm{svp}}^{(\min)})$ in $\mathcal{S}_{\mathrm{est}}$.

The computational infeasibility of exhaustively evaluating all possible strategies arises from the exponential scaling of the reduction algorithm's time cost

with respect to ξ. As a solution, we introduce two key components for accurately estimating the reduction effect of any given strategy: (1) `ReductionSim`, a polynomial-time simulator of the reduction algorithm $\mathcal{R}$; (2) `SVPDimEst`, a dimension estimation method for the SVP oracle.

`ReductionSim` can be initialized as one of the simulators: the BKZ simulator [CN11] or the PnJBKZ simulator proposed in Section 5, depending on the reduction algorithm $\mathcal{R}$ used, such as BKZ or PnJBKZ. The PnJBKZ simulator in Section 5 is the first polynomial-time simulator for PnJBKZ. Given the initial log-GS profile rr_0 and a reduction strategy S, $\texttt{ReductionSim}(\mathsf{rr}_0, \mathsf{S})$ predicts how the log-GS profile 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 $\texttt{ReductionSim}(\mathsf{rr}_0, \xi)$.

For a given two-step strategy $(\mathsf{S}, d_{\text{svp}})$, assuming the probability of successfully recovering the target vector in the SVP search step is $p_\star = 0.999$, the total time cost of Algorithm 1 is given by:

$$T_{\text{total}}(\mathsf{S}, d_{\text{svp}}) = T_{\mathcal{R}s}(\mathsf{S}) + T_{\mathcal{C}}(d_{\text{svp}}), \tag{3}$$

where $T_{\mathcal{R}s}(\mathsf{S})$ is the cumulative time cost of the reduction steps in S, and $T_{\mathcal{C}}(d_{\text{svp}})$ is the time cost of the SVP oracle. In strategy search, we therefore optimize the deterministic nominal cost $T_{\text{total}}(\mathsf{S}, d_{\text{svp}}) := T_{\mathcal{R}s}(\mathsf{S}) + T_{\mathcal{C}}(d_{\text{svp}})$.

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. Section 2.5) that in the search step, can accurately estimate the dimension for the SVP oracle $\mathcal{C}$ on a lattice basis that has already been reduced by $\mathcal{R}$;*
3. *The time-cost model $\mathcal{T}$ for algorithms $\mathcal{R}$ and $\mathcal{C}$.*

4.1.2 Finite Search under a Cost Bound.

For the cost calculation in Algorithm 2, write

$$T_{\mathcal{R}s}(\mathsf{S}) = \sum_{i=1}^{z} T_{\mathcal{R}}(\xi_i), \qquad \mathsf{S} = [\![\xi_1, \cdots, \xi_z]\!].$$

Here $T_{\mathcal{R}}(\xi)$ is the cost assigned by the fixed model to a call with parameter ξ. The empty reduction sequence is denoted by $\mathsf{S}_\emptyset = [\![\ ]\!]$ and has reduction cost zero.

Theorem 1 (Finite search under a cost bound). *Assume that every $\xi \in \Xi$ has a finite positive reduction cost, and define*

$$T_{\mathcal{R}}^{\text{lb}} := \min_{\xi \in \Xi} T_{\mathcal{R}}(\xi) > 0.$$

Assume also that each feasible terminal search has finite non-negative cost, and that the empty reduction sequence has a feasible terminal dimension

$$d_{\mathrm{svp}}^{(0)} := \mathit{SVPDimEst}(\mathsf{rr}_0, \sigma).$$

Let

$$T_0 := T_{\mathcal{C}}(d_{\mathrm{svp}}^{(0)}), \qquad \mathcal{S}_{\mathrm{est}}(T_0) := \{(\mathsf{S}, d_{\mathrm{svp}}) \in \mathcal{S}_{\mathrm{est}} : T_{\mathcal{R}s}(\mathsf{S}) \le T_0\}.$$

Then $\mathcal{S}_{\mathrm{est}}(T_0)$ is finite and non-empty, and

$$\min_{(\mathsf{S}, d_{\mathrm{svp}}) \in \mathcal{S}_{\mathrm{est}}} T_{\mathrm{total}}(\mathsf{S}, d_{\mathrm{svp}}) = \min_{(\mathsf{S}, d_{\mathrm{svp}}) \in \mathcal{S}_{\mathrm{est}}(T_0)} T_{\mathrm{total}}(\mathsf{S}, d_{\mathrm{svp}}).$$

In particular, a minimum-cost strategy exists in $\mathcal{S}_{\mathrm{est}}$.

Proof. If $(\mathsf{S}, d_{\mathrm{svp}}) \in \mathcal{S}_{\mathrm{est}}(T_0)$ and S contains z calls, then

$$z T_{\mathcal{R}}^{\mathrm{lb}} \le T_{\mathcal{R}s}(\mathsf{S}) \le T_0, \qquad z \le \left\lfloor \frac{T_0}{T_{\mathcal{R}}^{\mathrm{lb}}} \right\rfloor.$$

Since Ξ is finite, there are only finitely many reduction sequences satisfying this length bound. Each has at most one feasible terminal dimension selected by the fixed estimator. Hence $\mathcal{S}_{\mathrm{est}}(T_0)$ is finite. It contains $(\mathsf{S}_\emptyset, d_{\mathrm{svp}}^{(0)})$, whose total cost is T_0, so its minimum is attained and is at most T_0.

For any candidate outside $\mathcal{S}_{\mathrm{est}}(T_0)$, non-negativity of the terminal cost gives

$$T_{\mathrm{total}}(\mathsf{S}, d_{\mathrm{svp}}) = T_{\mathcal{R}s}(\mathsf{S}) + T_{\mathcal{C}}(d_{\mathrm{svp}}) \ge T_{\mathcal{R}s}(\mathsf{S}) > T_0.$$

No such candidate improves on the empty-reduction strategy. The minimum over $\mathcal{S}_{\mathrm{est}}(T_0)$ is therefore also the minimum over $\mathcal{S}_{\mathrm{est}}$. □

Algorithm 2 enumerates admissible reduction sequences over Ξ up to the finite length bound above. It retains those within the reduction-cost budget T_0, estimates their terminal dimensions, and evaluates feasible complete strategies using the given model. The empty-reduction strategy initializes the incumbent. Theorem 1 therefore yields the following result.

Corollary 1. *Under Condition 1 and the assumptions of Theorem 1, Simplified SSearch terminates and outputs a strategy with minimum modeled time cost in $\mathcal{S}_{\mathrm{est}}$.*

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 $\mathcal{T}$, a nonempty finite parameter set Ξ;

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

Function `SimplifiedSSearch`$((\mathbf{A}, \mathbf{b}, \sigma)$, `ReductionSim`, `SVPDimEst`, $\mathcal{T}$, Ξ`):`

- $\mathsf{rr}_0 \leftarrow$ The log-GS profile of a lattice basis $\mathbf{B}$ from LWE instance $(\mathbf{A}, \mathbf{b})$ by Kannan embedding technique (See. 2.5);
- $(T_{\mathcal{R}}, T_{\mathcal{C}}) \leftarrow \mathcal{T}$; $T_{\mathcal{R}}^{\text{lb}} \leftarrow \min_{\xi \in \Xi} T_{\mathcal{R}}(\xi)$; $d_{\text{svp}}^{(0)} \leftarrow$ `SVPDimEst`(rr_0, σ); $T_0 \leftarrow T_{\mathcal{C}}(d_{\text{svp}}^{(0)})$;
- $T_{\min} \leftarrow T_0$; $(\mathsf{S}_{\min}, d_{\text{svp}}^{(\min)}) \leftarrow (\llbracket\,\rrbracket, d_{\text{svp}}^{(0)})$;
- **for** *each admissible sequence* S *over* Ξ *with* $1 \le |\mathsf{S}| \le \lfloor T_0 / T_{\mathcal{R}}^{\text{lb}} \rfloor$ **do**
  - **if** $T_{\mathcal{R}s}(\mathsf{S}) > T_0$ **then**
    - continue;
  - $\mathsf{rr} \leftarrow$`ReductionSim`$(\mathsf{rr}_0, \mathsf{S})$; $d_{\text{svp}} \leftarrow$ `SVPDimEst`(rr, σ); **if** *the estimator returns no feasible terminal dimension* **then**
    - continue;
  - $T_{\text{total}}(\mathsf{S}, d_{\text{svp}}) \leftarrow T_{\mathcal{R}s}(\mathsf{S}) + T_{\mathcal{C}}(d_{\text{svp}})$;
  - **if** $T_{\text{total}} < T_{\min}$ **then**
    - $(\mathsf{S}_{\min}, d_{\text{svp}}^{(\min)}) \leftarrow (\mathsf{S}, d_{\text{svp}})$; $T_{\min} \leftarrow T_{\text{total}}$;
- **return** $(\mathsf{S}_{\min}, d_{\text{svp}}^{(\min)})$;

Algorithm 2: Simplified SSearch

When TwoStepSolver employs the minimum time-cost two-step strategy $(\mathsf{S}_{\min}, d_{\text{svp}}^{(\min)})$ determined by the SSearch algorithm, we name this algorithm MinTwoStepSolver, we demonstrate its high efficiency in solving LWE in Section 8.

4.2 Pruning SSearch

While the Simplified SSearch algorithm is guaranteed to find the minimum time-cost strategy in principle, the vastness of the strategy space renders a naive search infeasible. To overcome this, we propose Pruning SSearch (PSSearch, Algorithm 3), which efficiently explores the space by admissibly pruning partial strategies that cannot improve upon the current best objective, thus guaranteeing optimality. The correctness of PSSearch is formally proven in Section 4.2.2. The core of our approach separates the enumeration of reduction strategies into a dedicated subalgorithm (Algorithm 4).

Critically, same as Strategy Search Algorithm, Algorithm 4 is applicable to any lattice-based attack (primal, dual, or hybrid) that employs lattice-basis preprocessing, functioning as a drop-in component to minimize reduction time and quantify the achieved speed-up.

4.2.1 Construction of the Pruning SSearch 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 14 (Strategy Tree $\mathcal{W}$). *A Strategy Tree $\mathcal{W}$ consists of*

1. *The root node of $\mathcal{W}$ is* $\mathsf{root} = [\![\ ]\!]$.

2. *The i-th level node is in the form $\mathsf{S}_i = [\![\xi_1, ..., \xi_i]\!]$, which consists of a reduction strategy with length i.*

3. *For any i-th level strategy S_i, its child node is $\mathsf{S}_{i+1} = \mathsf{S}_i \| [\![\xi_{i+1}]\!]$, where $\xi_{i+1} \in \Xi$ can further improve the basis quality and $T_{\mathcal{R}s}(\mathsf{S}_{i+1}) \leq T_0$.*

Now, we describe the procedure of the PSSearch algorithm. Figure 5 presents an example of a strategy tree and the pruning process for each strategy node.

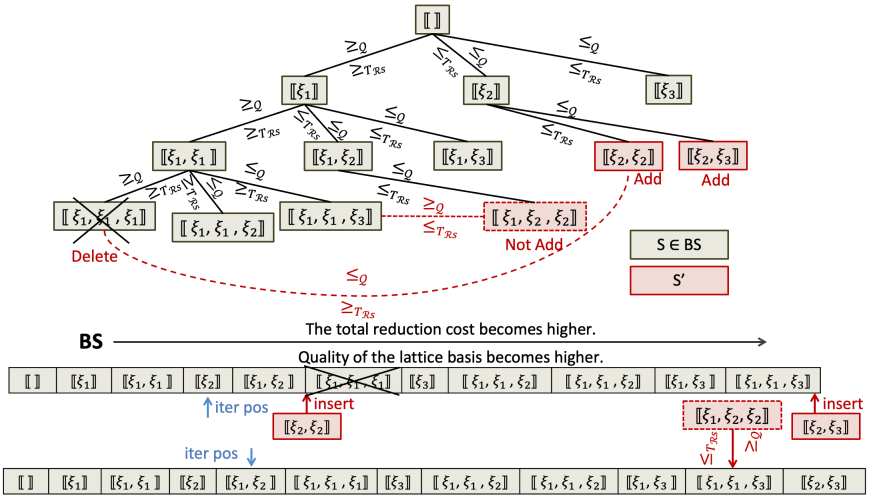


Fig. 5: Illustrative Example of PSSearch. "$\geq_{T_{\mathcal{R}s}}$" indicates that the reduction time cost of the strategy on the left side is greater than that of the strategy on the right side, i.e., $T_{\mathcal{R}s}(\mathsf{S}_{\text{left}}) \geq T_{\mathcal{R}s}(\mathsf{S}_{\text{right}})$. Each 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 reduction strategy S. Let rr_0 be log-GS profile of the input lattice basis $\mathbf{B}$. We start by adding a strategy $\mathsf{S}_\emptyset = [\![\ ]\!]$ with no reduction to the list BS, corresponding to the root node of $\mathcal{W}$, denoted as root. Let $d_{\text{svp}}^{(0)} = \texttt{SVPDimEst}(\mathsf{rr}_0, \sigma)$, then the time cost of this strategy is as follows: $T_{\text{total}}(\mathsf{S}_\emptyset, d_{\text{svp}}^{(0)}) = T_{\mathcal{R}s}(\mathsf{S}_\emptyset) + T_{\mathcal{C}}(d_{\text{svp}}^{(0)}) = 0 + T_{\mathcal{C}}(d_{\text{svp}}^{(0)}) = T_{\mathcal{C}}(d_{\text{svp}}^{(0)})$.

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. $[\![\xi]\!]$ 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 $T_{\mathcal{C}}(d_{\mathrm{svp}})$ decreases with $d_{\mathrm{svp}} = \mathtt{SVPDimEst}(\mathtt{ReductionSim}(\mathsf{rr}_0, \mathsf{S}), \sigma)$, 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 S' awaiting addition, we compare it against each current strategy $\mathsf{S} \in \mathsf{BS}$ 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 blocksizes 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 $\mathsf{S} \in \mathsf{BS}$ that has no greater reduction time cost and no worse basis quality than S', i.e. $T_{\mathcal{R}s}(\mathsf{S}) \leq T_{\mathcal{R}s}(\mathsf{S}')$ and $\mathtt{ReductionSim}(\mathsf{rr}_0, \mathsf{S}) \geq_{\mathcal{Q}} \mathtt{ReductionSim}(\mathsf{rr}_0, \mathsf{S}')$, then don't add S'.*

2. *Otherwise, add S' and remove all other strategies in BS that have no smaller reduction time cost and no better basis quality than S'.*

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 $\mathcal{T}$, a nonempty finite parameter set Ξ;
output: $(\mathsf{S}_{\min}, d_{\mathrm{svp}}^{(\min)})$;
1 **Function** `PSSearch`$((\mathbf{A}, \mathbf{b}, \sigma)$, `ReductionSim`, `SVPDimEst`, $\mathcal{T}$, Ξ`):`
2 $\mathsf{rr}_0 \leftarrow$ The log-GS profile of a lattice basis $\mathbf{B}$ from LWE instance $(\mathbf{A}, \mathbf{b})$ by Kannan embedding technique (See. 2.5);
3 $(T_{\mathcal{R}s}, T_{\mathcal{C}}) \leftarrow \mathcal{T}$; $d_{\mathrm{svp}}^{(0)} \leftarrow \mathtt{SVPDimEst}(\mathsf{rr}_0, \sigma)$; $T_0 \leftarrow T_{\mathcal{C}}(d_{\mathrm{svp}}^{(0)})$;
4 $\mathsf{BS} \leftarrow \mathtt{PRSSearch}(\mathsf{rr}_0$, `ReductionSim`, $T_{\mathcal{R}s}$, Ξ, T_0, $-\infty)$;
5 Find the minimum-cost complete strategy among $\mathsf{S} \in \mathsf{BS}$ for which $d_{\mathrm{svp}} = \mathtt{SVPDimEst}(\mathtt{ReductionSim}(\mathsf{rr}_0, \mathsf{S}), \sigma)$ is feasible, and denote it by $(\mathsf{S}_{\min}, d_{\mathrm{svp}}^{(\min)})$;
6 **return** $(\mathsf{S}_{\min}, d_{\mathrm{svp}}^{(\min)})$;

Algorithm 3: Pruning SSearch

input : The log-GS profile rr_0, a simulator `ReductionSim` for $\mathcal{R}$, a reduction time-cost model $T_{\mathcal{R}s}$, a nonempty finite parameter set Ξ, a reduction-cost budget T_0, the lower bound of the basis quality Q_{bound} ($Q_{\text{bound}} = -\infty$ means no additional quality lower bound);
output: BS;

Function `PRSSearch`(rr_0, `ReductionSim`, $T_{\mathcal{R}s}$, Ξ, T_0, Q_{bound})**:**
  $\mathsf{BS} \leftarrow [\![\,]\!]$; $k \leftarrow 0$;
  while $k < \sharp\mathsf{BS}$ **do**
    $\mathsf{S}_1 \leftarrow \mathsf{BS}[k]$;
    for $\xi' \in \Xi$ **do**
      $\mathsf{S}' \leftarrow \mathsf{S}_1 \| [\![\xi']\!]$; **if** $T_{\mathcal{R}s}(\mathsf{S}') > T_0$ **then**
        continue;
      $\mathsf{rr}' \leftarrow$ `ReductionSim`$(\mathsf{rr}_0, \mathsf{S}')$; **if** $\mathsf{rr}' \not\succeq_{\mathcal{Q}}$ `ReductionSim`$(\mathsf{rr}_0, \mathsf{S}_1)$ **then**
        continue;
      if $Q_{\text{bound}} \neq -\infty$ **then**
        if $\mathsf{rr}' \not\succeq_{\mathcal{Q}} Q_{\text{bound}}$ **then**
          continue;
      if $\exists\ \mathsf{S} \in \mathsf{BS}$ s.t. `ReductionSim`$(\mathsf{rr}_0, \mathsf{S}') \leq_{\mathcal{Q}}$ `ReductionSim`$(\mathsf{rr}_0, \mathsf{S})$ **and** $T_{\mathcal{R}s}(\mathsf{S}') \geq T_{\mathcal{R}s}(\mathsf{S})$ **then**
        continue;
      else
        $\mathsf{BS} \leftarrow \mathsf{BS} \| [\![\mathsf{S}']\!]$;
        for $\forall \mathsf{S} \in \mathsf{BS}$ s.t. $\mathsf{S} \neq \mathsf{S}'$ **and** `ReductionSim`$(\mathsf{rr}_0, \mathsf{S}') \geq_{\mathcal{Q}}$ `ReductionSim`$(\mathsf{rr}_0, \mathsf{S})$ **and** $T_{\mathcal{R}s}(\mathsf{S}') \leq T_{\mathcal{R}s}(\mathsf{S})$ **do**
          $j \leftarrow$ the current index of S in BS; **if** $j \leq k$ **then**
            $k \leftarrow k - 1$;
          Remove S from BS;
    $k \leftarrow k + 1$;
  return BS;

Algorithm 4: Pruning Reduction Strategy Search

4.2.2 Correctness Proof of Pruning SSearch

Theorem 2 establishes that PSSearch terminates and preserves a minimum-cost strategy in the candidate space $\mathcal{S}_{\text{est}}$ defined in Section 4.1.1.

Theorem 2. *Assume Condition 1 and the assumptions of Theorem 1. Suppose that the fixed* ***`ReductionSim`*** *preserves lattice volume and satisfies both parts of Property 1 for every $\xi \in \Xi$. Suppose also that, whenever* ***`SVPDimEst`*** *returns a feasible dimension for a profile, it returns a feasible dimension no larger for any profile of the same lattice volume that is at least as good under $\succeq_{\mathcal{Q}}$. If $T_{\mathcal{C}}$ is nondecreasing in the dimension, then PSSearch (Algorithm 3) terminates and returns a strategy minimizing T_{total} over $\mathcal{S}_{\text{est}}$ under the fixed simulator, estimator, and cost model.*

The following lemma verifies the dimension-estimator condition for the estimator in Section 2.5 under the Gaussian Heuristic and the quality-preservation property of Generic Reduction.

Lemma 1. *Suppose the Gaussian Heuristic holds, and let* `SVPDimEst` *be the estimator described in Section 2.5. Let* $\mathbf{C}$ *and* $\mathbf{D}$ *be bases of the same* d*-dimensional lattice obtained from an LWE instance with standard deviation* σ*, with log-GS profiles satisfying* $\mathsf{rr}_{\mathbf{D}} \geq_{\mathcal{Q}} \mathsf{rr}_{\mathbf{C}}$*. Apply the same reduction tour* $\mathcal{R}$*-*ξ*, satisfying Property 1, to obtain* $\mathbf{C}'$ *and* $\mathbf{D}'$*. If* `SVPDimEst` *returns a feasible dimension for* $\mathsf{rr}_{\mathbf{C}'}$*, then it also returns a feasible dimension for* $\mathsf{rr}_{\mathbf{D}'}$*, and*

$$\texttt{SVPDimEst}(\mathsf{rr}_{\mathbf{D}'}, \sigma) \leq \texttt{SVPDimEst}(\mathsf{rr}_{\mathbf{C}'}, \sigma).$$

Proof. Property 1 gives $\mathsf{rr}_{\mathbf{D}'} \geq_{\mathcal{Q}} \mathsf{rr}_{\mathbf{C}'}$. Since the total lattice volumes are equal, the prefix-volume inequalities imply

$$\mathrm{GH}(\mathbf{C}'_{\pi[d-k:d)}) \leq \mathrm{GH}(\mathbf{D}'_{\pi[d-k:d)}), \qquad 1 \leq k \leq d.$$

Thus, every candidate dimension k satisfying $\sigma\sqrt{k} \leq \mathrm{GH}(\mathbf{C}'_{\pi[d-k:d)})$ also satisfies $\sigma\sqrt{k} \leq \mathrm{GH}(\mathbf{D}'_{\pi[d-k:d)})$. Taking the smallest feasible dimension for each profile proves the claim. The same argument applies directly to $\mathbf{C}$ and $\mathbf{D}$ before reduction. □

Proof (Proof of Theorem 2). By Theorem 1, the finite set $\mathcal{S}_{\mathrm{est}}(T_0)$ contains a minimum-cost strategy in $\mathcal{S}_{\mathrm{est}}$. Moreover, every reduction sequence within the budget T_0 has length at most $\lfloor T_0/T_{\mathcal{R}}^{\mathrm{lb}} \rfloor$. PRSSearch expands each retained node once and tests each parameter in the finite set Ξ, so PSSearch terminates.

Let $(\mathsf{S} = [\![\xi_1, \cdots, \xi_z]\!], d_{\mathrm{svp}}) \in \mathcal{S}_{\mathrm{est}}(T_0)$ be a minimum-cost strategy for the given initial profile rr_0. For $1 \leq i \leq z$, write $\mathsf{S}_i = [\![\xi_1, \cdots, \xi_i]\!]$ for its prefix of length i. The empty prefix is $\mathsf{S}_\emptyset$. If S belongs to the final list BS, PSSearch returns a strategy with the same minimum cost. Otherwise, we show that pruning a prefix still preserves the minimum cost.

If a prefix S_i is rejected or removed by the dominance rule, there is another prefix S' such that

$$\texttt{ReductionSim}(\mathsf{rr}_0, \mathsf{S}') \geq_{\mathcal{Q}} \texttt{ReductionSim}(\mathsf{rr}_0, \mathsf{S}_i),$$
$$T_{\mathcal{R}s}(\mathsf{S}') \leq T_{\mathcal{R}s}(\mathsf{S}_i).$$

Consider the complete reduction sequence $\mathsf{S}^* := \mathsf{S}' \parallel [\![\xi_{i+1}, \cdots, \xi_z]\!]$, where $\parallel$ denotes ordered-list concatenation. Additivity of the reduction costs gives

$$\begin{aligned} T_{\mathcal{R}s}(\mathsf{S}^*) &= T_{\mathcal{R}s}(\mathsf{S}') + \sum_{j=i+1}^{z} T_{\mathcal{R}}(\xi_j) \\ &\leq T_{\mathcal{R}s}(\mathsf{S}_i) + \sum_{j=i+1}^{z} T_{\mathcal{R}}(\xi_j) = T_{\mathcal{R}s}(\mathsf{S}) \leq T_0. \end{aligned}$$

Applying the order-preservation property to the common suffix shows that

$$\texttt{ReductionSim}(\mathsf{rr}_0, \mathsf{S}^*) \geq_{\mathcal{Q}} \texttt{ReductionSim}(\mathsf{rr}_0, \mathsf{S}).$$

Every suffix call is nonworsening by the first part of Property 1. Omit any suffix calls that leave the simulated profile unchanged, retaining the notation S^* for the resulting sequence. This preserves the later simulated profiles and can only decrease the reduction cost, so S^* is admissible and remains within the budget. By the assumed property of `SVPDimEst`, its terminal dimension

$$d^*_{\mathrm{svp}} := \texttt{SVPDimEst}\big(\texttt{ReductionSim}(\mathsf{rr}_0, \mathsf{S}^*), \sigma\big)$$

is feasible and satisfies $d^*_{\mathrm{svp}} \leq d_{\mathrm{svp}}$. Therefore,

$$\begin{aligned} T_{\mathrm{total}}(\mathsf{S}^*, d^*_{\mathrm{svp}}) &= T_{\mathcal{R}s}(\mathsf{S}^*) + T_{\mathcal{C}}(d^*_{\mathrm{svp}}) \\ &\leq T_{\mathcal{R}s}(\mathsf{S}) + T_{\mathcal{C}}(d_{\mathrm{svp}}) = T_{\mathrm{total}}(\mathsf{S}, d_{\mathrm{svp}}). \end{aligned}$$

The replacement therefore preserves optimality, including when equality holds.

Because the search is finite and dominance is transitive, every prefix rejected or removed by the dominance rule has a dominating representative in the final list BS. Every retained prefix is expanded. Induction along $\xi_1, \cdots, \xi_z$, using the replacement argument above, therefore yields a final candidate of no greater reduction cost and no worse quality than S. This candidate is also optimal, and PSSearch returns the minimum cost among the feasible candidates in BS. □

5 The Design of PnJBKZ simulator

PnJBKZ [ADH+19], when combined with hk3-sieve, is considered the most efficient lattice reduction algorithm in practice. Prior to this work, PnJBKZ lacked a polynomial-time simulator capable of predicting reductions with nontrivial jump parameters $J > 1$. 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 $\mathsf{Jump} > 1$. Only then will we be able to generate optimized reduction strategies for PnJBKZ using PSSearch.

The rigorous dynamic analysis proposed in [LN20, Wan24] 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 [GN08, CN11, AWHT16, ADH+19].

Meanwhile, the classical BKZ simulator is unable to predict the performance of PnJBKZ when $\mathsf{Jump} > 1$. Figure 6 shows that a PnJBKZ reduction strategy with $\mathsf{Jump} > 1$ can significantly improve the efficiency of reduction. Specifically, the progressive reduction strategy with $\mathsf{Jump} = 9$ is four times faster than the strategy with $\mathsf{Jump} = 1$ in achieving the same slope of -0.019. To optimize

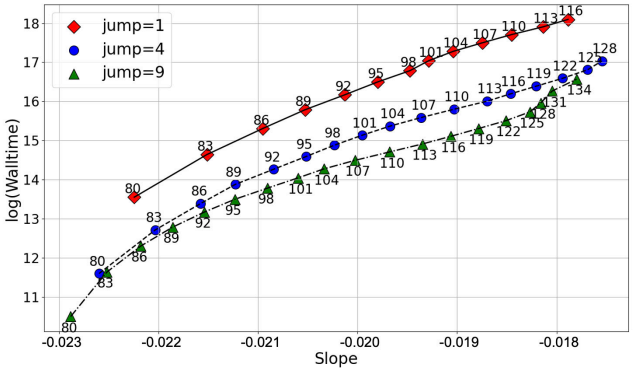


Fig. 6: Speedup in Reduction Efficiency via the Jump Strategy.

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.

The test in Figure 6 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, the blocksize of PnJBKZ.

5.1 Construction of the PnJBKZ simulator.

Before presenting the detailed construction of the PnJBKZ simulator, we first briefly review the main idea behind the BKZ 2.0 simulator proposed in [CN11], 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 $\mathrm{Sim}(l_0') = \ln\left(\mathrm{GH}(\mathcal{L}(\mathbf{B}^{(0)}_{\pi[\beta)})\right) \approx \frac{1}{2}\ln(\beta/2\pi e) + \frac{1}{\beta}\ln\left(\mathrm{Vol}(\mathcal{L}(\mathbf{B}^{(0)}_{\pi[\beta)}))\right)$ under the GH. Then by GH and the previously calculated value of $\mathrm{Sim}(l_0')$, it calculates $\mathrm{Sim}(l_1') := \ln\left(\mathrm{GH}(\mathcal{L}(\mathbf{B}^{(1)}_{\pi[1:\beta+1)}))\right) \approx \frac{1}{2}\ln(\beta/2\pi e) + \frac{1}{\beta}\left(\ln \mathrm{Vol}(\mathcal{L}(\mathbf{B}^{(0)}_{\pi[\beta+1)})) - \mathrm{Sim}($

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 $\mathcal{L}(\mathbf{B}^{(0)}_{\pi[\beta)})$ does not affect the volume of the projected sublattice basis $\mathcal{L}(\mathbf{B}^{(0)}_{\pi[\beta+1)})$. Therefore, we have $\mathrm{Vol}(\mathcal{L}(\mathbf{B}^{(1)}_{\pi[\beta+1)})) = \mathrm{Vol}(\mathcal{L}(\mathbf{B}^{(0)}_{\pi[\beta+1)}))$.

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 approxi-

mate 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 block-size 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 [CN11] and its variants cannot simulate the behavior of PnJBKZ when $\mathsf{Jump} > 1$. Let $\mathsf{Jump} = J$, and let $\mathbf{B}^{(i)}$ denote the intermediate basis after positions $0, \cdots, i-1$ have been processed, with $\mathbf{B}^{(0)}$ denoting the initial 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 $\mathrm{Pump}'(\mathbf{B}_{\pi[i:i+\beta)}, i, \beta, f)$, 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 $i \in [d)$ and k is defined in Eq. (4).

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, $\forall j \in [i)$, then $\ln\left(\mathrm{Vol}\left(\mathcal{L}(\mathbf{B}^{(i)}_{\pi[i:k)})\right)\right) := \ln\left(\mathrm{Vol}\left(\mathcal{L}(\mathbf{B}^{(0)}_{\pi[0:k)})\right)\right) - \ln\left(\mathrm{Vol}\left(\mathcal{L}(\mathbf{B}^{(i)}_{\pi[0:i)})\right)\right) = \sum_{j=0}^{k-1} l_j - \sum_{j=0}^{i-1}\mathrm{Sim}(l''_j)$, where $i \in [d)$ and k is defined below. Under GH, for $i \in [d)$, we can iteratively compute $\mathrm{Sim}(l''_i) :=$

$$\frac{1}{2}\ln\left(\frac{k-i}{2\pi e}\right) + \frac{1}{k-i}\left(\sum_{j=0}^{k-1} l_j - \sum_{j=0}^{i-1}\mathrm{Sim}(l''_j)\right),\ k = \begin{cases} i-(i \bmod J)+\beta, & i \in [d-\beta), \\ d, & i \in [d-\beta : d). \end{cases} \tag{4}$$

In other words, we only need to input the initial logarithmic 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. (4), which describes the change of the logarithmic GS norms for a lattice basis after each tour of the PnJBKZ(β, J) reduction. Here, l''_i are the actual logarithmic GS norms after a

tour of PnJBKZ(β, J) reduction. A detailed description of the PnJBKZ simulator is provided in Algorithm 5.

The tail constants $r_0, \cdots, r_{44}$ defined in Algorithm 5 are precomputed and reused, following the BKZ 2.0 simulator [CN11]. The dimension-dependent constants c_i are initialized for every $1 \leq i \leq \beta$, since a large jump can produce local blocks of dimension below 45 even when $\beta \geq 45$.

input : The log-GS profile $\mathsf{rr} = (l_0, \cdots, l_{d-1})$, blocksize $\beta \in \{45, \cdots, d\}$, Jump J and number of tours t.
output: A prediction for the log-GS profile $l_i'' = \ln(\|\mathbf{b}_i''^*\|)$ 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 $i \leftarrow 1$ **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
        $l_k'' \leftarrow \ln(\mathrm{V})/\beta' + c_{\beta'}$;
    $\ln(\mathrm{V}) \leftarrow \sum_{i=0}^{d-1} l_i - \sum_{i=0}^{d-46} l_i''$;
    for $k \leftarrow d-45$ **to** $d-1$ **do**
      $l_k'' \leftarrow \frac{\ln(\mathrm{V})}{45} + r_{k+45-d}$;
    for $k \leftarrow 0$ **to** $d-1$ **do**
      $l_k \leftarrow l_k''$;
  return $(l_0, \cdots, l_{d-1})$;

Algorithm 5: PnJBKZ simulator

When $i \equiv 0 (\mathrm{mod}\ J)$, the index i of 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 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 [CN11] $\mathrm{Vol}(\mathcal{L}'\left(\mathbf{B}^{(i)}_{\pi[i:i+\beta)}\right))=\prod_{j=0}^{i+\beta-1}\|\mathbf{b}^*_j\|/\prod_{j=0}^{i-1}\|\mathbf{b}^{*'}_j\|$ 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. (4). Consequently, when $i \equiv 1 (\mathrm{mod}\ J)$, the calculation of GH $\left(\mathcal{L}\left(\mathbf{B}^{(i)}_{\pi[i:i+\beta)}\right)\right)$ differs between the two simulators.

5.1.1 Consider the impact of LLL reduction in PnJBKZ. During each PnJBKZ-(β, J) tour, LLL reduction is applied after each Pump step. To approximate its effect, we apply an empirical smoothing correction to the log-GS profile predicted by the basic PnJBKZ simulator.

The correction is motivated by the conditions imposed by LLL reduction. The size-reduction condition requires $|\mu_{i,j}| \leq \frac{1}{2}$ for every $0 \leq j < i < d$. The Lovász condition with parameter 3/4 requires

$$\frac{3}{4}\|\mathbf{b}^*_i\|^2 \leq \|\mu_{i+1,i}\mathbf{b}^*_i + \mathbf{b}^*_{i+1}\|^2, \qquad 0 \leq i < d-1.$$

The simulator tracks log-GS norms and does not explicitly simulate the coefficients $\mu_{i,j}$ or verify the full LLL conditions. The empirical correction smooths selected predicted norms according to the jump-to-blocksize ratio J/β and local changes in the log-GS profile.

All verification experiments in this paper use the empirically corrected simulator. The corresponding results are presented in Appendix C.1.

5.1.2 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 HKZ-reduced blocks produced by Pump′ satisfy $\|\mathbf{b}^*_i\| = \lambda_1(\mathcal{L}_{\pi[i:k)})$ for $i \in [d)$, where k is defined in Eq. (4). 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 the logarithm of $\lambda_1\left(\mathcal{L}_{\pi[i:k)}\right)$, 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 C.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 [Duc18] is commonly used to accelerate the sieve in each Pump step of PnJBKZ. During the Pump-up stage, the Pump performs a $(\beta - \texttt{d4f}(\beta))$-dimensional progressive sieve, and can thus perform at most $(\beta - \texttt{d4f}(\beta))$ embeddings during the Pump-down stage. Consequently, the output basis of each β-dimensional Pump will inevitably include $\texttt{d4f}(\beta)$-dimensional GS vectors that do not satisfy the HKZ-reduced basis properties. Thus, the upper bound of Jump value should be smaller than $\beta - \texttt{d4f}(\beta)$. Otherwise, our PnJBKZ simulator also cannot be used due to the information of key vector norms remains unknown.

Finally, when predicting the reduction effect of PnJBKZ used in practice, which uses heuristic optimistic d4f value $\texttt{d4f}_{\text{op}}(\beta)$ for each Pump, we set the upper bound of Jump J to be $\texttt{d4f}_{\text{slope}}(s) = -\ln(4/3)/s$. As demonstrated in Section 5.2, 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 Section 5.2, Appendix C.1.2 and C.1.3.

5.2 Predicting PnJBKZ using the d4f technique

The ideal PnJBKZ simulation discussed in Section 5.1 did not consider the influence of using the dimension-for-free (d4f) technique. In practice, the implementation of PnJBKZ used in [ADH+19] and [DSvW21] considers the d4f technique 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 [ADH+19], the authors use a more optimistic d4f value compared to the theoretical one. Specifically, [Duc18] provides two theoretical d4f estimations for solving β-dimension SVP as $\texttt{d4f}(\beta) = \beta\ln(4/3)/\ln(\beta/2\pi)$ and $\text{d4f}(\beta) = \beta\ln(4/3)/\ln(\beta/2\pi e)$, while in the implementation of G6K [ADH+19], it gives a more relaxed value and we called it "optimistic d4f":

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

After using d4f to accelerate PnJBKZ, each Pump in the algorithm performs progressive sieving only in a $(\beta - \texttt{d4f}(\beta))$-dimensional space during the Pump-up stage. As a result, it can perform at most $(\beta - \texttt{d4f}(\beta))$ embeddings during the Pump-down stage. Consequently, the output basis of each β-dimensional Pump will inevitably include $\texttt{d4f}(\beta)$-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 the d4f

technique, including optimistic `d4f` values, we employ a refined `d4f` value estimation proposed in [WWW23]. Additionally, we modify our simulation strategy to improve the accuracy of the simulator in predicting the behavior of PnJBKZ reductions that utilize the `d4f` technique and optimistic `d4f` settings. In this Section 5.2, along with the subsequent verification experiments, we provide a detailed analysis of these adjustments and their impact on the overall performance. Appendix C.1.2, and C.1.3, all PnJBKZ reduction experiments use the "optimistic `d4f`" settings applied in the G6K implementation [ADH+19].

Wang et al. in [WWW23] 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 [Duc18]. The formula for this pessimistic estimation is given by:

$$\mathrm{GH}\left(\mathcal{L}\right) \leq \mathrm{GH}\left(\mathcal{L}_{\pi[f:d)}\right)\sqrt{4/3} \tag{5}$$

Using the GSA approximation $\rho \approx \delta^{-2}$ and the asymptotic GH formula, the above inequality becomes $2f\ln\delta \leq \ln(4/3) + \ln(1-f/d)$. Based on $\ln(\delta) = \Theta\left(\left(\ln\beta\right)/\beta\right) = \Theta\left(\left(\ln d\right)/d\right)$, [Duc18] gives an asymptotic value of `d4f` under pessimistic condition as:

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

The optimistic condition in [Duc18]:

$$\mathrm{GH}\left(\mathcal{L}\right)\sqrt{\frac{d-f}{d}} \leq \mathrm{GH}\left(\mathcal{L}_{\pi[f:d)}\right)\sqrt{\frac{4}{3}}. \tag{7}$$

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

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

The theoretical derivations above are based on the assumption that the current d-dimensional lattice basis is fully-BKZ-$d/2$ reduced. However, during the 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.(7) and Eq.(8).

Thus, the two different maximum values of `d4f` analyzed in [Duc18] 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.

Under the GSA model, the geometric mean of the GS norms in the projected block satisfies

$$\left(\prod_{i=f}^{d-1}\|\mathbf{b}_i^*\|\right)^{1/(d-f)} = \mathrm{Vol}(\mathcal{L})^{1/d}\rho^{f/2}.$$

Combining the Gaussian Heuristic with $\rho = \delta^{-2d/(d-1)} \approx \delta^{-2}$ from Section 2, we obtain

$$\begin{aligned}\lambda_1(\mathcal{L}_{\pi[f:d)}) &\approx \sqrt{\frac{d-f}{2\pi e}}\,\mathrm{Vol}(\mathcal{L})^{1/d}\rho^{f/2} \\ &= \sqrt{\frac{d-f}{2\pi e}}\,\frac{\mathrm{Vol}(\mathcal{L})^{1/d}}{\delta^{df/(d-1)}} \\ &\approx \sqrt{\frac{d-f}{2\pi e}}\,\frac{\mathrm{Vol}(\mathcal{L})^{1/d}}{\delta^{f}}.\end{aligned} \tag{9}$$

Applying the pessimistic condition in Eq. (5) to the final approximation in Eq. (9), we obtain a conservative `d4f` estimation based on GSA:

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

Applying the optimistic condition in Eq. (7) to the same approximation, we obtain an optimistic `d4f` estimation based on GSA:

$$f \leq \log_\delta \sqrt{\frac{4}{3}} \tag{11}$$

It illustrates that, under the current reduction quality of the lattice basis, the maximum `d4f` value should be $\texttt{d4f}_\delta = \log_\delta \sqrt{4/3} \approx \ln(4/3)/(-\text{slope})$. 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 $\texttt{d4f}_\delta$. If the Jump size is not limited, there may be a situation where $\texttt{d4f}_\delta <$ Jump $<$ the optimistic heuristic `d4f` value. In this case, the first (Jump $- \texttt{d4f}_\delta$) 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 $- \texttt{d4f}_\delta$) 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 $J \leq \texttt{d4f}_{\mathrm{slope}}(s) \ll \beta \leq d$, 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 $\min\{J, d-\kappa\}$ vectors in the 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 $i \in [\kappa : \min\{\kappa+J, d\})$, the expected norms of $\|\mathbf{b}_i^*\| \approx \mathrm{GH}(\mathcal{L}(\mathbf{B}_{\pi[i:\min\{\kappa+\beta,d\})}))$.*

Here $\texttt{d4f}_{\text{slope}}(s)$ is an upper bound of the `d4f` value estimation function based on the quality of the current lattice basis proposed in [WWW23]. $\texttt{d4f}_{\text{slope}}(s) := \log_\delta \sqrt{4/3} \approx \ln(4/3)/(-s)$.

As mentioned above, the default `d4f` function used in the implementation of PnJBKZ (both [ADH+19] and [DSvW21]) 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, [WWW23] calculates the refined `d4f` value estimation as $\texttt{d4f}_{\text{slope}}(s) = \ln(4/3)/(-s)$. In this case $J \leq \texttt{d4f}_{\text{slope}}(s)$, we calculate $\Delta_{\texttt{d4f}}(\beta, s) := \max\{0, \texttt{d4f}_{\text{optimistic}}(\beta) - \texttt{d4f}_{\text{slope}}(s)\}$. We round the non-negative correction to the nearest integer, with half-integer ties rounded upward, and set

$$\beta_{\text{sim}} = \beta - \left\lfloor \Delta_{\texttt{d4f}}(\beta, s) + \frac{1}{2} \right\rfloor .$$

We then replace each blocksize β by β_{sim} as the input of Algorithm 5.

The function $\Delta_{\texttt{d4f}}(\beta, s)$ 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 [ADH+19].

Finally, in Appendix C.1.2 and C.1.3, 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 the Section 3.2 of paper [WWW23]. 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.

Prediction accuracy. To assess the prediction accuracy of `PnJBKZSim`, we use the GS-norm ratio

$$\frac{\text{Real}(\|\mathbf{b}_i^*\|)}{\text{Sim}(\|\mathbf{b}_i^*\|)}, \qquad i \in [d).$$

Here, $\text{Real}(\|\mathbf{b}_i^*\|)$ denotes the observed norm of the ith Gram–Schmidt vector after reduction, and $\text{Sim}(\|\mathbf{b}_i^*\|)$ denotes the corresponding norm predicted by `PnJBKZSim` for the same initial profile and reduction parameters. When experimental results are averaged over repeated trials, $\text{Real}(\|\mathbf{b}_i^*\|)$ denotes the geometric mean of the observed GS norms, obtained by averaging their logarithms and

exponentiating the result. A ratio close to 1 indicates an accurate prediction; a ratio greater than 1 indicates an underestimate of the observed norm, whereas a ratio less than 1 indicates an overestimate. We quantify prediction accuracy by reporting the proportions of these ratios lying within $[0.95, 1.05]$ and $[0.90, 1.10]$. The experimental results are presented in Section 8.1 and Appendix C.1.2.

6 Refined LWE Estimator in Two-step Mode

In this section, we propose two refined LWE estimators in the two-step framework, building upon the PnJBKZ simulator and the Pump estimation method. In Section 6.4, we consider the trivial BKZ strategy in which the block size increases by one at each step. In Section 6.5, we employ PSSearch to identify a strategy that minimizes the time cost for solving the LWE instance.

Before presenting these estimators, we describe their success-probability model and the refined SVP-dimension estimate, followed by the time-cost and memory models.

6.1 Success Probability.

For the primal uSVP instance described in Section 2.5, the reduction-stage and search-stage success probabilities are evaluated using the distribution $\sigma^2\chi_k^2$. The target-projection model is described in Section 6.2. Following the notation of [PV21], let $\mathrm{W}_{\xi}^{(1)}$ denote the event of solving LWE using a generic reduction $\mathcal{R}$-ξ, and let $\mathrm{F}_{\xi}^{(1)} = \neg\mathrm{W}_{\xi}^{(1)}$.

Condition 2 *If the lattice basis is independently randomized each time using a reduction procedure ξ (which can further improve the quality of the basis), then the events $\mathrm{W}_{\xi_i}^{(1)}$ and $\mathrm{F}_{\xi_j}^{(1)}$ are independent for $i \neq j$.*

For the BKZ reduction strategy $\mathsf{S}_{\mathrm{end}} = [\![\beta_0, \cdots, \beta_{\mathrm{end}-1}]\!]$ and Pump as the SVP algorithm, we denote by W the event that the LWE instance is solved either during the execution of Progressive BKZ or by the final high-dimensional Pump in the two-step framework. Specifically, let $\mathrm{W}_{\beta}^{(1)}$ denote the event that LWE is solved using BKZ-β, let $\mathrm{F}_{\beta}^{(1)} = \neg\mathrm{W}_{\beta}^{(1)}$ be its complement, and let $\mathrm{W}_{(d_{\mathrm{svp}})}^{(2)}$ denote the event that a Pump algorithm in dimension d_{svp} successfully solves LWE. Here $\Pr[\mathrm{W}_{\beta}^{(1)}] = \Pr\left[x \leftarrow \sigma^2\chi_{\beta}^2 \middle| x \leq \exp(l_{d-\beta})^2\right]$, and $l_{d-\beta}$ is the log-GS profile of the first vector of projective sub-lattice $\mathcal{L}_{\pi[d-\beta:d)}$ of current lattice basis which has been reduced by Progressive BKZ with reduction strategy $\mathsf{S}_{\mathrm{end}}$. In Two-step mode we partition W as:

$$\begin{aligned}\Pr[\mathrm{W}] =& \Pr[\mathrm{W}^{(1)}_{\beta_0}] + \Pr[\mathrm{W}^{(1)}_{\beta_1} \wedge \mathrm{F}^{(1)}_{\beta_0}] + \Pr[\mathrm{W}^{(1)}_{\beta_2} \wedge \mathrm{F}^{(1)}_{\beta_1} \wedge \mathrm{F}^{(1)}_{\beta_0}] \\ &+ \cdots + \Pr\left[\mathrm{W}^{(1)}_{\beta_{\mathrm{end}-1}} \wedge \bigwedge_{j=0}^{\mathrm{end}-2} \mathrm{F}^{(1)}_{\beta_j}\right] + \Pr\left[\mathrm{W}^{(2)}_{d_{\mathrm{svp}}} \wedge \bigwedge_{j=0}^{\mathrm{end}-1} \mathrm{F}^{(1)}_{\beta_j}\right] \\ =& \sum_{i=0}^{\mathrm{end}-1} \Pr\left[\mathrm{W}^{(1)}_{\beta_i} \wedge \bigwedge_{j=0}^{i-1} \mathrm{F}^{(1)}_{\beta_j}\right] + \Pr\left[\mathrm{W}^{(2)}_{d_{\mathrm{svp}}}\right] \cdot \Pr\left[\bigwedge_{j=0}^{\mathrm{end}-1} \mathrm{F}^{(1)}_{\beta_j}\right].\end{aligned} \tag{12}$$

Here $\mathrm{W}^{(2)}_{d_{\mathrm{svp}}}$ means during the process of the final sieve, d_{svp}-dimension progressive sieving finds the projection vector of the target vector and $\mathrm{F}^{(2)}_{d_{\mathrm{svp}}} = \neg\mathrm{W}^{(2)}_{d_{\mathrm{svp}}}$. Event $\mathrm{W}^{(2)}_{d_{\mathrm{svp}}}$ happened means all BKZ-β in the reduction step fail to find the target vector, other else it will not call the final high-dimension sieve. So event $\mathrm{W}^{(2)}_{d_{\mathrm{svp}}}$ is independent with all events $\mathrm{F}^{(1)}_{\beta_j}$. When evaluating the concrete hardness of an LWE instance, the value of d_{svp} will be set to solve this LWE with a probability above 0.999. Here, end denotes the number of reduction calls, so the last call has index $\mathrm{end}-1$, and d_{start} is the dimension of the initial projection sub-lattice in the final sieve.

It is worth noticing that leaky-LWE-Estimator is based on a heuristic assumption that events $\mathrm{W}^{(1)}_{\beta_i}$ and $\mathrm{F}^{(1)}_{\beta_j}$ for $i \neq j$ are independent. See the discussion in Section 4.1 of [PV21] or the implementation of leaky-LWE-Estimator [3] for more details. The heuristic assumption that events $\mathrm{W}^{(1)}_{\beta_i}$ and $\mathrm{F}^{(1)}_{\beta_j}$ for $i \neq j$ are independent which leaky-LWE-Estimator based on, is reasonable to some extent if we assume that the lattice basis will be re-randomized each time it is reduced by a stronger BKZ reduction. Below we reformulate this assumption formally (The concretization of Condition 2):

Heuristic 3 *The lattice basis is randomized each time by a reduction of BKZ-β with larger β. Then events $\mathrm{W}^{(1)}_{\beta_i}$ and $\mathrm{F}^{(1)}_{\beta_j}$ for $i \neq j$ are independent.*

Besides, let $\mathrm{E}^{(1)}_{\beta_i}$, for $i \in \{0, \cdots, \mathrm{end}-1\}$, denote the event that LWE is solved during Progressive BKZ with calls BKZ-$\beta_0, \cdots$, BKZ-β_i. For a generic reduction prefix $\mathsf{S}_{i+1} = [\![\xi_0, \cdots, \xi_i]\!]$, the corresponding event is denoted by $\mathrm{E}^{(1)}_{\mathsf{S}_{i+1}}$. The empty prefix $\mathsf{S}_\emptyset$ has cumulative success probability $\Pr[\mathrm{E}^{(1)}_{\mathsf{S}_\emptyset}] = 0$. For $i = 0$, the preceding cumulative probability $\Pr[\mathrm{E}^{(1)}_{\beta_{i-1}}]$ in the formulas below is interpreted as this empty-prefix probability. Empty conjunctions are understood as the certain event. Based on Heuristic 3, we have $\Pr\left[\mathrm{W}^{(1)}_{\beta_i} \wedge \bigwedge_{j=0}^{i-1} \mathrm{F}^{(1)}_{\beta_j}\right] = \Pr\left[\mathrm{W}^{(1)}_{\beta_i}\right] \cdot \Pr\left[\bigwedge_{j=0}^{i-1} \mathrm{F}^{(1)}_{\beta_j}\right]$ and

$$\Pr[\mathrm{E}^{(1)}_{\beta_i}] = \Pr[\mathrm{E}^{(1)}_{\beta_{i-1}}] + \Pr[\mathrm{W}^{(1)}_{\beta_i}] \cdot \left(1 - \Pr[\mathrm{E}^{(1)}_{\beta_{i-1}}]\right). \tag{13}$$

[3] https://github.com/lducas/leaky-LWE-Estimator

We will use Eq. (13) to calculate the cumulative probability of solving LWE during reduction step, see line 10 of Algorithm 6.

However, the same method cannot be used to calculate the probability of solving LWE during the final sieve. Specifically, we use a progressive sieve algorithm as the final sieve, thus we also need to calculate the probability of solving LWE during each step of the progressive sieve. Specifically, we use $\mathrm{W}_i^{(2)}, \mathrm{F}_i^{(2)}$ as the success rate and failing rate that LWE can be solved using a i-dimensional progressive sieve (which performs sieving on projected sub-lattices with dimensions from 2 to i). Unlike in a progressive BKZ, the lattice basis will not change during sieving. Therefore, the similar heuristic assumption that events $\mathrm{W}_i^{(2)}$ and $\mathrm{F}_j^{(2)}$ for $i \neq j$ are independent cannot be established.

On the contrary, instead of considering that events $\mathrm{W}_{(i)}^{(2)}$ and $\mathrm{F}_{(j)}^{(2)}$ for $i \neq j$ are independent, we consider that there is an inclusive relationship between $\mathrm{W}_{(i)}^{(2)}$ and $\mathrm{W}_{(j)}^{(2)}$ for $j \leq i$, i.e $\mathrm{W}_{(i)}^{(2)} \supseteq \mathrm{W}_{(j)}^{(2)}$. Since the lattice basis will not change during the progressive sieving of Pump and running an i-dimension progressive sieving, it will run a j-dimension progressive sieving at first, for $j \leq i$.

Setting event $\mathrm{E}_\beta^{(2)}$ as the progressive sieving finds the projection of the target vector exactly after a β-dimensional sieve. More specifically, during one progressive sieving, all the sieving dimensions smaller than β failed to find the target vector but succeeded when the sieving dimension equals β. We give the following heuristic assumption.

Heuristic 4 *For* $i \in \{2, ..., d_{\mathrm{svp}}\}$, $\mathrm{W}_i^{(2)} \supseteq \mathrm{W}_{i-1}^{(2)} \supseteq \mathrm{W}_{i-2}^{(2)} \cdots \supseteq \mathrm{W}_2^{(2)}$. *Then* $\mathrm{E}_i^{(2)} = \mathrm{W}_i^{(2)} - \mathrm{W}_{i-1}^{(2)}$.

Set $\Pr\left[\mathrm{W}_{d_{\mathrm{start}}-1}^{(2)}\right]$=0, based on Heuristic 4 we calculate $\Pr\left[\mathrm{E}_{d_{\mathrm{svp}}}^{(2)}\right]$ by

$$\Pr\left[\mathrm{E}_{d_{\mathrm{svp}}}^{(2)}\right] = \Pr\left[\mathrm{W}_{d_{\mathrm{svp}}}^{(2)}\right] - \Pr\left[\mathrm{W}_{d_{\mathrm{svp}}-1}^{(2)}\right], \tag{14}$$

which is the key equality to calculate the number of gate in searching step. Then, the cumulative probability of solving LWE in Two-step LWE estimator can be expressed by

$$\begin{aligned} \Pr[\mathrm{W}] &= \Pr[\mathrm{E}_{\beta_{\mathrm{end}-1}}^{(1)}] + \left(1 - \Pr[\mathrm{E}_{\beta_{\mathrm{end}-1}}^{(1)}]\right) \sum_{i=d_{\mathrm{start}}}^{d_{\mathrm{svp}}} \Pr\left[\mathrm{E}_i^{(2)}\right] \\ &= \Pr[\mathrm{E}_{\beta_{\mathrm{end}-1}}^{(1)}] + \left(1 - \Pr[\mathrm{E}_{\beta_{\mathrm{end}-1}}^{(1)}]\right) \Pr\left[\mathrm{W}_{d_{\mathrm{svp}}}^{(2)}\right], \end{aligned} \tag{15}$$

see the line 16 of Algorithm 6 for more details.

6.2 A Refined SVP Dimension Estimation Method.

Section 2.5 introduced the classical SVP-dimension estimate, which replaces the norm of the k-dimensional target projection by the root-mean-square value $\sigma\sqrt{k}$.

Although this criterion is simple and has been used in practical LWE solvers, it does not control the success probability of the final SVP call. In particular, the actual projected norm may exceed $\sigma\sqrt{k}$, in which case the dimension selected by the classical estimate may be insufficient.

To account for this randomness, we adapt the probabilistic target-norm model previously used for BKZ-success estimation in [DSDGR20] to the standalone SVP oracle in the Search step.

To model this failure probability, let $\mathcal{D}_\sigma$ be a centered error distribution with standard deviation σ. For each $k \in [1 : d+1)$, define $X_k := \|\pi_{d-k}(\mathbf{t})\|^2$, where $\mathbf{t} = (\mathbf{e}, 1)$ is the target vector of the primal uSVP instance. Thus, X_k denotes the squared norm of the target-vector projection associated with the k-dimensional projected sublattice $\mathcal{L}_{\pi[d-k:d)}$.

Following the Gaussian projection approximation used in the probabilistic BKZ estimator of [DSDGR20], we approximate the distribution of X_k by $\sigma^2\chi_k^2$, where χ_k^2 denotes the chi-squared distribution with k degrees of freedom. Hence, for a reduced basis with log-GS profile rr, the cumulative success probability of a standalone SVP call in dimension k is estimated as $P_{\mathrm{suc}}(k; \mathsf{rr}, \sigma) := \Pr\left[\sigma^2\chi_k^2 \le \mathrm{GH}\big(\mathsf{rr}[d-k:d)\big)^2\right]$. For a prescribed success threshold p_{target}, we define $d_{\mathrm{svp}}^{\mathrm{ref}} := \min\{k \in [d_{\mathrm{start}} : d+1) : P_{\mathrm{suc}}(k; \mathsf{rr}, \sigma) \ge p_{\mathrm{target}}\}$. In our concrete estimator, we set $p_{\mathrm{target}} = 0.999$.

To address this issue, we propose `SVPDimEst` (Algorithm 7) for estimating the sieving dimension to solve general LWE problems with arbitrary target vector distributions. Our method extends the probabilistic BKZ-simulation model from [DSDGR20]. 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 provides a more accurate and directly applicable dimension estimation for a single SVP call. Similarly to Lemma 2, a detailed description is provided in Appendix A.1. It retains the same properties as stated in Lemma 1.

6.3 Time-Cost and Memory Models.

We concretize the time-cost model by adopting the gate-count model. According to the time-cost model introduced in Section 2.6, when the reduction procedure is BKZ and the search procedure is the asymptotic sieving method, their time and memory models can be described as follows:

Gate count of reduction step. In this part, we introduce how to count the number of gates when we solved LWE in the reduction step. After we calculate each $\Pr[\mathrm{E}_{\beta_i}^{(1)}]$ value for $i \in \{0, \cdots, \mathrm{end}-1\}$ by using Eq. (13) in the reduction step, we can calculate the expected value of gate count G_1 of reduction step. We evaluate the expected value of gate counts G_1 of reduction step by Eq. (16), see line 9 of Algorithm 6 for more details. Let $\mathtt{gate}(k)$ denote the gate count of one full sieve in dimension k. Under the exponential approximation $\mathtt{gate}(k) =$

$2^{ck+o(k)}$, the cumulative gate count of a progressive sieve that increases its sieving dimension from 1 to k is $\texttt{pgate}(k) := \sum_{j=1}^{k} \texttt{gate}(j) \approx C_{\text{prog}} \cdot \texttt{gate}(k)$, where $C_{\text{prog}} := \frac{1}{1-2^{-c}}$. For the classical BDGL sieve exponent $c = 0.292$, this gives $C_{\text{prog}} \approx 5.46$. When accounting for the dimension-for-free technique in BKZ tour, then define $\texttt{pbgate}(\beta) := (d - \beta + 1)\,\texttt{pgate}(\beta - \texttt{d4f}(\beta))$. Here, the $\texttt{d4f}(j)$ is calculated by Section 2.4. Then G_1 can be expressed as

$$G_1 = \sum_{i=0}^{\text{end}-1} \Pr[\mathrm{W}_{\beta_i}^{(1)}] \cdot \left(1 - \Pr[\mathrm{E}_{\beta_{i-1}}^{(1)}]\right) \cdot \left[\sum_{l=0}^{i} \texttt{pbgate}(\beta_l)\right]. \tag{16}$$

Gate count of searching step. In this part, we introduce how to calculate the gate count when solving LWE in the searching step. When we solved uSVP in the searching step, it meant that all the BKZ tours in the reduction step failed to find the target vector. Thus, based on Eq. (14) to calculate $\Pr\left[\mathrm{E}_i^{(2)}\right]$, $i \in \{d_{\text{start}}, \ldots, d_{\text{svp}}\}$, we use Eq. (17) to calculate the expected value of gates of the searching step, see line 15 of Algorithm 6 for more details.

$$\begin{aligned} G_2 = \sum_{i=d_{\text{start}}}^{d_{\text{svp}}} \Pr\left[\mathrm{E}_i^{(2)}\right] \cdot \left(1 - \Pr[\mathrm{E}_{\beta_{\text{end}-1}}^{(1)}]\right) \cdot \\ \left[\sum_{l=0}^{\text{end}-1} \texttt{pbgate}(\beta_l) + \texttt{pgate}\left(i - \ \texttt{d4f}(i)\right)\right] \end{aligned} \tag{17}$$

When considering the cost of solving LWE during the searching step, it means that all BKZ tours in the reduction step failed to find the target vector. We calculate the $\Pr\left[\bigwedge_{j=0}^{\text{end}-1} \mathrm{F}_{\beta_j}^{(1)}\right]$ in Eq. (17) to represent the probability of all BKZ tours in the reduction step failed to find the target vector. Besides, before starting the large dimensional sieve in the searching step, the total time cost of solving the uSVP in the searching step already contains the full-time cost of all BKZ tours in the reduction step.

Therefore, the total gate count of the reduction step is $\sum_{l=0}^{\text{end}-1} \texttt{pbgate}(\beta_l)$ and when the dimension of SVP Oracle we considered equals d_{svp}, the gate count of searching step is $\texttt{pgate}(d_{\text{svp}} - \texttt{d4f}(d_{\text{svp}}))$.

Finally, the total gate count for the Two-step mode of solving LWE $G := G_1 + G_2$.

Memory count of Two-step LWE concrete estimator. The memory count of the Two-step LWE concrete estimator is similar to gate count, just replace the function $\texttt{gate}(\beta)$ with the memory cost function $\texttt{bit}(\beta)$ which declares the memory cost of one sieve algorithm with dimension β. Since the memory cost of the final sieve and (progressive) BKZ with the same dimension is equal to the memory cost of one sieve algorithm with the same dimension, the memory count of the reduction process in our Two-step LWE Estimator is

$$B_1 = \sum_{i=0}^{\text{end}-1} \left[\Pr[\mathrm{W}_{\beta_i}^{(1)}] \cdot \left(1 - \Pr[\mathrm{E}_{\beta_{i-1}}^{(1)}]\right)\right] \cdot \texttt{bit}(\beta_i - \texttt{d4f}(\beta_i)), \tag{18}$$

and the memory count of searching step is

$$B_2 = \sum_{i=d_{\text{start}}}^{d_{\text{svp}}} \Pr\left[\mathrm{E}_i^{(2)}\right] \cdot \left(1 - \Pr[\mathrm{E}^{(1)}_{(\beta_{\text{end}}-1)}]\right) \cdot \mathtt{bit}\left(i - \ \mathtt{d4f}(i)\right). \tag{19}$$

The total memory count for Two-step mode of solving LWE $B := B_1 + B_2$.

6.4 Two-step LWE Estimator with Trivial Strategy

We set the reduction method to BKZ, choose Pump as the SVP algorithm, and use the trivial reduction strategy $\mathsf{S}_{\text{end}} = [\![3, 4, \cdots, \text{end} + 2]\!]$. Algorithm 6 gives the corresponding Two-step LWE Estimator[4].

input : n, m, q, σ, S;
output: $\mathsf{GB}_{\min}$;
Function `TwoStepLWEEstimator`$(n, m, q, \sigma, \mathsf{S})$**:**
  $\mathsf{GB}_{\min} \leftarrow (+\infty, +\infty)$; $\mathsf{GB} \leftarrow (0, 0)$; $\mathsf{GB}_{\text{pre}} \leftarrow (0, 0)$; $p_{\text{tot}} \leftarrow 0$;
  $\mathsf{rr} \leftarrow$ Estimated log-GS profile of an LLL-reduced $\mathrm{LWE}_{n,m,q,\mathcal{D}_\sigma}$ instance;
  for $i \leftarrow 0$ **to** $\sharp\mathsf{S} - 1$ **do**
    `// Extract` $\beta_i \leftarrow \mathsf{S}[i]$ `or` $(\beta_i, J_i) \leftarrow \mathsf{S}[i]$
    $\mathsf{rr} \leftarrow \mathsf{BKZSim}(\mathsf{rr}, \beta_i)$; // $\mathsf{PnJBKZSim}(\mathsf{rr}, \beta_i, J_i)$ if $J_i > 1$;
    $P(\beta_i) \leftarrow \Pr\left[x \leftarrow \sigma^2\chi^2_{\beta_i} \middle| x \le \exp(l_{d-\beta_i})^2\right]$;
    $\mathsf{GB}_{\text{cum}} \leftarrow (\sum_{k=0}^{i} \mathtt{pbgate}(\beta_k, J_k), \mathtt{bit}(\beta_i - \mathtt{d4f}(\beta_i)))$;
    $\mathsf{GB}_{\text{pre}} \leftarrow \mathsf{GB}_{\text{pre}} + \mathsf{GB}_{\text{cum}} \cdot (1 - p_{\text{tot}}) \cdot P(\beta_i)$;
    $p_{\text{tot}} \leftarrow p_{\text{tot}} + (1 - p_{\text{tot}}) \cdot P(\beta_i)$; $\mathsf{GB}_{\text{csieve}} \leftarrow (0, 0)$; $P(d_{\text{start}} - 1) \leftarrow 0$;
    for $d_{\text{svp}} \leftarrow d_{\text{start}}$ **to** d **do**
      $P(d_{\text{svp}}) \leftarrow \Pr\left[x \leftarrow \sigma^2\chi^2_{d_{\text{svp}}} \middle| x \le (\mathrm{GH}(\mathsf{rr}_{[d-d_{\text{svp}}:d)}))^2\right]$;
      $\mathsf{G}_{\text{curgate}} \leftarrow \mathsf{GB}_{\text{cum}}[0] + \mathtt{pgate}(d_{\text{svp}} - \mathtt{d4f}(d_{\text{svp}}))$;
      $\mathsf{G}_{\text{curbit}} \leftarrow \max\{\mathsf{GB}_{\text{cum}}[1], \mathtt{bit}(d_{\text{svp}} - \mathtt{d4f}(d_{\text{svp}}))\}$;
      $\mathsf{GB}_{\text{csieve}} \leftarrow$
       $\mathsf{GB}_{\text{csieve}} + (\mathsf{G}_{\text{curgate}}, \mathsf{G}_{\text{curbit}}) \cdot (1 - p_{\text{tot}}) \cdot (P(d_{\text{svp}}) - P(d_{\text{svp}} - 1))$;
      if $p_{\text{tot}} + (1 - p_{\text{tot}}) \cdot P(d_{\text{svp}}) \ge p_{\text{target}}$ **then**
        $\mathsf{GB} \leftarrow \mathsf{GB}_{\text{pre}} + \mathsf{GB}_{\text{csieve}}$;
        if $\mathsf{GB}[0] < \mathsf{GB}_{\min}[0]$ **then**
          $\mathsf{GB}_{\min} \leftarrow \mathsf{GB}$;
        break;
  return $\mathsf{GB}_{\min}$;

Algorithm 6: Two-step LWE Estimator

[4] Batch "refined-lwe-estimator" in https://github.com/Summwer/lwe-estimator-with-pnjbkz.git

6.5 Two-step LWE Estimator with Refined Strategy

PSSearch selects S_{op} under the strategy-search model defined in Section 4. The concrete estimator in this section then evaluates S_{op} and S_0 under the fixed cumulative-success target $p_{\mathrm{target}} = 0.999$ and the probability-weighted gate-count model. Accordingly, the optimality statements of Theorems 1 and 2 concern the strategy-search model.

By applying the strategy selection algorithm PSSearch to the Two-step LWE estimator, we obtain a security evaluation of LWE instances under the two-step mode, where the PSSearch-selected strategy is evaluated using the refined two-step estimator.

The main differences between the PSSearch-based estimator and the two-step LWE Estimator proposed in Section 6.4 are as follows. First, the use of PnJBKZ allows us to adjust the reduction strategy more freely. Instead of the trivial reduction strategy $\mathsf{S} = [\![(3,1),(4,1),\cdots,(\mathrm{end}+2,1)]\!]$ used in leaky-LWE-Estimator and Section 6.4, we can choose a more efficient reduction strategy given by PSSearch (Algorithm 3).

Secondly, we use the PnJBKZ simulator proposed in Section 5 instead of the original BKZ simulator to simulate how the quality of lattice basis changes during the reduction by a series of PnJBKZ with $J > 1$. The simulator is purely based on the Gaussian Heuristic, which avoids the problem that GSA (Heuristic 1) is not strictly held during the reduction of PnJBKZ.

For notational uniformity, we use $\texttt{pbgate}$ to denote the modeled gate count of one reduction tour. The one-argument notation $\texttt{pbgate}(\beta)$, used for the BKZ-based estimator in Section 6.3, is the $J = 1$ specialization of the two-argument model: $\texttt{pbgate}(\beta) := \texttt{pbgate}(\beta, 1)$. Let $\texttt{pbgate}(\beta, J) := \frac{d-\beta+1}{J}\texttt{pgate}\big(\beta - d4f(\beta)\big)$. for a PnJBKZ-$(\beta, J)$ tour,

The gate count and memory count can be calculated similarly, we only need to replace the events $\mathrm{W}^{(1)}_{\beta_i}, \mathrm{F}^{(1)}_{\beta_i}$ by $\mathrm{W}^{(1)}_{(\beta_i,J_i)}, \mathrm{F}^{(1)}_{(\beta_i,J_i)}$ which allow $J > 1$ when calculating the probability. We omit further details here.

7 Primal-Attack Cost Estimation for NIST Schemes

We re-estimate the hardness of the LWE-based NIST schemes [CEN22] to estimate the influence of the optimized solving strategy found by PSSearch. Under the classical gate-count model, with memory reported separately, the primal attack estimated security bit of LWE in NIST schemes [CEN22] is reduced by 3.4∼4.6 bit compared to the estimation generated by Leaky-LWE-Estimator[6] in [DSDGR20] under gate-count model, which adopts the improved list-decoding technique proposed in [MAT22]. The estimated results are listed in Table 2. Our new concrete hardness estimation of LWE[7] answers Q7 in Section 5.3 of Kyber Round-3 specification [ABD+21] and narrows the security estimation error interval. For more details, refer to the citation [XWW+24].

[6] https://github.com/lducas/leaky-LWE-Estimator

[7] https://github.com/Summwer/lwe-estimator-with-PnJBKZ.git

Table 2: Refined Two-Step Primal-Attack Cost Estimates for Selected NIST Schemes.[♮]

	$\log_2$ G/$\log_2$(gates)				$\log_2$ B/$\log_2$(bits)			$\Delta \log_2$ G	
	NIST Required	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	3.4	4.6
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. "$\mathsf{S}_0 = [\![(3,1),(4,1),\cdots,(\beta+2,1)]\!]$" is a trivial progressive BKZ+Pump in Two-step mode to estimate security as [XWW+24] 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" in the base-2 logarithm of the gate count for strategies S_0 or S_{op}. The gate count of all estimations in this Table uses the same improved list-decoding technique proposed by MATZOV [MAT22].

Comparison with the dual attack. The gate-count model used in our estimator coincides with the classical circuit model CC of [CMHST25], including the same underlying cost model for lattice sieving and nearest-neighbor search. Therefore, the corresponding $\log_2$-gate estimates are directly comparable at the cost-model level. For Kyber512, Kyber768, and Kyber1024, our optimized primal-attack estimates are 141.4, 204.3, and 276.5 bits, respectively, whereas [CMHST25] reports dual-attack costs of 139.5, 195.1, and 259.7 bits under its CC model. Thus, under the native success-probability conventions of the two analyses, the dual attack gives the lower reported cost for all three Kyber parameter sets. Consequently, Table 2 should be interpreted as refining the estimated cost of the primal-attack branch, rather than as providing the overall minimum attack cost for Kyber. We also note that our estimator targets cumulative success probability at least 0.999, whereas the parameter sets in [CMHST25] provide a constant attack success probability of approximately 0.31; a success-probability-normalized comparison would therefore require an additional amplification convention.

Assuming independent executions and sequential termination after verified recovery of the complete secret, increasing the cumulative success probability from approximately 0.31 to at least 0.999 requires at most 19 executions. The expected number of executions is approximately 3.223, corresponding to an expected-work overhead of 1.69 bits. The normalized dual estimates are therefore approximately 141.2, 196.8, and 261.4 bits.

Under an expected-work normalization to the same full-secret-recovery target, the primal and dual estimates are nearly tied for Kyber-512, while the dual estimate remains lower for Kyber-768 and Kyber-1024.

8 Experiments

In this section, we present extensive experimental results that validate the accuracy of the essential components designed for the construction of the Strategy Search (SSearch) algorithm and the Pruning SSearch (PSSearch) algorithm. Additionally, we demonstrate the multiple-fold improvement in solving efficiency achieved through strategy optimization in practical applications. The main contents of this section include:

-Verification experiments for the accuracy of the PnJBKZ simulator in Section 8.1, which is a key component of PSSearch;

-Verification experiments for the refined SVP dimension estimation in Section 8.2;

-Verification experiments for the efficiency of the two-step mode in Section 8.3;

-Verification experiments for simulation accuracy of MinTwoStepSolver in Section 8.4;

-The optimized solving strategies generated by PSSearch, introduced in Section 8.5;

-A comparison of the practical performance of different LWE solving strategies in Section 8.6;

-New TU Darmstadt LWE Challenge records successfully completed using optimized solving strategies in Section 8.7;

-The acceleration effects achieved by the optimal strategies discovered through the strategy search algorithm, as discussed in Section 8.8 and 8.9.

Specifically, 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.

The MinTwoStepSolver algorithm combines TwoStepSolver with a strategy generated by the PSSearch, all under a specific time-cost model $\mathcal{T}$, to effectively solve LWE.

8.1 Verification Experiments for PnJBKZ simulator

Appendix C.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 [WWW23]. This refined estimation is used to adjust the optimistic `d4f` settings during the simulation of PnJBKZ reduction. For further details, see Section 5.2.

The experimental results in Appendix C.1.2 quantify the accuracy of the GS-norm ratios $\frac{\mathrm{Real}(\|\mathbf{b}_i^*\|)}{\mathrm{Sim}(\|\mathbf{b}_i^*\|)}$ under practical `d4f` settings. Across all recorded tours of the 20 tested practical parameter configurations, 79.69% of the 60,416 ratios lie in $[0.95, 1.05]$, and 99.03% lie in $[0.90, 1.10]$. Each ratio corresponds to one

GS coordinate after one tour for one parameter configuration, with the observed norm aggregated as a geometric mean over 20 independent trials. The mean absolute deviation of these ratios from 1 is 0.0293. These statistics include all GS coordinates and exclude the separate ideal-Pump experiments. For details on predicting PnJBKZ with the `d4f` technique, see Section 5.2 and Appendix C.1.2.

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 C.1.4, 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. Using the rounded slope values reported in these tables, the relative slope prediction error, defined as the absolute difference between the simulated and observed slopes divided by the absolute observed slope, does not exceed 7.26% over the 21 reported reduction tours. Therefore, the PnJBKZ simulator we have constructed is sufficiently effective for its intended purpose. Additional verification results are provided in Appendices C.1.2 and C.1.3, together with the reproducibility instructions and stored experimental data described there.

8.2 Verification Experiments for the Refined SVP Dimension Estimation

In particular, we use different parameters of the LWE instances[4] to test the success probabilities of the final sieve when using different progressive sieving dimensions shown in Figure 7.[5] We choose four different LWE parameters (n=40, α=0.005, q=1601, m=1600), (n=40,α=0.015, q=1601, m=1600), ($n = 60, \alpha = 0.005$, $q = 3607$), (n=45, α=0.010, q=2027, m=2025) for our experiments. For each LWE parameter, we initialize 100 random LWE instances to construct 100 different lattice bases. Each lattice basis corresponds to a uSVP instance with a different target vector. Then we use BKZ/PnJBKZ to do pre-processing by some trivial reduction strategy S. Using LWE parameter (n=40, α=0.005, q=1601, m=1600) for example, we set $\mathsf{S} = [\![10, 11, \cdots, 17]\!]$. The 100 independently generated instances are used to estimate the empirical recovery rate and compare it with the probability predicted by the chi-squared projection model.

After pre-processing, we set the key parameter $\kappa \in \{0, ..., d-1\}$ to determine the size of the final sieve in the searching phase. In [ADPS16], it is assumed that one can solve an LWE by solving a $d - \kappa$ dimension SVP on $\mathcal{L}_{\pi[\kappa:d)}$ as long as $\sigma\sqrt{d-\kappa} < \mathrm{GH}(\mathcal{L}_{\pi[\kappa:d)})$. Here $\sigma\sqrt{d-\kappa}$ is the expected norm of the projected target vector. However, since we consider the square sum of the length

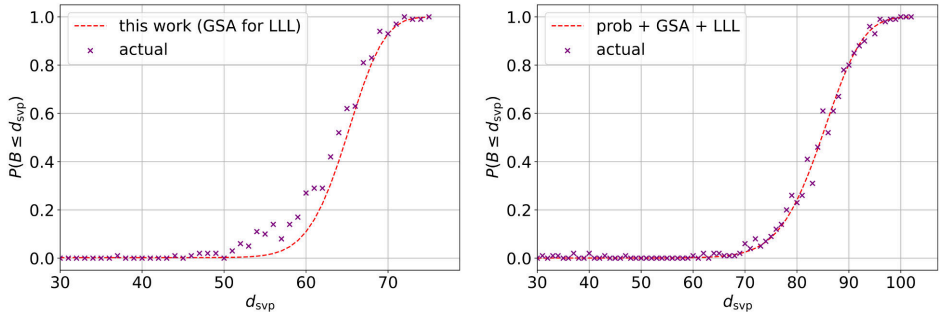
[4] https://www.latticechallenge.org/lwe_challenge/challenge.php

[5] We've uploaded our verification experiment in the file "lwe_prob_test.py " in the website https://github.com/Summwer/test-for-refined-lwe-estimator.

of the projected target vector as a chi-squared distribution with $d-\kappa$ degrees of freedom, we calculate the cumulative probability of solving LWE when using a high-dimension Pump in Section 6 by Eq. (15), and the line 16 of Algorithm 6. To verify the Heuristic 4 and the accuracy of Eq. (15), we test the actual success rate of solving LWE under different lattice sieving with different κ value.

More precisely, we set $d_{\text{svp}} = d-\kappa$ in lattice sieving from 30 to d by adjusting the value of κ and use each sieve with different d_{svp} value to try to find the solution of LWE on 100 different lattice basis after pre-processing. Meantime, we record the actual success rate of each sieve with different d_{svp} values on 100 different lattice bases. Finally, we compare the actual success rate of each sieve with different d_{svp} with our estimation success rate of solving LWE by the final sieve in Eq. (15), and the line 16 of Algorithm 6. See Figure 7 for more detail.

From Figure 7 we can see that the prediction of the success rate of solving LWE given by Eq. (15) is consistent with the experimental results, which means our analysis and estimation in Section 6 is accurate.



(a) $n = 40, \alpha = 0.005,\ q = 1601$ (b) $n = 40, \alpha = 0.015,\ q = 1601$

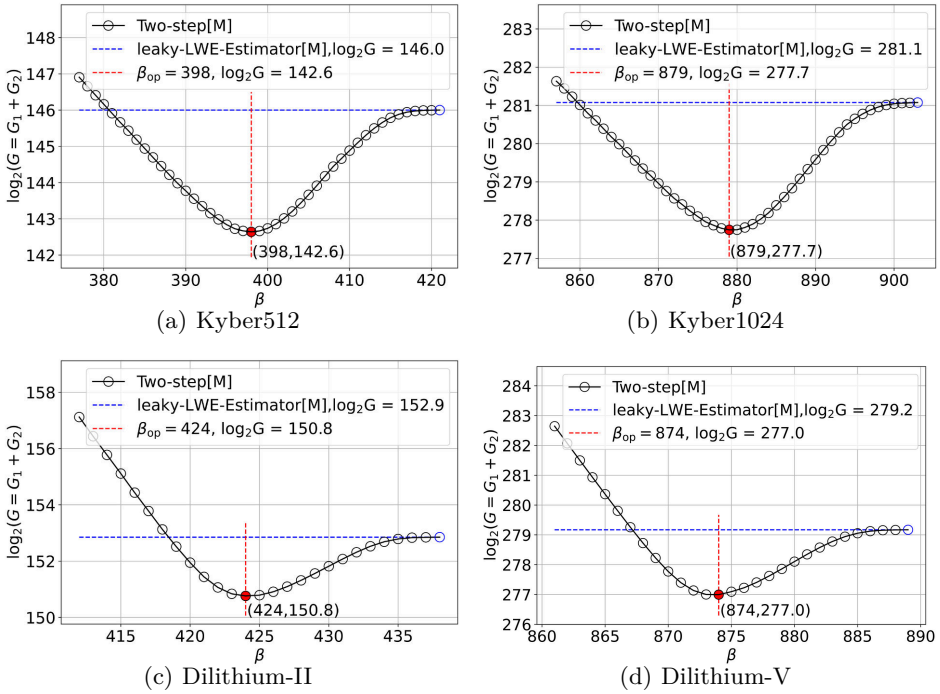
(c) $n = 60, \alpha = 0.005,\ q = 3607$ (d) $n = 45, \alpha = 0.010,\ q = 2027$

Fig. 7: Verification experiments of the fitness of the theoretical total success probability $P(d_{\text{svp}})$ = Eq. (15) (the dashed line) to the actual success probability. Test 100 trials and count the success rate for each d_{svp}.

8.3 Verification Experiments for Efficiency of Two-step Mode

In this part, we give an experiment to verify the efficiency of the Two-step mode. In the experiment, we test the public keys of Kyber512, Kyber1024, Dilithium-II, and Dilithium-V as the input LWE instances, then call a Two-step Estimator with $\mathsf{S}_{\leq\beta} = [\![3, 4, \cdots, \beta]\!]$. The estimator stops at $\beta = \beta_{\text{end}-1}$ such that the accumulated probability of $\mathsf{S}_{\leq\beta_{\text{end}-1}}$ is no less than 0.999, i.e. $\sum_{i=0}^{\text{end}-1}\left[\Pr[W_{\beta_i}^{(1)}]\cdot\left(1-\Pr[E_{\beta_{i-1}}^{(1)}]\right)\right] \geq 0.999$, which is also the condition in leaky-LWE-Estimator.

The Figure 8 shows the gate count [AGPS20] of Two-step mode under different reduction strategy $\mathsf{S}_{\leq\beta}$, where $\beta \in \{3, \ldots, \beta_{\text{end}-1}\}$ and estimated the number of gates given by the leaky-LWE-Estimator. The x-axis of Figure 8 is the final blocksize β in reduction strategy $\mathsf{S}_{\leq\beta}$. Figure 8 reflects that in solving LWE, the Two-step mode is more efficient than that of using BKZ reduction only and the security estimation given by the leaky-LWE-Estimator [DSDGR20] is indeed an over-optimistic estimation. Besides, there is optimal timing β_{op} for ending the reduction and entering the searching step as the quality of the lattice basis improved gradually by progressive BKZ.



(a) Kyber512 (b) Kyber1024

(c) Dilithium-II (d) Dilithium-V

Fig. 8: Two-step efficiency verification Experiment.

8.4 Verification Experiments for Simulation Accuracy of MinTwoStepSolver

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 C.1.4 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	−0.0275	6.2
(80, 10)	−0.0250	6.3	−0.0244	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	−0.0192	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 [DSvW21].

8.5 Optimized strategy generated by MinTwoStepSolver

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

Table 5: 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 5, 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`"

[5] The data in Table 3 is extracted from a test in Figure 9 for comparing the quality and walltime between our simulations and actual experiments. For more results please see Appendix C.1.4.

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 C.3.

8.6 Efficiency of MinTwoStepSolver for solving LWE

We use the Blocksize and Jump Strategy Selection Algorithm(BSSA, Appendix B) for comparison in experiments. BSSA is derived from ProBKZ [AWHT16]. It replaces BKZ and enumeration with PnJBKZ and Pump, respectively, while retaining the shortest-path strategy-selection procedure.

From the result of Figure 9 and Table 4, we observe that using the strategy selected by PSSearch (resp. BSSA) significantly decreased the walltime cost by about $7.2 \sim 23.4$ ($5.2 \sim 10.2$) 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 Figure 9 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, Figure 10 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.

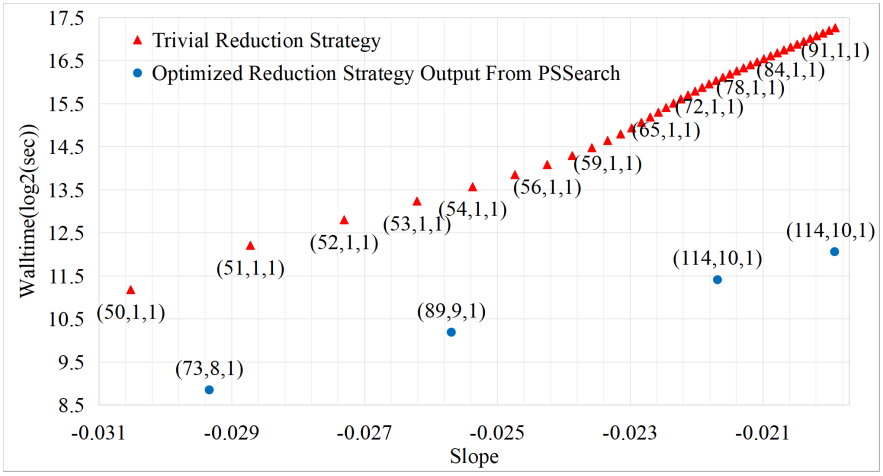
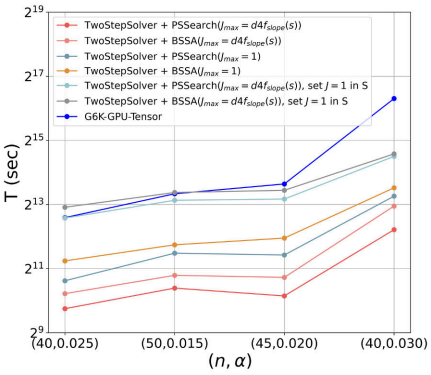
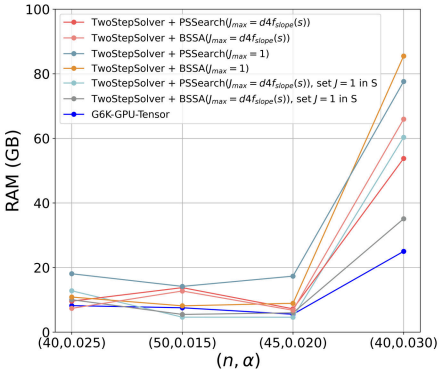


Fig. 10: Speedup in Reduction Step via the optimized strategy generated from PSSearch.Test on LWE challenge (n=60,α = 0.010), which constructs a 222-dimensional lattice basis by using the primal attack. The points are labeled by the reduction parameters (β,J,tours).

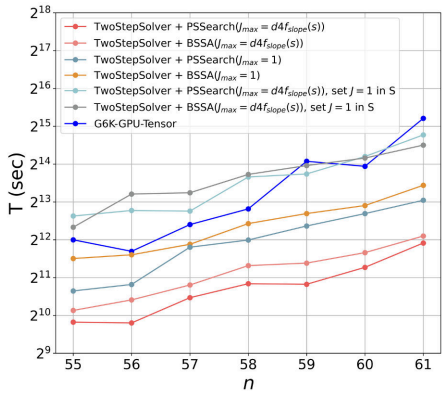
All experimental results, except those in Table 4, were obtained using 32 threads and 2 GPUs on a workstation with an Intel Xeon 5128 (16 cores, 32



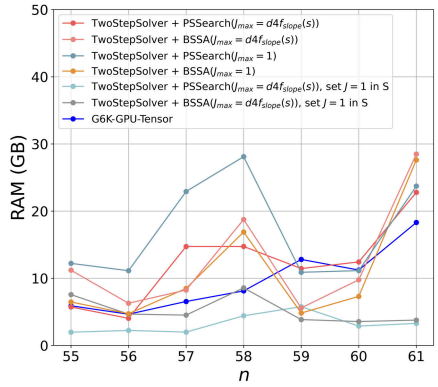
(a) LWE Challenges: Time.



(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 first generating a strategy S with $J_{\max} > 1$, and then setting every jump value to 1 while retaining the blocksize sequence.

Fig. 9: Comparison of Different LWE-solving Algorithms under the same benchmark conditions. §

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 C.2.1. For more information about BSSA, refer to Appendix B.

8.7 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 Figure 4. Specifically, we used the following machines for our experiments: Machine A, with AMD EPYC^{TM} 7002 Series 128@2.6GHz, NVIDIA 3090 * 8, 1.5T RAM; Machine B, with AMD EPYC^{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 $(n, \alpha) = (40, 0.035)$ using the G6K-GPU [DSvW21] is 23.4 times longer than the time of our method.

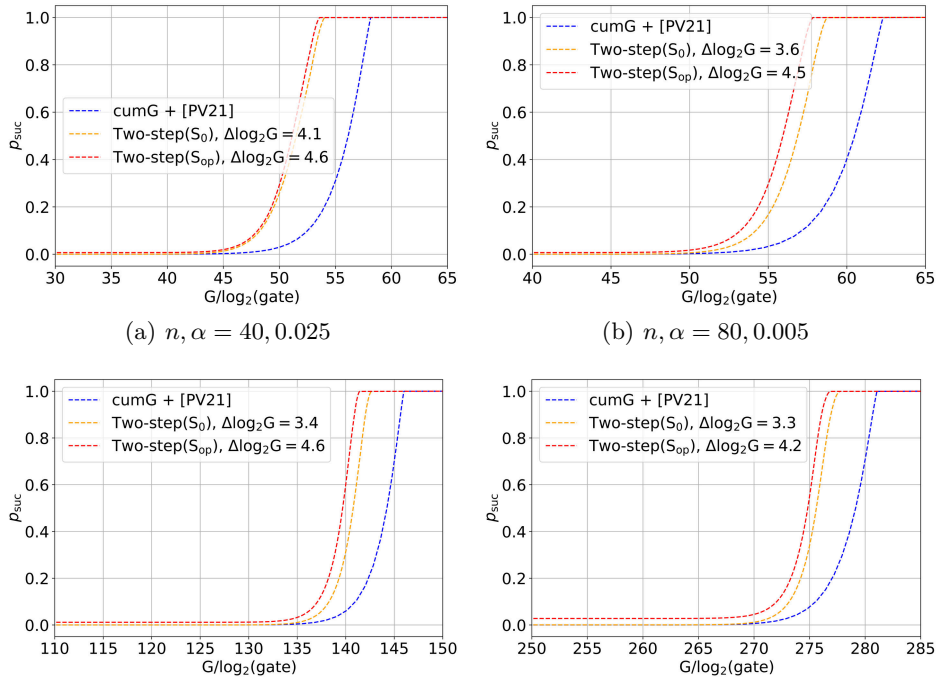
8.8 The comparison of different estimation modes

In this part, we compare our Two-step mode estimator using different reduction strategies and different gate count models with the leaky-LWE-Estimator.

Figure 11 compares the expected gate count and cumulative success probability of different LWE estimators. The blue curves correspond to progressive BKZ in BKZ-only mode. The curves labeled "Two-step(S_0)" use the trivial progressive reduction strategy S_0, followed by the final Search step. For the instances shown, S_0 is a proper prefix of the reduction sequence used in the corresponding BKZ-only run: the Two-step baseline ends the reduction stage earlier and then switches to Search. The curves labeled "Two-step(S_{op})" use the optimized reduction strategy selected by PSSearch, with optimized blocksizes and Jump values, followed by the final Search step. [6]

From Figure 11 we can see that in both the LWE challenge instances and the LWE instances in NIST standard algorithms, the accumulation success probability of solving LWE by Two-step mode approaches 1 is much faster than that of the leaky-LWE-Estimator. In addition, the expected number of gates in the Two-step solving mode is smaller than that of the leaky-LWE-Estimator when

[6] We use the gate-count model which adopts the improved list-decoding technique proposed in [MAT22]. It fixed the estimate done in [AGPS20] of the list-decoding technique proposed in [BDGL16].



(a) $n, \alpha = 40, 0.025$ (b) $n, \alpha = 80, 0.005$

(c) Kyber512 (d) Kyber1024

Fig. 11: The relation among the growth of cumulated cost and the success probability. Comparison between the output of cumulated Cost Version of [PV21] and Two-step mode(Algorithm 6, this work) for LWE challenge $(n, \alpha) \in \{(40, 0.025), (80, 0.005)\}$ and on Kyber 512 and Kyber 1024 [ABD+21]. "Two-step(S_0)" uses a trial progressive BKZ+Pump in Two-step mode to estimate security. "Two-step(S_{op})" uses a progressive BKZ+Pump with the optimized strategy selected by PSSearch in Two-step mode to estimate security. We set $\Delta \log_2 \mathrm{G}$ as the gate count difference between our estimator and the leaky-LWE-estimator both using the same gate count[6].

the accumulation success probability of solving LWE approaches 1. Therefore, the evaluation result shows that the leaky-LWE-Estimator gives an optimistic estimation. Besides, both the optimized blocksize and Jump selection strategy and the improved list-decoding technique proposed in [MAT22], which fixed the estimate done in [AGPS20] of the list decoding technique proposed in [BDGL16], can further decrease the estimated security strength by replacing the trivial reduction strategy or gate-count model in Two-step mode. See Figure 11 for more details about the difference between different estimation models.

8.9 The Acceleration Effect of the Optimized Strategy

8.9.1 Effect of the Practical Sieve-Cost Model. 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

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

As shown in the expression 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. 9, 10 and Table 4.

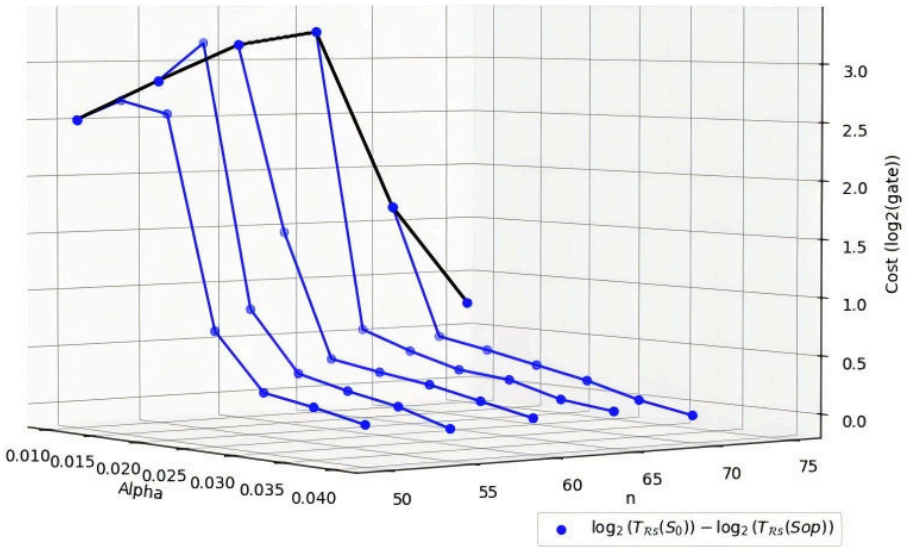


Fig. 12: Acceleration effect of the optimized strategy. LWE challenge instances with dimension and error rate parameters (n,α).

Figure 12 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;

(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. Under the stated primal-attack and gate-count assumptions, three primal-attack estimates fail to meet the required security level shown in Table 2. Relative to the trivial Two-step solving strategy S_0, the modeled costs of the primal-attack branch decrease by 1.2–1.3 bits.

8.9.2 The Acceleration Effect grows with (n, q) To evaluate the efficiency of our approach, we randomly generated Learning with Errors (LWE) instances with a fixed error standard deviation σ while varying the LWE dimension n and the modulus q. The time cost for the security evaluation was measured using the `gate_count` metric introduced by Matzov. Figure 13 illustrates the difference in the base-2 logarithm of the time cost, denoted as $\Delta(\log_2 T) = \log_2(T(\mathsf{S}_0)) - \log_2(T(\mathsf{S}_{\mathrm{op}}))$, between our optimized strategy (S_{op}) and the trivial strategy S_0. As shown in the figure, the time cost advantage of S_{op} over S_0 expands significantly when n and q increase concurrently. By extracting the maximum advantage points across the parameter space, we empirically fitted the trend line (highlighted in orange line): $\Delta(\log_2 T) = 0.01887n - 0.98$ under the scaling relationship $\log_2 q = 0.198n^{0.832}$.Along this tested scaling relation, the empirically fitted speedup factor grows exponentially with n.

9 Conclusion and Scope of the Results

We presented a unified strategy-search framework for two-step primal LWE attacks and instantiated it with PnJBKZ reduction and progressive lattice sieving. Within the strategy space and under the specified simulator and cost model, PSSearch selects a minimum modeled-cost strategy. Our implementation shows that reduction-schedule optimization can substantially improve practical LWE solving, while the concrete estimates demonstrate an additional reduction in the modeled cost of the primal-attack branch.

The conclusions are conditional on the Gaussian and independence heuristics, the accuracy of the PnJBKZ simulator, and the adopted sieving-cost model. The reported ML-KEM and ML-DSA figures therefore represent primal-attack estimates rather than overall security levels. Under a success-probability-normalized comparison, recent dual attacks remain less costly for the larger ML-KEM parameter sets. Extending the strategy-search methodology to attack-specific stan-

dalone SVP, dual, hybrid, BDD, and approximate-CVP terminal procedures is left for future work.

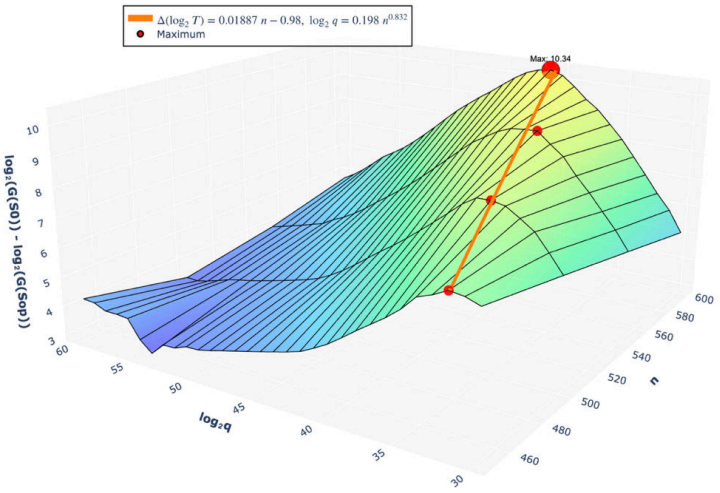


Fig. 13: Acceleration effect of the optimized strategy over the trivial strategy. The 3D surface shows the base-2 logarithmic time complexity difference $\Delta(\log_2 T)$ across varying LWE dimensions n and moduli q with a fixed σ. The orange line indicates the maximum advantage, bounded by the fitted relations $\Delta(\log_2 T) = 0.01887n - 0.98$ and $\log_2 q = 0.198n^{0.832}$.

References

ABD+21. Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS-Kyber: Algorithm specifications and supporting documentation, version 3.02. Submission to Round 3 of the NIST Post-Quantum Cryptography Standardization Project, August 2021. https://pq-crystals.org/kyber/data/kyber-specification-round3-20210804.pdf.

ABF+20. Martin R. Albrecht, Shi Bai, Pierre-Alain Fouque, Paul Kirchner, Damien Stehlé, and Weiqiang Wen. Faster enumeration-based lattice reduction: Root hermite factor$k^{1/(2k)}$ time $k^{k/8+o(k)}$. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology – CRYPTO 2020*, pages 186–212, Cham, 2020. Springer International Publishing.

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

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

ADM+24. Gorjan Alagic, Quynh Dang, Dustin Moody, Angela Robinson, Hamilton Silberg, Daniel Smith-Tone, et al. Module-lattice-based key-encapsulation mechanism standard. 2024.

ADPS16. Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. Post-quantum key Exchange—A new hope. In *25th USENIX Security Symposium (USENIX Security 16)*, pages 327–343, Austin, TX, August 2016. USENIX Association.

AFG14. Martin R. Albrecht, Robert Fitzpatrick, and Florian Göpfert. On the efficacy of solving lwe by reduction to unique-svp. In Hyang-Sook Lee and Dong-Guk Han, editors, *Information Security and Cryptology – ICISC 2013*, pages 293–310, Cham, 2014. Springer International Publishing.

AGPS20. Martin R. Albrecht, Vlad Gheorghiu, Eamonn W. Postlethwaite, and John M. Schanck. Estimating quantum speedups for lattice sieves. In Shiho Moriai and Huaxiong Wang, editors, *Advances in Cryptology – ASIACRYPT 2020*, pages 583–613, Cham, 2020. Springer International Publishing.

AGVW17. Martin R. Albrecht, Florian Göpfert, Fernando Virdia, and Thomas Wunderer. Revisiting the expected cost of solving usvp and applications to lwe. In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology – ASIACRYPT 2017*, pages 297–322, Cham, 2017. Springer International Publishing.

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

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

AYH. Martin R. Albrecht, Cathie Yun, and Hamish Hunt. lattice-estimator. https://github.com/malb/lattice-estimator.

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

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

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

BGJ15. Anja Becker, Nicolas Gama, and Antoine Joux. Speeding-up lattice sieving without increasing the memory, using sub-quadratic nearest neighbor search. https://eprint.iacr.org/2015/522, 2015.

BMW19. Shi Bai, Shaun Miller, and Weiqiang Wen. A refined analysis of the cost for solving lwe via usvp. In Johannes Buchmann, Abderrahmane Nitaj, and Tajjeeddine Rachidi, editors, *Progress in Cryptology – AFRICACRYPT 2019*, pages 181–205, Cham, 2019. Springer International Publishing.

Boy13. Xavier Boyen. Attribute-based functional encryption on lattices. In Amit Sahai, editor, *Theory of Cryptography*, pages 122–142, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

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

CEN22. Information Technology Laboratory COMPUTER SECURITY RESOURCE CENTER. Post-quantum cryptography pqc selected algorithms 2022. https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022, 2022.

Che13. Yuanmi Chen. *Réduction de réseau et sécurité concrète du chiffrement complètement homomorphe*. PhD Thesis, 2013.

CKKS17. Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. Homomorphic encryption for arithmetic of approximate numbers. In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology – ASIACRYPT 2017*, pages 409–437, Cham, 2017. Springer International Publishing.

CMHST25. Kevin Carrier, Charles Meyer-Hilfiger, Yixin Shen, and Jean-Pierre Tillich. Assessing the impact of a variant of matzov's dual attack on kyber. In *Annual International Cryptology Conference*, pages 444–476. Springer, 2025.

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

DLL+24. Thinh Dang, Jacob Lichtinger, Yi-Kai Liu, Carl Miller, Dustin Moody, Rene Peralta, Ray Perlner, Angela Robinson, et al. Module-lattice-based digital signature standard. 2024.

DLP14. Léo Ducas, Vadim Lyubashevsky, and Thomas Prest. Efficient identity-based encryption over ntru lattices. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology – ASIACRYPT 2014*, pages 22–41, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

DSDGR20. Dana Dachman-Soled, Léo Ducas, Huijing Gong, and Mélissa Rossi. LWE with Side Information: Attacks and Concrete Security Estimation. In *Advances in Cryptology – CRYPTO 2020: 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17–21, 2020, Proceedings, Part II*, pages 329–358, Berlin, Heidelberg, August 2020. Springer-Verlag.

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

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

EK20. Thomas Espitau and Paul Kirchner. The nearest-colattice algorithm: Time-approximation tradeoff for approx-cvp. *Open Book Series*, 4(1):251–266, 2020.

FBB+15. Robert Fitzpatrick, Christian Bischof, Johannes Buchmann, Özgür Dagdelen, Florian Göpfert, Artur Mariano, and Bo-Yin Yang. Tuning gausssieve for speed. In Diego F. Aranha and Alfred Menezes, editors, *Progress in Cryptology - LATINCRYPT 2014*, pages 288–305, Cham, 2015. Springer International Publishing.

GN08. Nicolas Gama and Phong Q. Nguyen. Predicting lattice reduction. In Nigel Smart, editor, *Advances in Cryptology – EUROCRYPT 2008*, pages 31–51, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

GNR10. Nicolas Gama, Phong Q. Nguyen, and Oded Regev. Lattice Enumeration Using Extreme Pruning. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110, pages 257–278. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

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

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

HP13. Md Haque and Josef Pieprzyk. Analysing progressive-bkz lattice reduction algorithm. 11 2013.

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

KKNM25. Alexander Karenin, Elena Kirshanova, Julian Nowakowski, and Alexander May. Fast slicer for batch-cvp: Making lattice hybrid attacks practical. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 100–132. Springer, 2025.

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

LG08. Alberto Leon-Garcia. *Probability, statistics, and random processes for electrical engineering*. Pearson/Prentice Hall, Upper Saddle River, NJ, 3. ed edition, 2008.

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

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

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

LN13. Mingjie Liu and Phong Q. Nguyen. Solving BDD by Enumeration: An Update. In Ed Dawson, editor, *Topics in Cryptology – CT-RSA 2013*, Lecture Notes in Computer Science, pages 293–309, Berlin, Heidelberg, 2013. Springer.

LN20. Jianwei Li and Phong Q. Nguyen. A complete analysis of the bkz lattice reduction algorithm. https://eprint.iacr.org/2020/1237, 2020.

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

MAT22. MATZOV. Report on the Security of LWE: Improved Dual Lattice Attack. https://doi.org/10.5281/zenodo.6412487, April 2022.

MKMS22. Jose Maria Bermudo Mera, Angshuman Karmakar, Tilen Marc, and Azam Soleimanian. Efficient lattice-based inner-product functional encryption. In Goichiro Hanaoka, Junji Shikata, and Yohei Watanabe, editors, *Public-Key Cryptography – PKC 2022*, pages 163–193, Cham, 2022. Springer International Publishing.

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

Ngu09. Phong Q. Nguyen. Hermite's Constant and Lattice Algorithms. In Phong Q. Nguyen and Brigitte Vallée, editors, *The LLL Algorithm*, pages 19–69. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009. Series Title: Information Security and Cryptography.

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

NV09. Phong Q. Nguyen and Brigitte Valle. *The LLL Algorithm: Survey and Applications*. Springer Publishing Company, Incorporated, 1st edition, 2009.

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

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

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

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

SS12. Claus-Peter Schnorr and Taras Shevchenko. Solving subset sum problems of densioty close to 1 by "randomized" bkz-reduction. *IACR Cryptol. ePrint Arch.*, page 620, 2012.

TKH18. Tadanori Teruya, Kenji Kashiwabara, and Goichiro Hanaoka. Fast lattice basis reduction suitable for massive parallelization and its application to

the shortest vector problem. In *IACR International Workshop on Public Key Cryptography*, pages 437–460. Springer, 2018.

Wan24. Leizhang Wang. Analyzing pump and jump bkz algorithm using dynamical systems. In Markku-Juhani Saarinen and Daniel Smith-Tone, editors, *Post-Quantum Cryptography*, pages 406–432, Cham, 2024. Springer Nature Switzerland.

WWW23. Leizhang Wang, Yuntao Wang, and Baocang Wang. A trade-off svp-solving strategy based on a sharper pnj-bkz simulator. In *Proceedings of the 2023 ACM Asia Conference on Computer and Communications Security*, ASIA CCS '23, page 664–677, New York, NY, USA, 2023. Association for Computing Machinery.

Xag10. Keita Xagawa. Cryptography with Lattices. https://xagawa.net/pdf/2010Thesis.pdf, 2010.

XWW+24. Wenwen Xia, Leizhang Wang, Geng Wang, Dawu Gu, and Baocang Wang. A Refined Hardness Estimation of LWE in Two-Step Mode. In Qiang Tang and Vanessa Teague, editors, *Public-Key Cryptography – PKC 2024*, pages 3–35, Cham, 2024. Springer Nature Switzerland.

YD18. Yang Yu and Léo Ducas. Second order statistical behavior of lll and bkz. In Carlisle Adams and Jan Camenisch, editors, *Selected Areas in Cryptography – SAC 2017*, pages 3–22, Cham, 2018. Springer International Publishing.

ZD26. Ziyu Zhao and Jintai Ding. Towards large-scale lattice attack: New lattice records by disk-based sieving. In *EUROCRYPT 2026*, pages 154–182. Springer, 2026.

ZDY25. Ziyu Zhao, Jintai Ding, and Bo-Yin Yang. Sieving with streaming memory access. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2025(2):362–384, 2025.

A Detailed Description of Algorithms

In this section, we provide a detailed description of some algorithms. First, Appendix A.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 Appendix A.2.

A.1 Detailed Description of the Refined SVPDimEst

We now give the algorithmic details of the probabilistic SVP-dimension estimator introduced in Section 6.2. Recall that $P_{\mathrm{suc}}(k;\mathsf{rr},\sigma) := \Pr\left[\sigma^2\chi_k^2 \le \mathrm{GH}\big(\mathsf{rr}[d-k:d)\big)^2\right]$. The estimator returns the smallest dimension k for which this probability reaches the prescribed threshold.

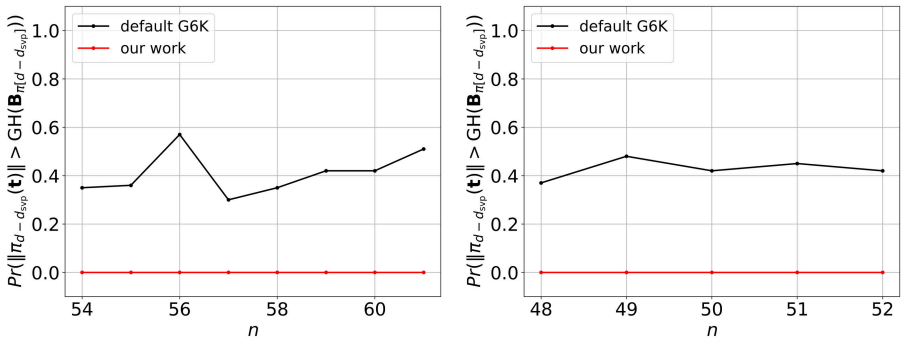
input : The log-GS profile rr, the deviation of the discrete Gaussian Distribution σ;
output: Estimated SVP dimension d_{svp};
Function `SVPDimEst(rr,σ)`**:**
  for $d_{\mathrm{svp}} \leftarrow d_{\mathrm{start}}$ **to** d **do**
    $P_{\mathrm{suc}}(d_{\mathrm{svp}};\mathsf{rr},\sigma) \leftarrow \Pr\left[x \leftarrow \sigma^2\chi^2_{d_{\mathrm{svp}}} : x \le \left(\mathrm{GH}(\mathsf{rr}_{[d-d_{\mathrm{svp}}:d)})\right)^2\right]$;
    if $P_{\mathrm{suc}}(d_{\mathrm{svp}};\mathsf{rr},\sigma) \ge 0.999$ **then**
      return d_{svp};

Algorithm 7: Dimension Estimation for the SVP call on solving LWE.

Figure 14 shows that even if $\sigma\sqrt{d_{\mathrm{svp}}} \le \mathrm{GH}(\mathbf{B}_{\pi[d-d_{\mathrm{svp}}:d)})$, $\|\pi_{d-d_{\mathrm{svp}}}(\mathbf{t})\|$ is possibly larger than $\mathrm{GH}(\mathbf{B}_{\pi[d-d_{\mathrm{svp}}:d)})$, 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 $\|\pi_{d-d_{\mathrm{svp}}}(\mathbf{t})\| \le \mathrm{GH}(\mathbf{B}_{\pi[d-d_{\mathrm{svp}}:d)})$ 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 ($\ge$0.999) for recovering the target vector with a new estimated dimension in Algorithm 7 to solve LWE problem.

Lemma 2 shows that improving basis quality cannot increase the dimension selected by the probabilistic estimator, or the corresponding cost under a non-decreasing Search-cost model.

Lemma 2. *Let* `SVPDimEst` *be Algorithm 7, using the same standard deviation* σ*, the same candidate dimensions* $k \in \{d_{\mathrm{start}},\cdots,d\}$*, and the same success threshold* p_{target} *for both inputs. Here* d_{start} *is the initial candidate dimension, and* $p_{\mathrm{target}} = 0.999$ *in the algorithm. Let* $\mathbf{C}$ *and* $\mathbf{D}$ *be bases of the same* d*-dimensional lattice with* $\mathsf{rr}_{\mathbf{D}} \succeq_{\mathcal{Q}} \mathsf{rr}_{\mathbf{C}}$*. Apply the same reduction tour* $\mathcal{R}$*-*ξ*, satisfying Property 1,*



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

Fig. 14: 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 Section 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 Algorithm 7 in our work, the condition $\|\pi_{d-d_{\mathrm{svp}}}(\mathbf{t})\| \leq \mathrm{GH}(\mathbf{B}_{\pi[d-d_{\mathrm{svp}},d]})$ was satisfied in all 100 tested instances.

to obtain $\mathbf{C}'$ *and* $\mathbf{D}'$*. If the estimator returns a feasible dimension for* $\mathsf{rr}_{\mathbf{C}'}$*, then it also returns a feasible dimension for* $\mathsf{rr}_{\mathbf{D}'}$*, and*

$$\mathit{SVPDimEst}(\mathsf{rr}_{\mathbf{D}'}, \sigma) \leq \mathit{SVPDimEst}(\mathsf{rr}_{\mathbf{C}'}, \sigma).$$

Consequently, if $T_{\mathcal{C}}$ *is nondecreasing in the dimension, the Search cost for* $\mathbf{D}'$ *is no greater than that for* $\mathbf{C}'$*.*

Proof. Property 1 gives $\mathsf{rr}_{\mathbf{D}'} \geq_{\mathcal{Q}} \mathsf{rr}_{\mathbf{C}'}$. Since both bases have the same total lattice volume, the prefix-volume inequalities imply

$$\mathrm{GH}(\mathbf{C}'_{\pi[d-k:d)}) \leq \mathrm{GH}(\mathbf{D}'_{\pi[d-k:d)}), \qquad 1 \leq k \leq d.$$

For each fixed k, the cumulative distribution function of $\sigma^2\chi^2_k$ is nondecreasing in its argument. Hence

$$P_{\mathrm{suc}}(k; \mathsf{rr}_{\mathbf{C}'}, \sigma) \leq P_{\mathrm{suc}}(k; \mathsf{rr}_{\mathbf{D}'}, \sigma).$$

Every candidate dimension reaching p_{target} for $\mathbf{C}'$ therefore also reaches it for $\mathbf{D}'$. Taking the smallest feasible dimension proves the dimension inequality. Applying the nondecreasing function $T_{\mathcal{C}}$ proves the cost comparison, including equality. The same argument applies directly to the input profiles before the common reduction tour and uses only their equal total volume and quality ordering. □

The probabilistic estimator gives the following instance of Theorem 2.

Theorem 3. *Assume the hypotheses of Theorem 2 other than its dimension-estimator condition. Let* `SVPDimEst` *be Algorithm 7, with a fixed initial candidate dimension* d_{start} *and success threshold* $p_{\mathrm{target}} = 0.999$*. Then PSSearch*

(Algorithm 3) terminates and returns a strategy minimizing T_{total} over the corresponding candidate space $\mathcal{S}_{\text{est}}$ under the fixed simulator, estimator, and cost model.

Proof. Lemma 2 verifies that the probabilistic estimator preserves feasibility and cannot increase the selected dimension when basis quality improves. This supplies the dimension-estimator condition of Theorem 2; all its remaining hypotheses hold by assumption. The conclusion follows. □

A.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 technique 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 A.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 to minimize the time cost of solving LWE challenges.

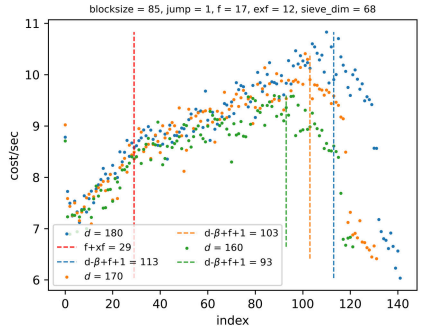
Additionally, when constructing the actual time-cost model by testing PnJBKZ on a specific machine, we find that each Pump in PnJBKZ has a different time cost, as shown in Figure 15. In particular, the time cost of the first Pump is higher than that of subsequent Pumps. It increases with the incremental index from 2nd 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 A.2.2, we propose a new time-cost model for PnJBKZ that more accurately reflects its time cost performance in practical applications.

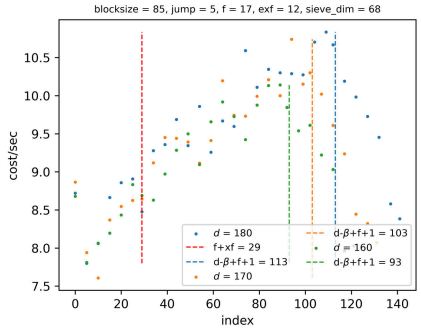
A.2.1 Practical Cost Model of Pump

We can regard T_{Pump} as the computational cost model for the d_{svp}-dimensional progressive sieve. T_{Pump} in [ADH+19]

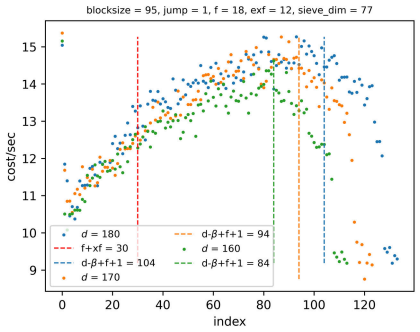
[6] While the dimension of sieving exceeds 128, the time cost for Pump fits the theoretical value well, and we can directly use the time-cost model of triple_gpu sieve described in [DSvW21].



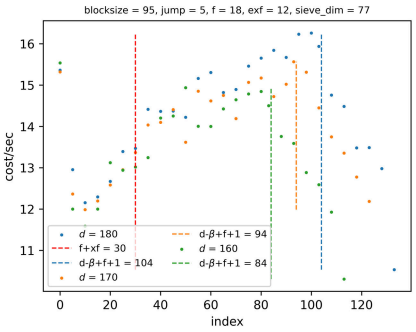
(a) PnJBKZ(85,1)



(b) PnJBKZ(85,5)



(c) PnJBKZ(95,1)



(d) PnJBKZ(95,5)

Fig. 15: 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.

is considered as

$$T_{\text{Pump}}(d_{\text{svp}}) = 2\sum_{j=\beta_0}^{d_{\text{svp}}} T_{\text{sieve}}(j) = 2\sum_{j=\beta_0}^{d_{\text{svp}}} 2^{c\cdot j+o(j)} \approx 2\cdot\frac{2^{c(d_{\text{svp}}+1)}-2^{c\beta_0}}{2^c-1}\cdot 2^{o(d_{\text{svp}})} \approx 2^{cd_{\text{svp}}+c_1}, \tag{20}$$

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 16, we observe that R^2 is close to 1, indicating a good fitting effect. Figure 16 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.

A.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 Figure 7 in [DSvW21] 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 Figure 17) through the curve fitting method.

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

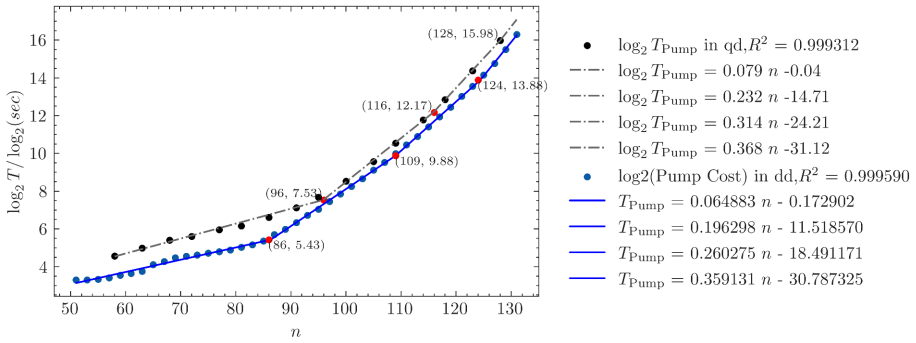


Fig. 16: Pump Cost Figure while $d = 180$, Sieve used in Pump is gpu_sieve, and it's running on Machine C with 2 GPUs and 32 threads: Relation between $\log_2(T_{\text{Pump}})$ and sieve dimension n.

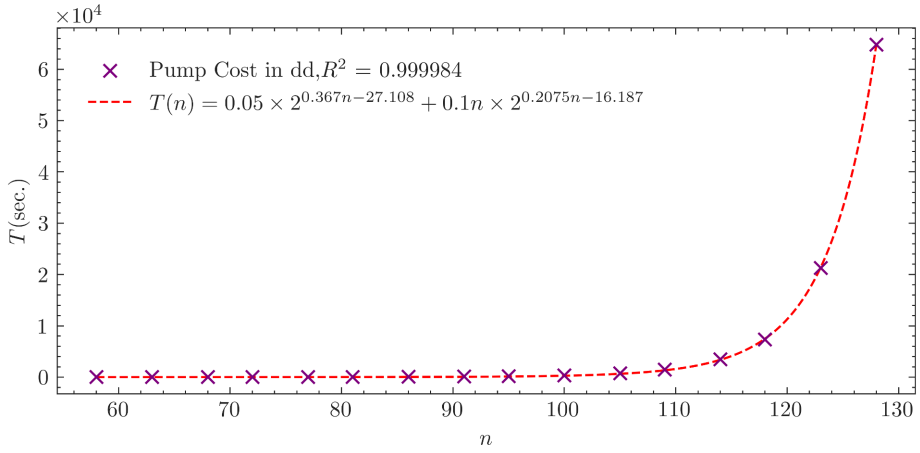


Fig. 17: Pump cost model considers the cost to generate the hash value: The vertical axis represents T_{Pump} and the horizontal axis represents the sieve dimension n.

A.2.2 Practical Cost Model of PnJBKZ PnJBKZ consists of a series of Pumps. Under the simplified assumption that all Pumps have equal time cost, its computational cost can be approximated by the cost of $\frac{d+2f-\beta}{J}$ progressive sieves on the $(\beta - f)$-dimensional projected sublattice with jump J.

However, as shown in Figure 15, the Pump cost varies across calls: after the first Pump, it initially increases and then decreases. The growth rate also differs between the preceding and middle portions of the tour. To describe these portions, we use f_{extra} for the extra dimension-for-free value in G6K, set to 12 in the default setting to enhance PnJBKZ's efficiency.

We index Pump calls by their zero-based positions in a tour. For $J = 1$, we divide the PnJBKZ cost into four parts: the first Pump at index 0, the preceding calls in $[1, f + f_{\text{extra}})$, the middle calls in $[f + f_{\text{extra}}, d - \beta + f + 1)$, and the later calls in $[d - \beta + f + 1, d - \beta + 2f + f_{\text{extra}} + 1)$. Let T_{first}, T_{pre}, T_{mid}, and T_{later} denote the corresponding costs at $J = 1$. In the middle range, we fit the cost of the ith Pump by $T_i = A \cdot i + B$, so its first and last indices are $f + f_{\text{extra}}$ and $d - \beta + f$, respectively. The resulting middle-range cost is

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

We measured T_{first}, T_{pre}, and T_{later} and fitted the coefficients A and B in dimension $d = 180$, with jump $J = 1$ and the "dd" float type. The simulated cost model is shown in Figure 18. For jump J, we scale the four baseline components as follows:

$$\begin{aligned} T_{\text{PnJBKZ}}(d, \beta, J, f, f_{\text{extra}}) = & T_{\text{first}} + T_{\text{pre}} \cdot \frac{\lceil \frac{f+f_{\text{extra}}}{J} \rceil - 1}{f + f_{\text{extra}} - 1} \\ & + T_{\text{mid}}(d, \beta, f, f_{\text{extra}}) \cdot \frac{\lceil \frac{d-\beta-f_{\text{extra}}}{J} \rceil + 1}{d - \beta - f_{\text{extra}} + 1} \\ & + T_{\text{later}} \cdot \frac{\lceil \frac{f+f_{\text{extra}}}{J} \rceil}{f + f_{\text{extra}}}, \end{aligned} \tag{21}$$

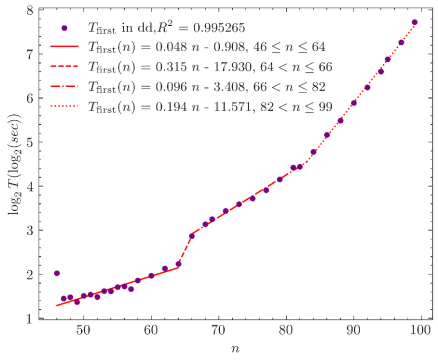
where f is the dimension for free value of β.

We have also used Eq. 21 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 19.

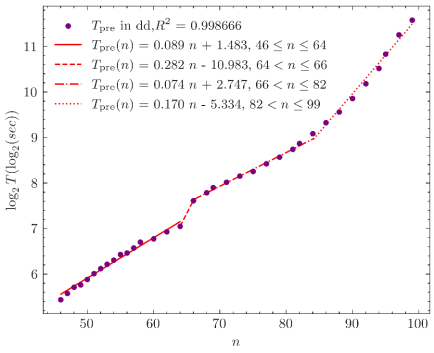
B Blocksize and Jump Strategy Selection based on ProBKZ

The *blocksize and jump strategy selection algorithm based on ProBKZ* (BSSA, Fig. 20) applies the *Shortest Path Algorithm* to strategy selection.

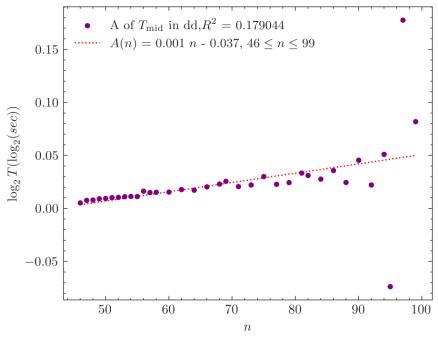
BSSA takes as input the same LLL-reduced lattice basis as PSSearch, represented by its initial log-GS profile rr_0. It tries to find the shortest path from



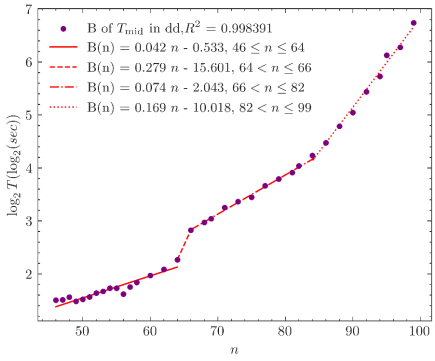
(a) T_{first}



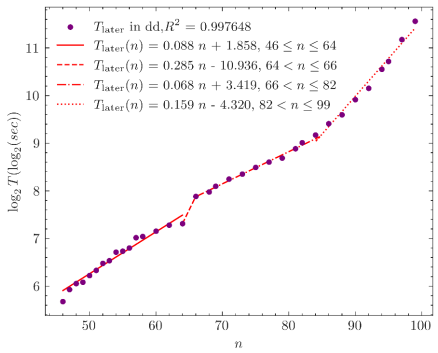
(b) T_{pre}



(c) A of T_{mid}

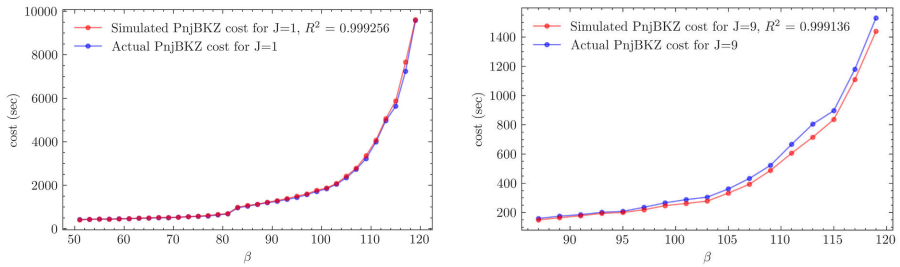


(d) B of T_{mid}



(e) T_{later}

Fig. 18: 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. 19: Simulate each PnJBKZ Cost using Eq. (21) 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.

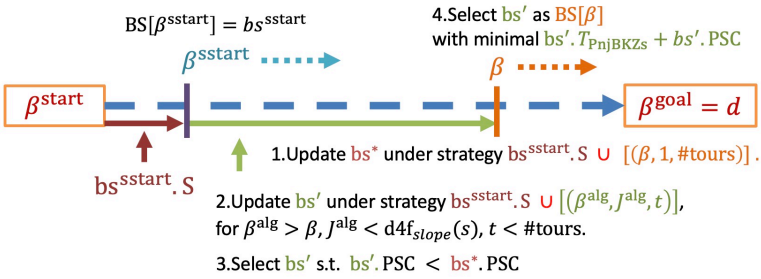


Fig. 20: BSSA Process.

BKZ-β_{start} to BKZ-β_{goal} reduced lattice basis by setting several middle nodes (such as $\beta^{\text{sstart}} = \beta_i$, for $\beta^{\text{start}} < \beta_i < \beta^{\text{goal}}$) from β^{start} to β^{goal} as a measure of basis quality. For edges between nodes β_i and β_j, BSSA determines the tuple $(\beta^{\text{alg}}, J^{\text{alg}}, t)$ that minimizes the reduction time cost $T_{\mathcal{R}s}$ 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 $\mathsf{BS}[\beta^{\text{goal}}]$, in which the key is each middle node β_i and the value is a tuple of $\mathsf{bs} = (\mathsf{rr}, \mathsf{S}, T_{\mathcal{R}s}, T_{\mathcal{C}})$, where rr is the log-GS profile of a fully BKZ-β^{goal} reduced basis, S means the blocksize and jump selection strategy which will improve the quality of lattice basis from fully BKZ-β^{start} reduced to fully BKZ-β^{goal} reduced, which is the combination of $(\beta^{\text{alg}}, J^{\text{alg}}, t, T_{\mathcal{R}s})$ stored on each edge in the shortest path from node β^{start} to β^{goal} with respect to the sum of simulated reduction cost $T_{\mathcal{R}s} = \sum_{\beta^{\text{alg}}, J^{\text{alg}}, t} T_{\mathcal{R}s}(\beta^{\text{alg}}, J^{\text{alg}}, t)$. BSSA uses a shortest-path procedure equivalent to Dijkstra's algorithm on the prescribed graph. $T_{\mathcal{C}}$ is the time cost of the terminal SVP oracle at the dimension returned by `SVPDimEst` (Algorithm 7), which means the estimated search step time cost for LWE to be solved 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 reduction time cost $T_{\mathcal{R}s}$ of improving the lattice basis quality via a series of PnJBKZ$(\beta, J) \in \mathsf{S}$ and the search time cost $T_{\mathcal{C}}$ of the final SVP oracle. For more details on BSSA, see Algorithm 8 .

Since BSSA uses the quality of the BKZ-β fully reduced basis as the intermediate node to find the optimized block strategy, the BSSA strategy with more flexible intermediate node settings can be constructed by the slope of the logarithmic Gram-Schmidt norms or combining the root Hermite factor value δ and Geometric Series Assumption. We plan to finish this work in the future.

C Experiments

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

C.1 PnJBKZ simulator Accuracy Verification Experiments

In this section, we first demonstrate in C.1.1 that the PnJBKZ simulator can accurately predict the reduction effect of PnJBKZ in the ideal case, where each

input : rr_0, σ, $\beta^{\text{start}} \leftarrow 50$ and an admissible jump-bound function $J_{\max}(\cdot)$;
output: $T_{\min}$, $S_{\min}$;

Function `BSSA`(rr_0, σ, $\beta^{\text{start}} \leftarrow 50$)**:**
  $d \leftarrow \text{len}(\mathsf{rr}_0)$; $d^{(0)}_{\text{svp}} \leftarrow$ `SVPDimEst`(rr_0, σ); $\mathsf{BS}[\beta^{\text{start}}] = (\mathsf{rr}_0, [\,], 0, T_{\mathcal{C}}(d^{(0)}_{\text{svp}}))$;
  for $\beta \leftarrow \beta^{\text{start}}$ **to** d **do**
    $T^{(\min)}_{\mathcal{R}s} \leftarrow +\infty$;
    for $\beta^{\text{sstart}} \leftarrow \beta^{\text{start}}$ **to** $\beta - 1$ **do**
      $\mathsf{bs}^{\text{sstart}} \leftarrow \mathsf{BS}\,[\beta^{\text{sstart}}]$; $\mathsf{bs} \leftarrow (\emptyset, \emptyset, +\infty, +\infty)$;
      Update bs^* under strategy $\mathsf{bs}^{\text{sstart}}.\mathsf{S} \| [(\beta, 1, \sharp\text{tours}(\mathsf{bs}^{\text{sstart}}.\mathsf{rr}, \text{BKZ-}\beta))]$;
      for $\beta^{\text{alg}} \leftarrow \beta + 1$ **to** d **do**
        for $j \leftarrow J_{\max}(\beta^{\text{alg}})$ **to** 1 **do**
          $T' \leftarrow +\infty$;
          for $t \leftarrow 1$ **to** $\sharp\text{tours}(\mathsf{bs}^{\text{sstart}}.\mathsf{rr}, \text{PnJBKZ-}(\beta^{\text{alg}}, j))$ **do**
            Update bs' under strategy $\mathsf{bs}^{\text{sstart}}.\mathsf{S} \| [(\beta^{\text{alg}}, j, t)]$;
            if $\mathsf{bs}'.T_{\mathcal{C}} < \mathsf{bs}^*.T_{\mathcal{C}}$ **then**
              $T' \leftarrow \mathsf{bs}'.T_{\mathcal{R}s}$;
              break;
          if $\mathsf{bs}.T_{\mathcal{R}s} > T'$ **then**
            $\mathsf{bs} \leftarrow \mathsf{bs}'$;
      if $T^{(\min)}_{\mathcal{R}s} > \mathsf{bs}.T_{\mathcal{R}s}$ **then**
        $T^{(\min)}_{\mathcal{R}s} \leftarrow \mathsf{bs}.T_{\mathcal{R}s}$; $\mathsf{BS}\,[\beta] \leftarrow \mathsf{bs}$;
  $\mathsf{bs}_{\min} \leftarrow \min_{\mathsf{bs}.T_{\mathcal{R}s} + \mathsf{bs}.T_{\mathcal{C}}} \mathsf{BS}$;
  return $T_{\min} \leftarrow \mathsf{bs}_{\min}.T_{\mathcal{R}s} + \mathsf{bs}_{\min}.T_{\mathcal{C}}$, $\mathsf{S}_{\min} \leftarrow \mathsf{bs}_{\min}.\mathsf{S}$;

Algorithm 8: BSSA

pump used by PnJBKZ outputs a HKZ-reduced lattice basis. Then, Section 5.2 and C.1.2 illustrate how to simulate PnJBKZ using the `d4f` technique in practice. Finally, C.1.3 and C.1.4 present additional accuracy verification experiments for the PnJBKZ simulator when employing the `d4f` technique in practice.

C.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 $\beta - \texttt{d4f}(\beta)$ dimensional progressive sieving during the Pump-up stage, and can thus perform at most $\beta - \texttt{d4f}(\beta)$ embeddings during the Pump-down stage. Consequently, the output basis of each β-dimensional Pump will inevitably include $\texttt{d4f}(\beta)$-dimensional GS vectors that do not meet the HKZ-reduced basis properties. Thus, in this subsection C.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, LLL reduction is applied after each Pump step during a PnJBKZ-(β, J) tour, and the experimental GS profiles exhibit a smooth downward trend. To approximate the effect of LLL reduction on the predicted profile, we apply the empirical smoothing correction to the simulated log-GS values described in Section 5.1.1.

We calculate $\mathrm{Sim}(l_i'')$ from the properties of the HKZ-reduced basis, together with the empirical correction described above. To assess prediction accuracy, we use the ratio $\frac{\mathrm{Real}(\|\mathbf{b}_i^*\|)}{\mathrm{Sim}(\|\mathbf{b}_i^*\|)}$ 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 Figure 21, which shows the prediction accuracy of the PnJBKZ simulator under different jump values. Here, $\mathrm{Real}(\|\mathbf{b}_i^*\|)$ represents the geometric mean of the observed Gram–Schmidt vector lengths over 10 independent reduction experiments with the same parameters (β, J, tours), and $\mathrm{Sim}(\|\mathbf{b}_i^*\|)$ is the simulated Gram-Schmidt vector lengths.

Section 4.3 and Figure 1 of BKZ 2.0 [CN11] compare the norms returned by local enumeration with the Gaussian heuristic for the corresponding local blocks; for intermediate block indices, most of those ratios lie in $[0.95, 1.05]$. Our experiment evaluates the GS-norm profile predicted after each PnJBKZ tour using the Real/Sim ratios defined in Section 5. Figure 21 reports these ratios for the ideal-Pump configurations. The two evaluations concern different

quantities, so these figures alone do not establish a ranking of the two simulators by prediction accuracy.

C.1.2 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.

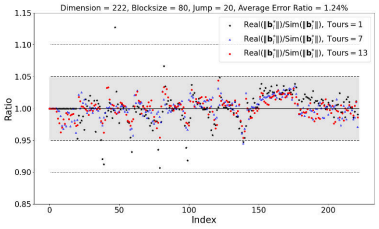
In this part, our experiments were tested on the TU Darmstadt LWE Challenge lattice basis with parameter (n=60, α=0.010). Before running the PnJBKZ simulator, we performed a small block reduction to remove the influence of q-ary vectors in the LWE Challenge lattice basis. After this pre-processing, we 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 $\frac{\text{Real}(\|\mathbf{b}_i^*\|)}{\text{Sim}(\|\mathbf{b}_i^*\|)}$ for $i \in [0, d-1]$ in each tour of PnJBKZ's reduction, see Figure 22. Here, $\text{Real}(\|\mathbf{b}_i^*\|)$ represents the geometric mean of the observed Gram–Schmidt vector lengths over 20 independent reduction experiments using the same reduction parameters (β, J, tours). Meanwhile, $\text{Sim}(\|\mathbf{b}_i^*\|)$ denotes the simulated Gram-Schmidt vector lengths.

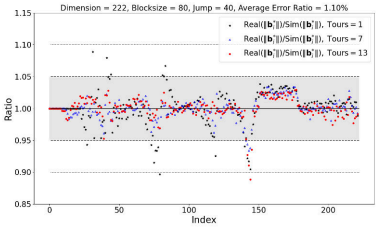
We calculate $\text{Sim}(l_i'')$ strictly based on the properties of the HKZ-reduced basis and the Gaussian Heuristic. Therefore, in addition to serving as a criterion for evaluating the accuracy of the PnJBKZ simulator, the ratio $\frac{\text{Real}(\|\mathbf{b}_i^*\|)}{\text{Sim}(\|\mathbf{b}_i^*\|)}$ can also be used to assess whether Heuristic 2 holds. In particular, as shown in Figure 22, 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: $\texttt{d4f}_{\text{slope}}(s = -0.0248) \approx 11.6 \leq 12$. For the four tested parameter configurations satisfying $\texttt{jump} \leq \lceil \texttt{d4f}_{\text{slope}}(s = -0.0248) \rceil = 12$, we recorded five tours for $(\beta, J) = (100, 1)$ and thirteen tours for each of the other three configurations. Across all 9,468 recorded GS-norm ratios, 67.15% lie in $[0.95, 1.05]$, and 98.94% lie in $[0.90, 1.10]$. The mean absolute deviation of these ratios from 1 is 0.0587. Figure 22 demonstrates that, based on the refined d4f estimation, our PnJBKZ simulator accurately predicts the reduction effect of PnJBKZ. Furthermore, Heuristic 2 holds when $J \leq \texttt{d4f}_{\text{slope}}(s)$.

For the experiments with optimistic `d4f` settings, we use the same empirical smoothing correction as in the ideal case to approximate the effect of LLL reduction on the simulated log-GS profile; see Section 5.1.1. All subsequent experiments use this empirically corrected simulator, and we omit further explicit mention of the correction.

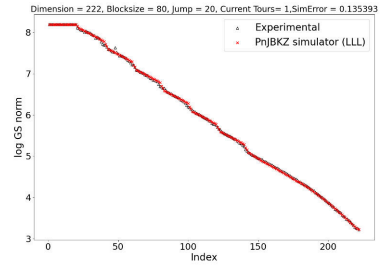
Moreover, this is further verified by (e) and (f) in Figure 23 when the tours increase to 13. Meanwhile, the PnJBKZ simulator uses Eq. (4) as an approximate estimate for the actual value l_i''. This estimate effectively reflects how the average norms of Gram-Schmidt vectors change during each tour's reduction of PnJBKZ(β, J), which uses the optimistic `d4f` setting in practice.



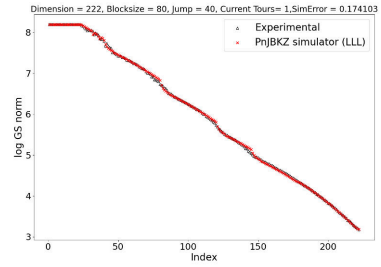
(a) β=80, jump=20



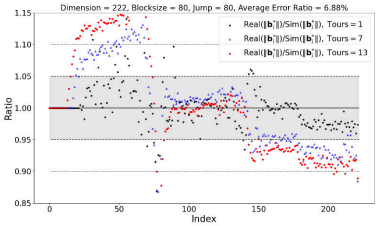
(b) β=80, jump=40



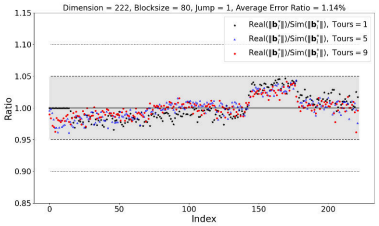
(c) β=80, jump=20



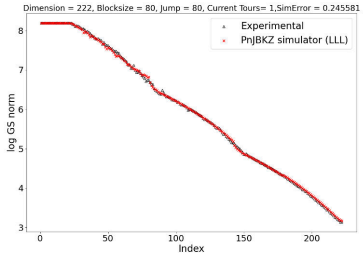
(d) β=80, jump=40



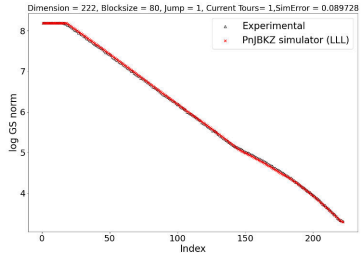
(e) β = 80, jump=80



(f) β=80, jump=1

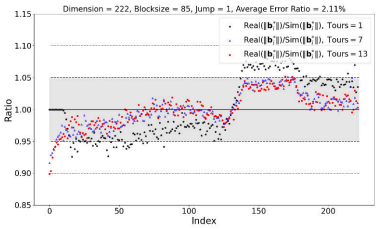


(g) β=80, jump=80

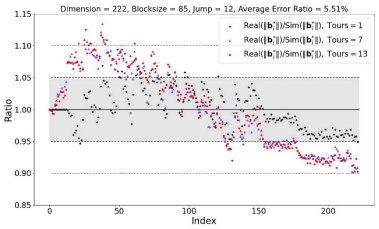


(h) β=80, jump=1

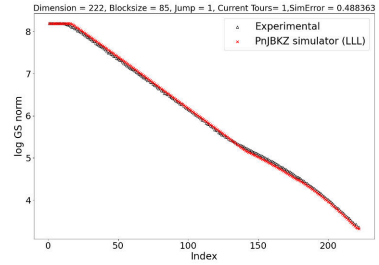
Fig. 21: Ratio $\frac{\text{Real}(\|\mathbf{b}_i^*\|)}{\text{Sim}(\|\mathbf{b}_i^*\|)}$. and corresponding overall prediction effect of PnJBKZ simulator. Run PnJBKZ(β, J) reduction on a 222-dimension LWE lattice basis (n=60,α=0.010) and record the ratio values. We test 10 times for each reduction parameter.



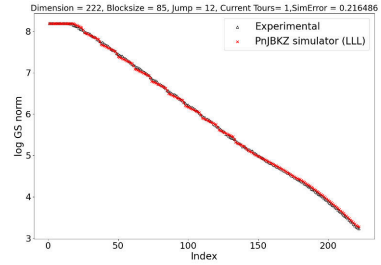
(a) β=85, jump=1



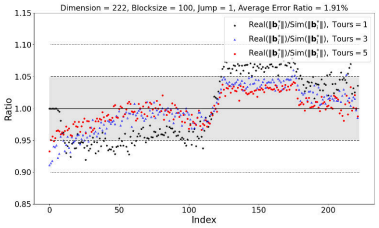
(b) β=85, jump=12



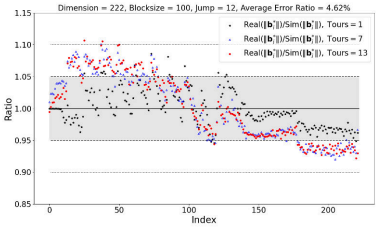
(c) β=85, jump=1



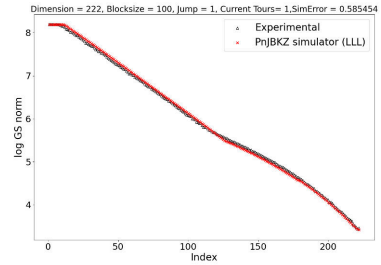
(d) β=85, jump=12



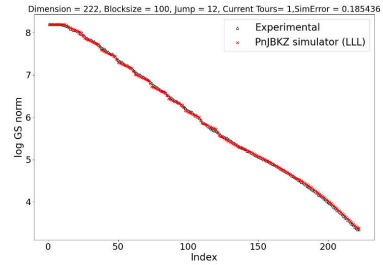
(e) β=100, jump=1



(f) β=100, jump=12



(g) β=100, jump=1



(h) β=100, jump=12

Fig. 22: Ratio $\frac{\mathrm{Real}(\|\mathbf{b}_i^*\|)}{\mathrm{Sim}(\|\mathbf{b}_i^*\|)}$. 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.

We define the prediction error using the logarithmic Gram-Schmidt lengths as

$$\text{SimError}(\sharp\text{tours}) = \sum_{i=0}^{d-1} \left(l''_{i(\sharp\text{tours})} - \text{Sim}(l''_i)_{(\sharp\text{tours})} \right)^2,$$

where $\sharp$tours represents the number of current tours, $l''_{i(\sharp\text{tours})}$ is the actual logarithmic GS value, and $\text{Sim}(l''_i)_{(\sharp\text{tours})}$ is the simulated logarithmic GS value predicted by the PnJBKZ simulator with $\sharp$tours. Figure 23 illustrates that the overall prediction error of the PnJBKZ simulator for different jumps is similar to that observed for jump=1.

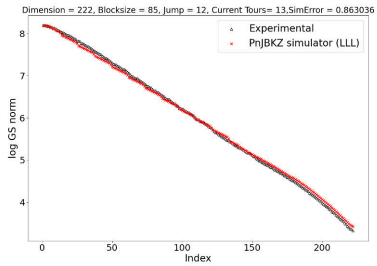
More verification experiment results with different reduction parameters on various lattice bases can be found in Appendix C.1.3.

In addition, our experiments, detailed in Table 3 and Appendix C.1.4, 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 purposes.

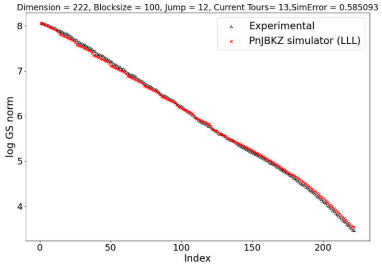
C.1.3 More experimental details about PnJBKZ simulator for predicting PnJBKZ with optimistic `d4f` In this section, we present additional verification experiments of our PnJBKZ simulator for predicting the reduction effect of the PnJBKZ algorithm, which uses optimistic `d4f` setting. These experiments 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 preprocessing, we obtain LWE challenge lattice basis $(n = 70, \alpha = 0.005)$ having a slope of $-0.04921/2$. The corresponding $\texttt{d4f}_{\text{slope}}(s)$ values range from 11.7 to 13.7. Since we need the maximum jump value $J \leq \texttt{d4f}_{\text{slope}}(s)$ to ensure the accuracy of the PnJBKZ simulator (see Section 5 for details), we set the maximum jump value $J \leq 12$ in these experiments.

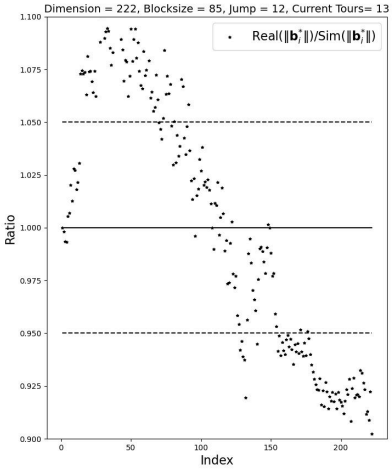
We present results on two preprocessed LWE Challenge lattice bases. For $(n, \alpha) = (75, 0.005)$, the displayed experiments use $\beta \in \{75, 80, 85, 95, 100\}$, with $J \in [1, 12]$, and the results are shown in Figures 24∼25. For $(n, \alpha) = (70, 0.005)$, the displayed experiments use $\beta \in \{60, 65, 70, 90\}$, with $J \in [1, 12]$, and the results are shown in Figure 26 ∼28. The verification results indicate that the PnJBKZ simulator performs well over the tested block-size ranges specified above and for $J \in [1, 12]$ on both LWE Challenge lattice bases.



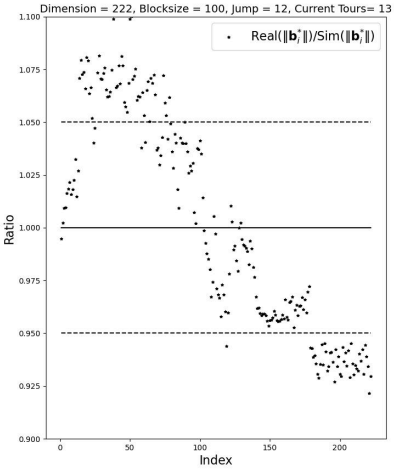
(a) $\beta = 85, \text{jump} = 12, \sharp\text{tours} = 13$



(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. 23: Overall Prediction effect of PnJBKZ simulator. Ratio $\frac{\text{Real}(\|\mathbf{b}_i^*\|)}{\text{Sim}(\|\mathbf{b}_i^*\|)}$. 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 $\lceil \texttt{d4f}_{\text{slope}}(s) \rceil$=12. We test also 20 times for each reduction parameter and show the average value of the experiments.

Figures 24 $\sim$ 28 show that, for different lattice bases with different reduction parameters, as long as the jump $\leq \texttt{d4f}_{\text{slope}}(s)$, even when the number of tours increase to 13, overall, the simulated profiles closely match the observed profiles. Most of the ratios $\frac{\text{Real}(\|\mathbf{b}_i^*\|)}{\text{Sim}(\|\mathbf{b}_i^*\|)}$ 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 provide empirical support for Heuristic 2 under the tested parameters. The PnJBKZ simulator predicts the evolution of the geometric-mean GS-norm profile by exponentiating the simulated log-GS values in Eq. (4). Across the two LWE instances $(n, \alpha) = (70, 0.005)$ and $(75, 0.005)$, the 16 tested parameter configurations and their recorded tours yield 50,648 GS-norm ratios. Of these, 82.11% lie in $[0.95, 1.05]$, and 99.04% lie in $[0.90, 1.10]$, with a mean absolute deviation from 1 of 0.0274.

Additionally, Table 3 and Appendix C.1.4 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. Using the rounded slope values reported in these tables, the relative slope prediction error, defined as the absolute difference between the simulated and observed slopes divided by the absolute observed slope, does not exceed 7.26% over the 21 reported reduction tours. 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.

Additional reproducibility material for these experiments is available in the `PnJBKZ_Simulator` directory of our artifact repository.[7] In the accompanying supplementary package, the experiment code is in `Figures_21-28_Simulator_Verification/PnJBKZ_Simulator`. To regenerate the ratio and profile-comparison figures, select the corresponding experiment parameters in the script's configuration block and run the following script from that directory:

```
Figures_21-28_PnJBKZ_Simulator_Verification.py
```

The directory `gs-lengths-simulator` contains additional observed and simulated log-GS profiles, real-to-simulated norm ratios, and comparison results for other lattice bases and for different choices of the number of tours, block sizes, and jump parameters.

C.1.3.1 LWE challenge lattice basis ($n = 75, \alpha = 0.005$).
Figure 24 $\sim$ Figure 25.

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

C.1.3.2 LWE challenge lattice basis ($n = 70, \alpha = 0.005$).
Figure 26 $\sim$ Figure 28.

C.1.4 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 three LWE Challenges using the "dd" floating-point type in Tables 6, 7 and Table 8. 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 6, 7 and Table 8 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.

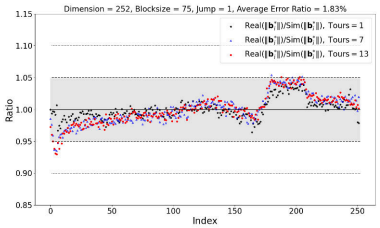
Table 8: Quality and log(walltime) ($\log_2(T)$ in seconds) during reduction of LWE Challenge $(n, \alpha) = (40, 0.025)$.

(β, J)	Simulation		Practical	
	Slope	$\log_2(T)$	Slope	$\log_2(T)$
(77,8)	-0.0281	6.5	-0.0262	6.7
(81,10)	-0.0249	6.2	-0.0239	6.5
(102,11)	-0.0217	7.5	-0.0214	7.8
(102,11)	-0.0205	7.5	-0.0208	7.9

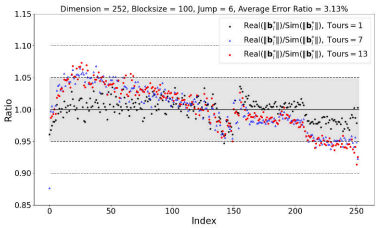
C.2 Experiments and Application to LWE

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 PSSearch (Algorithm 3).

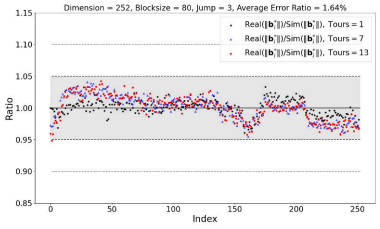
C.2.1 Efficiency of MinTwoStepSolver for solving LWE The default LWE solving algorithm in G6K is the script `lwe_challenge.py` in the implementation of G6K-GPU-Tensor [DSvW21], which uses a heuristic reduction strategy.



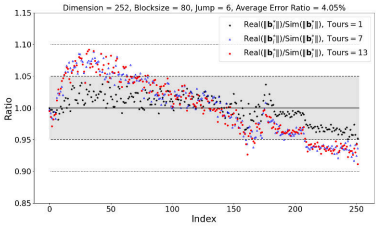
(a) β=75, jump=1



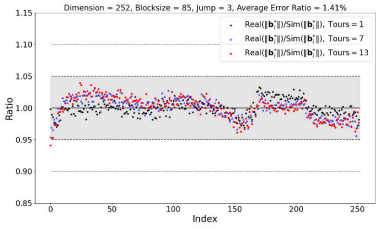
(b) β=100, jump=6



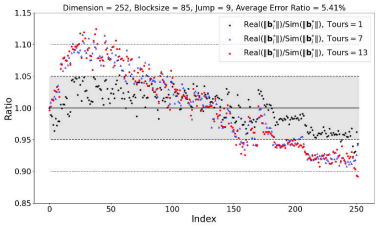
(c) β=80, jump=3



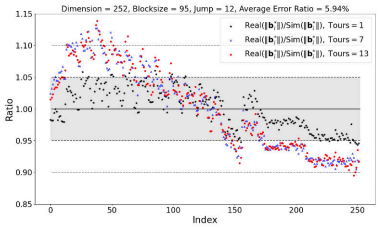
(d) β=80, jump=6



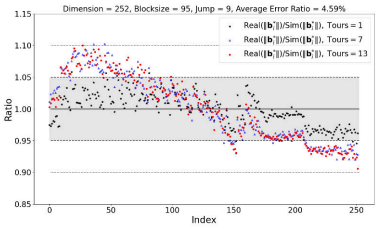
(e) β=85, jump=3



(f) β=85, jump=9

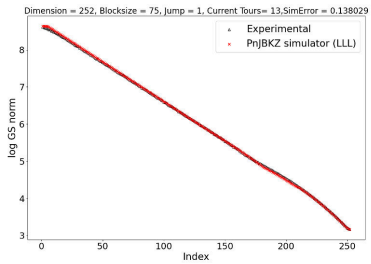


(g) β=95, jump=12

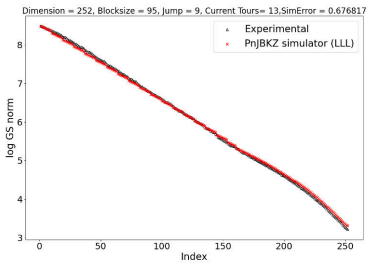


(h) β=95, jump=9

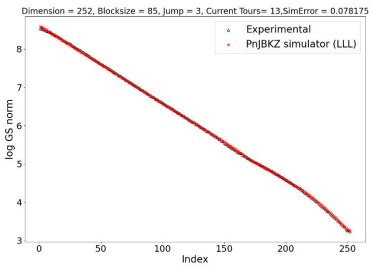
Fig. 24: Ratio $\frac{\mathrm{Real}(\|\mathbf{b}_i^*\|)}{\mathrm{Sim}(\|\mathbf{b}_i^*\|)}$. Run 13 tours of PnJBKZ$(\beta, J)$ reduction on a 252-dimension LWE lattice basis $(n = 75, \alpha = 0.005)$, and record the ratio values. We test 20 times for each reduction-parameter setting.



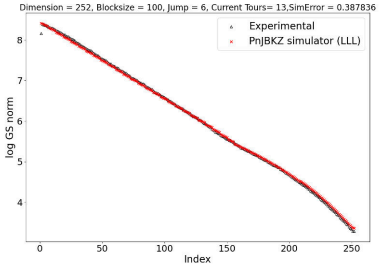
(a) $\beta = 75, \text{jump} = 1$



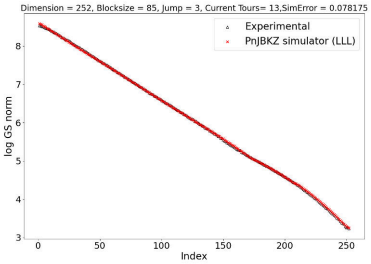
(b) $\beta = 95, \text{jump} = 9$



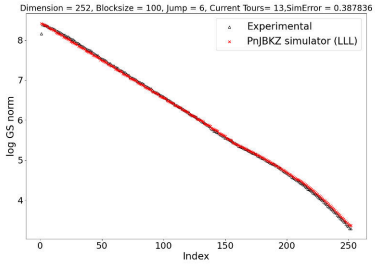
(c) $\beta = 85, \text{jump} = 3$



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

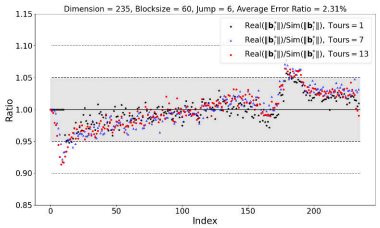


(e) $\beta = 85, \text{jump} = 3$

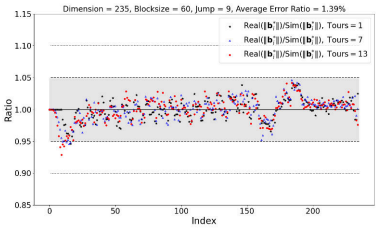


(f) $\beta = 100, \text{jump} = 6$

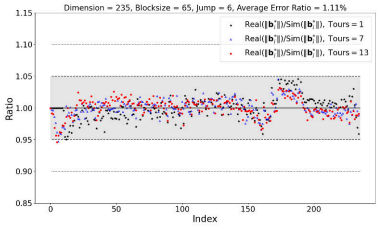
Fig. 25: Overall prediction effect of the PnJBKZ simulator. The figure illustrates the real and simulated basis profiles at the 13th tour of the PnJBKZ(β, J) reduction on a 252-dimensional LWE lattice basis ($n = 75, \alpha = 0.005$). Results are averaged over 20 independent trials for each parameter setting.



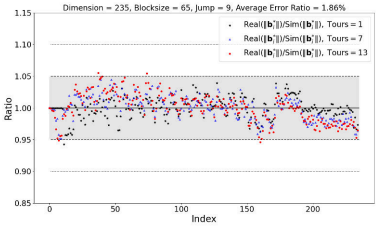
(a) $\beta = 60, \text{jump} = 6$



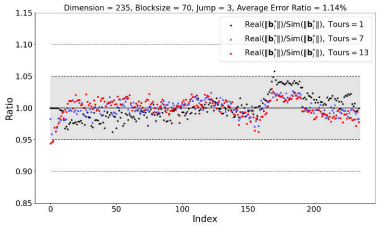
(b) $\beta = 60, \text{jump} = 9$



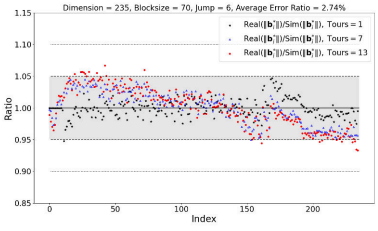
(c) $\beta = 65, \text{jump} = 6$



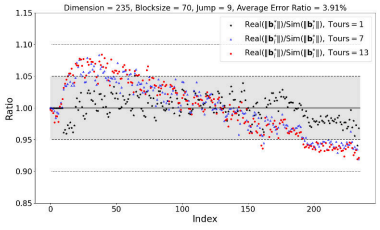
(d) $\beta = 65, \text{jump} = 9$



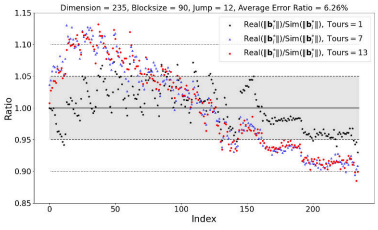
(e) $\beta = 70, \text{jump} = 3$



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

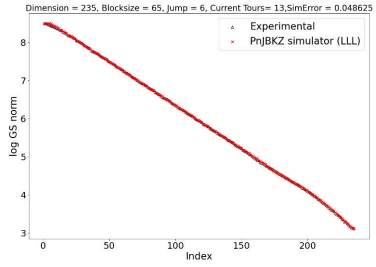


(g) $\beta = 70, \text{jump} = 9$

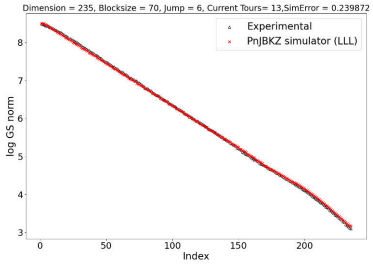


(h) $\beta = 90, \text{jump} = 12$

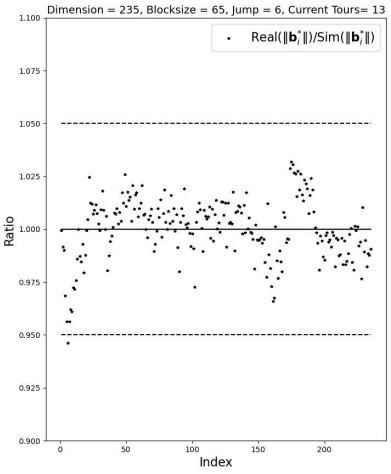
Fig. 26: Ratio $\frac{\text{Real}(\|\mathbf{b}_i^*\|)}{\text{Sim}(\|\mathbf{b}_i^*\|)}$. Run 13 tours of PnJBKZ$(\beta, J)$ reduction on a 235-dimension LWE lattice basis $(n = 70, \alpha = 0.005)$, and record the ratio values. We test 20 times for each reduction parameter.



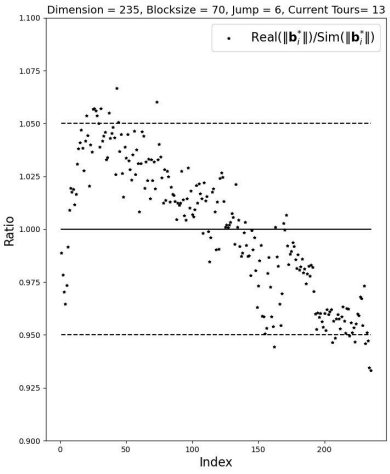
(a) $\beta = 65, \text{jump} = 6, \sharp\text{tours} = 13$



(b) $\beta = 70, \text{jump} = 6, \sharp\text{tours} = 13$

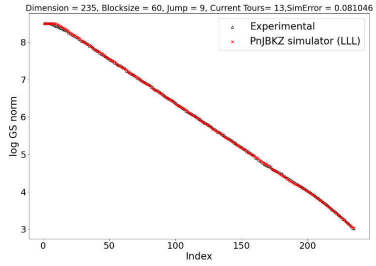


(c) $\beta = 65, \text{jump} = 6$

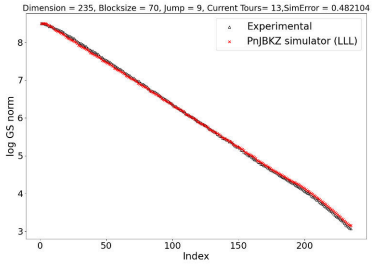


(d) $\beta = 70, \text{jump} = 6$

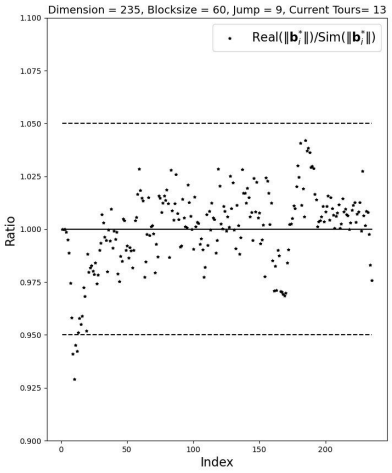
Fig. 27: Simulation accuracy of the PnJBKZ(β, J) simulator on a 235-dimensional LWE lattice basis ($n = 70, \alpha = 0.005$) after 13 tours. Subfigures (a) and (b) compare the logarithmic Gram-Schmidt norms of the real and simulated bases, while (c) and (d) detail the prediction ratio $\frac{\text{Real}(\|\mathbf{b}_i^*\|)}{\text{Sim}(\|\mathbf{b}_i^*\|)}$ for different values of β with $J = 6$. Results are averaged over 20 independent trials for each parameter setting.



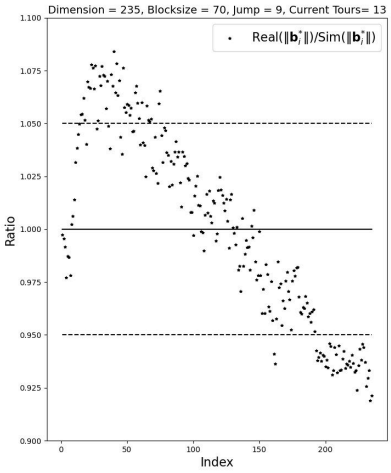
(a) $\beta = 60, \text{jump} = 9$



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



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



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

Fig. 28: Ratio $\frac{\text{Real}(\|\mathbf{b}_i^*\|)}{\text{Sim}(\|\mathbf{b}_i^*\|)}$. Run 13 tours of PnJBKZ$(\beta, J)$ 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.

(β, J)	Simulation		Practical	
	Slope	$\log_2(T)$	Slope	$\log_2(T)$
(70,8)	-0.0288	6.4	-0.0276	6.6
(80,10)	-0.0256	6.4	-0.0247	6.7
(102,11)	-0.0221	7.7	-0.0218	8.0
(102,11)	-0.0207	7.7	-0.0209	8.0
(103,11)	-0.0202	7.8	-0.0205	8.1

Table 6: Quality and wall time (T in seconds) during reduction of LWE Challenge $(n, \alpha) = (45, 0.020)$.

(β, J)	Simulation		Practical	
	Slope	$\log_2(T)$	Slope	$\log_2(T)$
(56,8)	-0.0307	6.2	-0.0295	6.5
(66,9)	-0.0279	6.2	-0.0272	6.4
(80,10)	-0.0254	6.5	-0.0249	6.8
(81,10)	-0.0238	6.6	-0.0237	6.9
(102,11)	-0.0215	7.8	-0.0216	8.2
(102,11)	-0.0205	7.8	-0.0208	8.2

Table 7: Quality and log(walltime) ($\log_2(T)$, where T is measured in seconds) during reduction of LWE Challenge $(n, \alpha) = (50, 0.015)$.

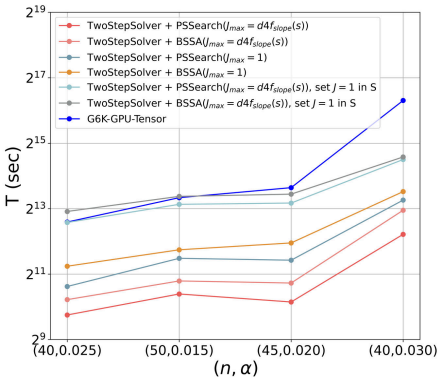
For more detail about the default LWE solving algorithm in G6K-GPU-Tensor, refer to the script at [4]. Figure 29 presents the experimental result of different LWE-solving algorithms. We use the practical cost model proposed in Appendix A.2 as $\mathcal{T}$. The blue curves and scatter points in Figure 29 represent the experimental time or memory cost of the default G6K strategy. The remaining curves correspond to TwoStepSolver instantiated with strategies generated by PSSearch or BSSA; see Section 4.2 and Appendix B, respectively.

For the LWE Challenge comparisons in Figure 29(a) and Table 4, the strategies selected by PSSearch achieve walltime speedups of approximately 7.2–23.4 over the default G6K strategy. The four instances in the figure give speedups of approximately 7.2–17.0, while the 23.4-fold speedup comes from the $(n, \alpha) = (40, 0.035)$ comparison in the table ($1180/50.4 \approx 23.4$). For the random LWE instances with $\alpha = 0.010$ in Figure 29(c), the corresponding speedups range from 3.71 to 9.81. These comparisons use the same "dd" floating-point type. The log files of Figure 29 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].

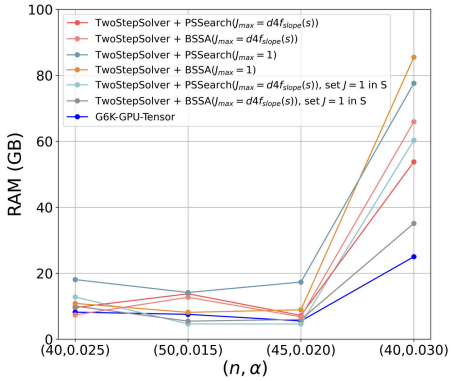
As shown in Figure 4, our approach significantly improves the speed of solving LWE and effectively addresses practical challenges. All experimental results, except those in Table 4, were obtained using 32 threads and 2 GPUs on a workstation with an Intel Xeon 5128 (16 cores, 32 threads)@2.3 GHz, 1.48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

Additionally, the experiments with $\alpha = 0.010$ shown in Figure 29(c) indicate that the flexible use of a solving strategy with jump>1 achieves a speedup factor of 4.88 to 7.85 compared with executing the same strategy after setting every jump value to 1. Specifically, in Figure 29(c), we define S_1 as the strategy generated by PSSearch with $J_{\max}$=$\mathtt{d4f}_{\text{slope}}(s)$, and S_2 as the strategy generated by PSSearch with $J_{\max}$=1. Executing S_1 achieves a walltime speedup factor of 1.77 to 2.91 over S_2, and a factor of 4.88 to 7.85 over executing S_1 after setting every jump value to 1 while retaining its blocksize sequence. Here, the s in $\mathtt{d4f}_{\text{slope}}(s)$ represents the simulated slope of the lattice basis during the reduction process.

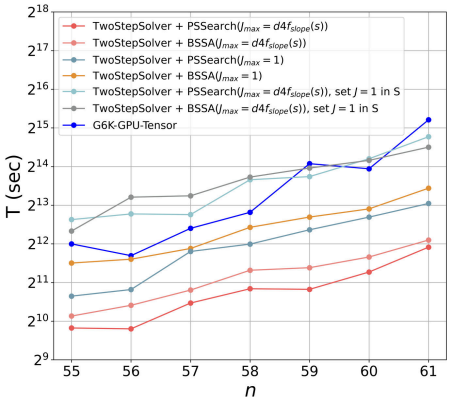
[4] https://github.com/WvanWoerden/G6K-GPU-Tensor/blob/main/lwe_challenge.py



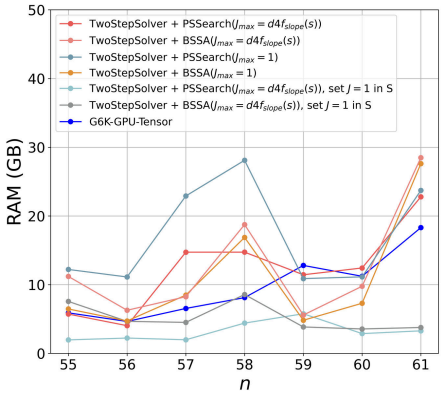
(a) LWE Challenges: Time.



(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 (Section 8.7) with threads=32 and GPUs=2. "default G6K" refers to the method implemented in `lwe_challenge.py`. TwoStepSolver + PSSearch(·) (resp. TwoStepSolver + BSSA(·)) denotes TwoStepSolver instantiated with a strategy generated by PSSearch (resp. BSSA; see Appendix B). $J_{\max}$ denotes the maximum jump value in the strategy. "Set J=1 in S" means first generating a strategy S with $J_{\max} > 1$, and then setting every jump value to 1 while retaining the blocksize sequence.

Fig. 29: Comparison of Different LWE-solving Algorithms under the same benchmark conditions. §

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.

C.3 The Optimized Strategy for the LWE Challenge

In Table 9, 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 "implement_unsolved_lwechal.sh" in source code [2].

Table 9: 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