Improved Progressive BKZ with Lattice Sieving

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

[1] School of Cyber Engineering, Xidian University, Xi'an, 710071, China
xiawenwen@stu.xidian.edu.cn
[2] Lab of Cryptology and Computer Security, Shanghai Jiao Tong University, Shanghai, 200240, China
[3] State Key Laboratory of Integrated Service Networks, Xidian University, Xi'an, 710071, China
lzwang_2@stu.xidian.edu.cn, bcwang@xidian.edu.cn
[4] School of Electronic, Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai, 200240, China
{wanggxx, dwgu}@sjtu.edu.cn

Abstract. The *unique Shortest Vector Problem* (uSVP) is one of the core hard problems in lattice-based cryptography. In our current knowledge, G6K-GPU-Tensor (Ducas et al, Eurocrypt 2021), which includes the code implementation for the heuristic uSVP solving algorithm in G6K-GPU-Tensor, is considered the fastest and contains the state-of-art BKZ and sieving implementations currently. In this paper, we propose a new way named *Improved Progressive BKZ with Lattice Sieving(ProPnJBKZ)* to accelerate the uSVP solving algorithm in G6K-GPU-Tensor. We break the TU Darmstadt LWE Challenges with $(n, \alpha) \in \{(40, 0.035), (40, 0.040), (50, 0.025), (55, 0.020), (90, 0.005)\}$. We test the efficiency of our ProPnJBKZ with the TU Darmstadt LWE Challenge. The experiment result shows that ProPnJBKZ is 7.2∼17.0 times faster than the heuristic uSVP solving algorithm in G6K-GPU-Tensor under same benchmark in solving TU Darmstadt LWE Challenges with $(n, \alpha) \in \{(40, 0.025), (40, 0.030), (45, 0.025), (50, 0.015)\}$. Besides, our algorithm also answers the open question Q7 mentioned in Kyber. We discuss the novel techniques used in ProPnJBKZ as follows: (1) Based on PnJBKZ and Sieve, this paper solves the problem of how to do the reduction and when to move from the first step (the reduction process using PnJBKZ) to the second step (search process using Sieve) to minimize the total time cost of solving uSVP by using our new designed strategy selection algorithms EnumBS, BSSA; (2) Present a PnJBKZ simulator that can predict PnJBKZ with a large jump and an estimation method for sieve algorithm in second step.

Keywords: Lattice cryptanalysis · uSVP · progressive BKZ · PnJBKZ Simulator · Optimized blocksize and jump strategy selection.

1 Introduction

To date, many post-quantum cryptosystems are lattice-based, e.g. Dilithium [1], Kyber [2] which have been accepted as NIST standards and are considered to be immune from both classical and quantum attacks. A large fraction of lattice-based cryptographic mechanisms are built upon the LWE [3] and its variants [2–5]. One of the best-known cryptanalytic techniques against LWE is primal attack [6] which solves the LWE by reducing it to the *unique Shortest Vector Problem* (uSVP_γ). BKZ [7] as the most popular lattice reduction algorithm usually used in solving uSVP_γ. BKZ is a combination of the LLL [8] and an SVP algorithm to balance the time cost and the success probability using blocksize β. Many cryptanalysts improved the BKZ algorithm, e.g. the extreme pruning [9] to speed up enumeration, BKZ 2.0 [10] based on [9], approximate enumeration oracle [11], and parameters optimization in BKZ such as *Improved Progressive BKZ* (ProBKZ) [12]. [13] showed that the unique shortest vector is recovered by first finding its projection in a projected sublattice then lifting it to the full lattice and verifying the BKZ successful condition of solving uSVP_γ in [14].

In 2019, Albrecht *et al.* [15] designed the *General Sieve Kernel* (G6K), implemented the progressive sieving algorithm [16] named `Pump` which can selectively call the Gauss sieve [17,18], NV sieve [19], k-list sieve [20,21] or BGJ1 sieve [22]. Progressive sieve and dimension-for-free (d4f) technique [23] are used in `Pump` for acceleration. `Pump` is a sieve and insertion algorithm that sieves on the projected sublattice and can insert more than one vector into the lattice basis. Ducas *et al.* [24] improved G6K using GPU (named G6K-GPU-Tensor) and implemented the fastest sieving algorithm BDGL16 [25] in both G6K and G6K-GPU-Tensor. G6K provides an algorithm to solve LWE (uSVP_γ), which will conditional call `Pump` to find short vectors on the projected sublattice and lift them into the full lattice basis after running several tours of PnJBKZ. PnJBKZ uses `Pump` as its SVP oracle so that even if it calls the SVP oracle with a jump value, it still can ensure the skipped basis vector is reduced by `Pump`.[1] It solves TU Darmstadt LWE Challenges 400 times faster than the previous records. However, the default mode in G6K still has room to improve: *(1) The condition in G6K calls* `Pump` *heuristically, not to minimize the total time cost of solving* $uSVP_\gamma$*. (2) The inserted* `Pump` *possibly fails to find the target vector and G6K will return to call PnJBKZ for further reduction and the cost of a failed* `Pump` *is wasted. (3) It only considers PnJBKZ with jump=1 during the reduction, while Fig. 6 shows that PnJBKZ with a large jump is more efficient. (4) The blocksize strategy used in G6K is trivial and optimizable.*

In fact, behind the improvement of (3) how to optimize the reduction parameters in the reduction like the selection of the blocksize in progressive BKZ has long been an open question (Q7 in Section 5.3 of Kyber [2]) in Two-step mode. Here the Two-step mode means that first calls several BKZ or PnJBKZ tours to improve the quality of lattice basis and then calls a sieve algorithm in the end.

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

Two-step Mode. The idea of the Two-step mode was first proposed in solving BDD and LWE problems in [26]. [12] uses it in solving approximate SVP_γ with 1.05 GH approximate factor. [27] proves that given the same blocksize strategy and assume GSA holds, the Two-step mode can solve an uSVP_γ in less time than BKZ-only mode. [28] also shows that a Two-step mode uses progressive BKZ or PnJBKZ as preprocessing and only applies HKZ-slide reduction in the final step finding the short vector is more efficient than other solving modes.

BKZ Simulator. In 2020, Li and Nguyen [29] present the first rigorous dynamic analysis of BKZ which proves that after at most $\Theta\left((d^2/\beta^2)\log d\right)$ tours reduction of BKZ-β, the norm of the first basis vector output from BKZ-β at most $\gamma_\beta^{\frac{d-1}{2(\beta-1)}+\frac{\beta(\beta-2)}{2d(\beta-1)}} \cdot \det(\mathcal{L})^{\frac{1}{d}}$. However, the estimation proposed in [29] considers the reduction effort of fixed blocksize BKZ-β after running a large number of tours. In our work, we need a polynomial-time BKZ simulator to predict the practical reduction effect of BKZ with more flexible tours. Since the running tours of a blocksize fixed BKZ-β are very small in most cases during our reduction. A BKZ simulator is necessary to optimize the selection of the blocksize during reduction, which is a polynomial time algorithm to predict the reduction effort of the exponential time BKZ algorithm. Based on the Gaussian heuristic, [10] refined the sandpile model from [30] and provided a BKZ simulator. Using the property of HKZ reduced basis and Gaussian heuristic, [12] proposed a simulator for predicting BKZ-β fully reduced basis. Bai *et al.* [31] proposed a simulator to predict and explain the phenomenon of head concavity of lattice basis reduced by multiple tours of BKZ-β. However, as above simulators cannot be directly used for PnJBKZ simulation when it uses the jump strategy.

Strategy Selection Algorithm. Based on the BKZ simulator, in 2016, Aono *et al.* presented an improved progressive BKZ (ProBKZ) [12] which proposed a blocksize strategy selection algorithm to generate the progressive blocksize for reduction. It uses the shortest path algorithm to give an optimized blocksize strategy by setting multiple different middle reduction qualities as the inner nodes. But ProBKZ only considers enumeration as its subroutine, without using the more efficient lattice sieving algorithm. Besides, in this paper, we shall show that their method is not supposed to generate a blocksize strategy with minimum expected time cost, and their strategy still can be improved.

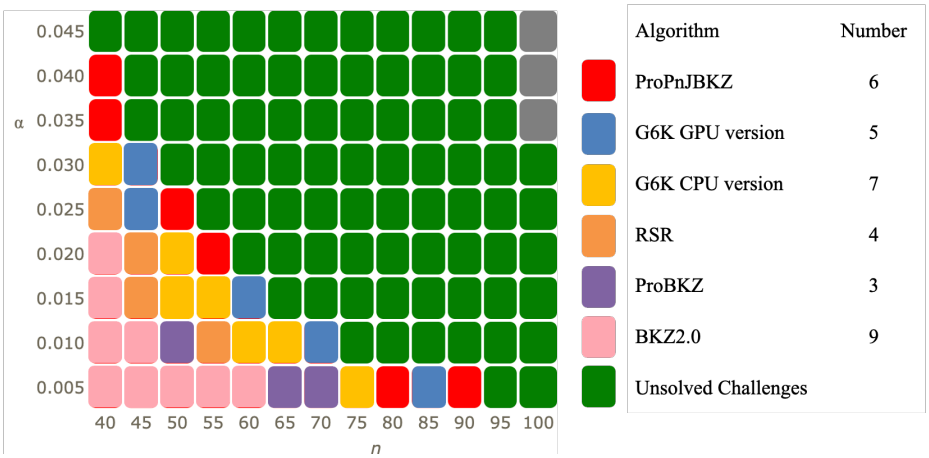


Fig. 1: LWE Challenges and the Algorithms to Solve it

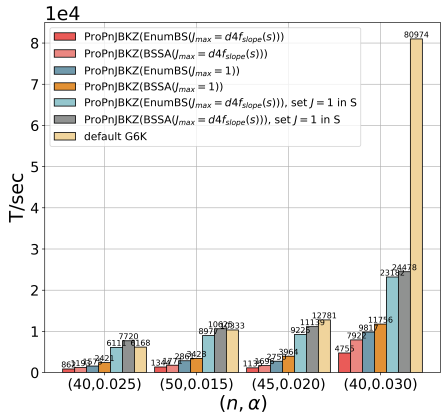
Contribution. In this work, we aim to further solve the above three problems and then propose the *Improved Progressive PnJBKZ*(ProPnJBKZ, Alg. 1) based on the following optimizations: Firstly, the experiments in Sec. 3.1 show that the Two-step mode is not only faster than the BKZ-only mode, which has been proven under GSA in Theorem 1 of [27], but also faster than the default G6K mode if we use the same reduction strategy and find an appropriate timing to enter the search step. Then, to simulate the quality improvement of the lattice basis during progressive lattice reduction and estimate the expected cost of the last sieve algorithm, we design a polynomial-time PnJBKZ simulator to predicate the reduction effect of exponential-time PnJBKZ algorithm even in jump>1. Finally, based on the PnJBKZ simulator and a `Pump` sieving dimension estimation, we design several strategy selection algorithms (including BSSA and EnumBS) to find the optimized blocksize and jump strategy and the most suitable timing of entering `Pump` to minimize the total time cost of solving uSVP$_\gamma$.

Our ProPnJBKZ executable code are made public on GitHub[2] which successfully cracked the TU Darmstadt LWE Challenge[3] by `ProPnJBKZ_for_lwe.py`. This challenge generates LWE instances with varying difficulty levels based on parameters n and α to test the efficiency of algorithms for solving LWE. Our approach detailed in Fig. 1, significantly improves solving uSVP$_\gamma$ and effectively tackles practical challenges. All the experimental results except Table 7 are tested with 32 threads and 2 GPUs on a workstation with Intel Xeon 5128 16c 32@2.3GHz, 1.48T RAM and NVIDIA RTX 3090*2, denoted as machine C.

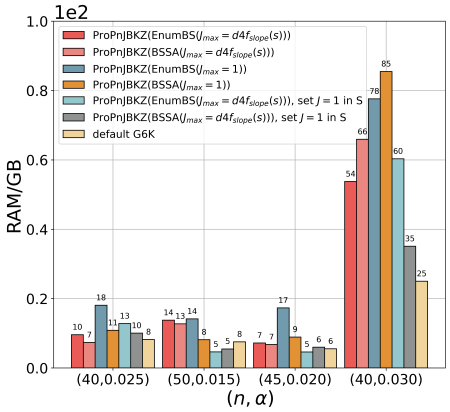
Besides, we also test the performance of ProPnJBKZ for solving uSVP$_\gamma$ based on our blocksize and jump strategy selection algorithm and the Two-step mode. Using the blocksize and jump strategy chosen from EnumBS, the algorithm significantly increases the efficiency at most 17.0 times (at least 7.2 times) under the

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

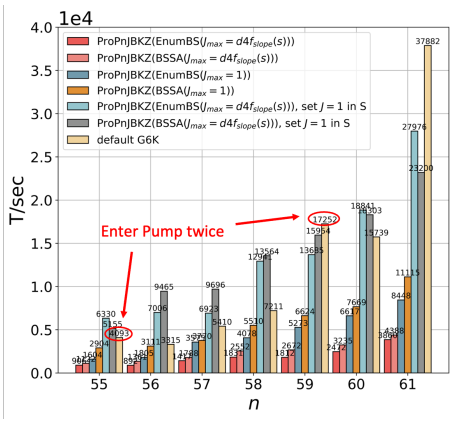
same benchmark in solving some of TU Darmstadt LWE Challenges compared with that of the default LWE solver in G6K shown in Fig. 2(a) with an acceptable memory cost (Fig. 2(b)). Meanwhile, Fig. 2(c) shows that the ProPnJBKZ using the strategy generated by EnumBS (abbreviated as ProPnJBKZ(EnumBS(·))) is faster than ProPnJBKZ using the strategy generated by BSSA (abbreviate as ProPnJBKZ(BSSA(·))). ProPnJBKZ(EnumBS(·)) is 3.71∼9.81 times faster than default G6K and ProPnJBKZ(BSSA(·)) is 2.43∼8.63 times faster than default G6K in testing the given LWE instances $(n, \alpha) \in \{(n, 0.010) | n{=}55, \cdots, 61\}$, especially in the case of large n with the acceptable memory cost(Fig. 2(d)). Especially, the cost of LWE instance with $(n, \alpha) \in \{(55, 0.010), (59, 0.010)\}$ in Fig. 2(c) shows that the time cost of default G6K is high since it calls the reduction first then enter the `Pump` and repeated such a process twice before default G6K found the solution. We test various $J_{\max}$ to illustrate the effect of $\mathsf{jump} > 1$ in accelerating solve LWE. For instance, set S_1 as the strategy generated by EnumBS with $J_{\max}{=}\mathrm{d4f}_{\mathrm{slope}}(s)$ and S_2 as the strategy generated by EnumBS with $J_{\max}{=}1$. Then the computational cost incurred using strategy S_1 is 2.02 to 2.91 times faster than that of S_2 and 4.88 to 7.85 times faster than the scenario where J is set to 1 in S_1. Here s is the slope of the lattice basis during the simulated process. It demonstrates a substantial improvement in efficiency when the maximum jump exceeds 1, potentially by streamlining the computational steps involved. Furthermore, one can also limit the maximum memory usage of solving uSVP_γ by changing parameter "`--max_RAM`" to generate the solving strategy. We also designed an LWE sample optimized selection algorithm (Alg. 6) to optimize the number of chosen LWE samples in Appendix F, although its efficiency improvement is not significant (at most 2.2% in our test).



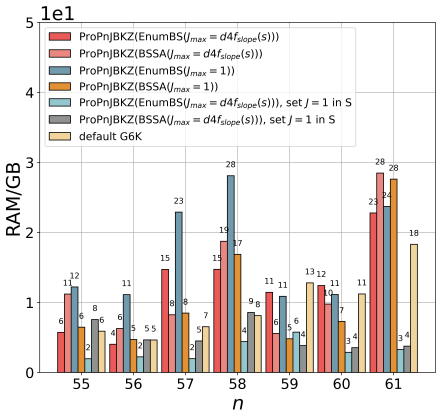
(a) LWE Challenges: Time.



(b) LWE Challenges: Memory.



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



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

§ The experiment used "dd" float type and pump/down=True, under identical benchmark conditions on a machine C (Sec. 6.4) with threads=32 and GPUs=2. "default G6K" refers to the method in g6k implemented in lwe_challenge.py. ProPnJBKZ(EnumBS(·)) (ProPnJBKZ(BSSA(·))) represents the cost of running ProPnJBKZ with strategies from EnumBS (BSSA). $J_{\max}$ denotes the maximum jump value in the strategy. "Set J=1 in S" means generating a strategy S and then setting the jump value as 1.

Fig. 2: Comparison of Different LWE-solving Algorithms under same benchmark. §

Key idea. Four main parts of ProPnJBKZ: Two-step mode, PnJBKZ Simulator, Pump Estimation method and Strategy Selection Algorithm (Fig. 3):

- *We show that the Two-step mode can solve an uSVP$_\gamma$ in less time than the BKZ-only mode and default mode in G6K through experiments.* Concretely, to minimize the total time cost of solving uSVP$_\gamma$, based on our PnJBKZ simulator and Pump sieving dimension estimation, we give a Two-step mode, which firstly

calls a series of PnJBKZs following a blocksize and Jump selection strategy to do reduction then at a suitable timing, calling a `Pump` to search the target vector. Besides it saves the time cost of a failed `Pump` that occurs during the solving process in G6K's default strategy by a non-negligible probability.

Table 1: Technical comparison from ProBKZ and G6K.

Technique	ProBKZ [12]	G6K [15, 24]	Our Algorithm
Reduction Strategy	Approximate minimum simulated cost by progressive BKZ and enumeration	Trival blocksize strategy	Minimum simulated cost by progressive PnJBKZ and `Pump`
Strategy Generation	Shortest Path Algorithm	No	EnumBS (in Parallel)
Search Cost Estimation	FEC based	New Hope [25] with d4f [23]	New `Pump` dimension estimation and PSC based
Search Timing	Minimize total time cost	Heuristic	Minimize total time cost
Predicting $\|\mathbf{b}_i^*\|$	Plain BKZ simulator	No	PnJBKZ simulator J$\geq$1

- *Construct PnJBKZ simulator.* To improve the efficiency of reduction by using a more flexible jump and blocksize strategy, we construct a polynomial-time PnJBKZ simulator, which can accurately predicate the behavior of PnJBKZ even jump>1 without actually running exponential time PnJBKZ. Specifically, since the `Pump` used by PnJBKZ to do the reduction in each block, inserts more than one vector into the lattice basis, and each of the projected sub-lattice basis reduced by a `Pump` is almost HKZ-reduced, we predicate the reduction effect of PnJBKZ with jump>1 by calculating these unknown Gram-Schmidt norms based on the property of HKZ reduced basis. It has been verified experimentally.

- *Provide a new estimation for `Pump`.* Our experiment (Fig. 9) demonstrates that using the estimated dimension [14, 32] from Sec. 6.4 of the G6K paper [15] has a non-negligible probability of causing failure in finding the target vector. Therefore, we introduce a new method for estimating the dimension of `Pump` and provide the `Pump` Solvable Time Cost (PSC, see Sec. 4.2) by considering the distribution of the projected target vector to ensure the accuracy. Our new estimation ensures solving the uSVP$_\gamma$ problem by `Pump` with estimated dimension.

- *Design two new strategy selection algorithms EnumBS, BSSA.* Based on the simulators and Two-step mode, to improve the efficiency of the reduction step, we give two new reduction strategy generation algorithms, which generate the blocksize and `jump` strategy and the suitable sieving dimension for `Pump` to minimize the expected cost of solving uSVP$_\gamma$. The reduction strategy generation algorithms proposed in this work solved the problem of how to do the reduction most efficiently (improving the quality of lattice basis to any specified level with minimum time cost), while most of the previous works [10, 15, 24, 33] choose the heuristic reduction strategy. In addition, we also solved the problem of when to end the reduction. The code for strategy selection algorithms for solving LWE is available in the folder "strategy_gen" of the code[2]. We borrow the same shortest

path algorithm as in ProBKZ to design our first algorithm called *blocksize and jump strategy selection algorithm based on ProBKZ* (BSSA) by replacing the estimated cost and simulation algorithm of BKZ and enumeration algorithm with PnJBKZ and `Pump` respectively. Then to obtain a reduction strategy to minimize the simulated cost for ProPnJBKZ, we design a new strategy selection algorithm named *blocksize and jump strategy enumeration* (EnumBS). EnumBS can obtain a blocksize and jump selection strategy that can solve an uSVP_γ instance in less time but at a higher theoretical complexity than BSSA in generating, but the time is still acceptable for low-dimensional lattices. EnumBS first generates the reduction strategy list BS to store the information about the strategy, whose child may be the strategy with minimum simulated cost(if Heuristic 6 holds). All the strategies in BS are ordered from the growth of estimated PnJBKZ cost and decrease of PSC. Let S be a father strategy in BS, the child strategy denotes S joined with another strategy S' such that the child strategy could improve the basis quality. The main process of EnumBS is to iteratively determine whether each child of the strategy in BS would have a shorter time overhead. If so, add the child strategy and delete the strategies in BS of which the simulated PnJBKZ cost and PSC are both longer than the added strategy.

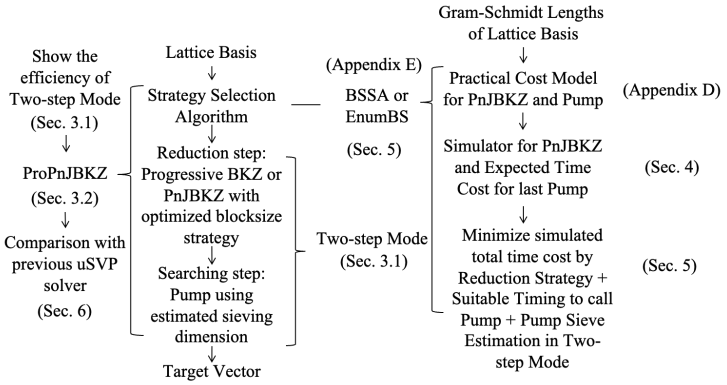


Fig. 3: Roadmap: ProPnJBKZ Process

Comparison and discussion. Table 1 compares the techniques used in the ProBKZ [12], G6K [15, 24] and our algorithm. Our Algorithm uses the strategy selection algorithm EnumBS which has a smaller time cost in solving LWE compared with that of ProBKZ. Our algorithm terminates the PnJBKZ strategy by using a sieve dimension estimation proposed in [27] and we give a `Pump` Solvable Cost(PSC) estimation for the last `Pump` considering the distribution of the norm of the projected target vector on the sublattice, instead of the Full Enumeration Cost(FEC) estimation in ProBKZ. To predict the reduction effect of the PnJBKZ (especially with jump>1), we design the PnJBKZ simulator.

Roadmap. The paper is organized as Fig. 3. Sec. 2 presents the notations and preliminaries. We firstly show the efficiency of Two-step mode in Sec. 3.1 and give the sketch of ProPnJBKZ in Sec. 3.2. Next, we propose a PnJBKZ

simulator and PSC estimation for generating blocksize and jump strategy in Sec. 4 and give a practical cost model of PnJBKZ and Pump for designing the strategy selection algorithm. Then, Sec. 5 gives a detailed description of our two blocksize and jump strategy selection algorithms BSSA and EnumBS. Besides, we experiment compare their cost with default G6K, give the optimized strategy of LWE Challenge, and verify the simulation accuracy of ProPnJBKZ in Sec. 6.

2 Preliminaries

2.1 Notations for algorithms description.

Let BKZ-β (or PnJBKZ(β, J)) be an abbreviation of a one-tour BKZ (or PnJBKZ) with blocksize β and jump value J. Assume $\mathbf{B} = (\mathbf{b}_0, \cdots, \mathbf{b}_{d-1})$, its Gram-Schmidt basis is $\mathbf{B}^* = (\mathbf{b}_0^*, \cdots, \mathbf{b}_{d-1}^*)$. Denote l_i by the logarithm of Gram-Schmidt norm, i.e. $l_i = \ln(\|\mathbf{b}_i^*\|)$, for $i \in \{0, \cdots, d-1\}$. Let $\mathsf{rr}(\mathbf{B}) = (l_0, \cdots, l_{d-1})$, abbreviate to rr, $\mathsf{rr}_{[i:j]} = (l_i, \cdots, l_{j-1})$.

Denote BKZSim by the BKZ simulator proposed in [10]. The simulation for PnJBKZ is denoted as PnJBKZSim$(\mathsf{rr}, \beta, J, t)$, which simulates a PnJBKZ(β, J) with t tours on the lengths rr and return the new lengths. Moreover, if we have a blocksize and jump strategy S that stores a series of (β_i, J_i), then PnJBKZSim$(\mathsf{rr}, \mathsf{S})$ means iteratively calling a tour of PnJBKZ(β_i, J_i) simulator on rr, where $(\beta_i, J_i) \in \mathsf{S}$. Assume the input basis is $\mathbf{B}$, and the basis $\mathbf{B}$ reaches a basis quality after calling sufficient tours of BKZ-β. To simplify the above step, we use β to imply the quality of a BKZ-β reduced basis. Let $\sharp$tours(rr, BKZ-β) (or $\sharp$tours(rr, PnJBKZ(β, J))) be the minimum tours for BKZ-β (or PnJBKZ(β, J)) from a lattice basis with GS-lengths rr to reach a BKZ-β (or PnJBKZ(β, J)) reduced basis. Denote t or $\sharp$tours as the number of tours for implementing BKZ (or PnJBKZ) with a fixed blocksize β. Let $T_{\mathrm{BKZ}}(\beta)$ (or $T_{\mathrm{PnJBKZ}}(\beta, J)$) be the time cost of one-tour BKZ (or PnJBKZ) with blocksize β and jump value J. Let $T_{\mathrm{PnJBKZs}}(\mathsf{S})$ be the total time cost for a series of PnJBKZ with a specific reduction strategy $\mathsf{S}=\{(\beta_0, J_0), \cdots, (\beta_{n-1}, J_{n-1})\}$, abbreviate it as T_{PnJBKZs}. Denote $T_{\mathrm{Pump}}(d_{\mathrm{svp}})$ as the time cost of Pump with d_{svp} sieving dimension, abbreviate it as T_{Pump}. Let *Pump Solvable Cost* (abbreviate as PSC) be the expected Pump cost to find the target vector (See Sec. 4.2).

2.2 Technologies in G6K

Dimension for Free (d4f) Technique. D4f technology [23] can bring sub-exponential time speedup and memory decrease for sieve algorithms. [23] has given two theoretical d4f estimations for solving β-dimension SVP as $\mathrm{d4f}(\beta) = \beta \ln(4/3)/\ln(\beta/2\pi)$ and $\mathrm{d4f}(\beta) = \beta \ln(4/3)/\ln(\beta/2\pi e)$, while in the implementation of G6K [15], it gives a more relaxed value $\mathrm{d4f}(\beta)$: when $\beta < 40$, $\mathrm{d4f}(\beta) = 0$; when $40 \le \beta \le 75$, $\mathrm{d4f}(\beta) = \lfloor \beta - 40/2 \rfloor$; when $\beta > 75$, $\mathrm{d4f}(\beta) = \lfloor 11.5 + 0.075\beta \rfloor$. However above d4f estimations are fixed according to the β, [34] also proposed an upper bound of d4f value estimation function based the quality of current

lattice basis that under GSA one can use δ to measure the quality of lattice basis and [34] illustrates that $\mathrm{d4f}_\delta = \ln_\delta \sqrt{4/3} \approx \ln(4/3)/(-slope)$. Here the *slope* is the slope value of the logarithm of Gram-Schmidt norms l_i for $\forall i \in \{1, ..., d\}$. More detail about $\mathrm{d4f}_{slope}(s)$ can see the Eq. (5) in [34].

Pump in G6K. Albrecht *et al.* proposed `Pump` in [15], which is improved based on *Progressive Sieve* [16] with d4f technique [23] and the insertion tricks in [15]. There are four input parameters for `Pump` algorithm: lattice basis $\mathbf{B}$, left insertion bound κ, insertion upper bound d_{svp} and d4f value $\mathrm{d4f}(d_{\mathrm{svp}})$. Here $\kappa{+}d_{\mathrm{svp}}{=}d$ and the upper bound of sieve dimension is $d_{\mathrm{svp}} - \mathrm{d4f}(d_{\mathrm{svp}})$. It's worth emphasizing that the `Pump` algorithm will insert up to $d_{\mathrm{svp}} - \mathrm{d4f}(d_{\mathrm{svp}})$ short vectors into the basis index from κ to d by the short vector that has the shortest norm in the short vector set obtained by each sieve on the projected sublattice $\mathcal{L}_{\pi[d-d_{\mathrm{svp}}+i:d]}$ for i from κ to d. Thus, it performs a partial HKZ reduction and outputs a nearly HKZ-reduced context as the paper of G6K [15] mentioned.

Slope in G6K. To measure the quality of lattice basis [15] use the averaged quality measurement. It is the least squares fit coefficient of the slope of $\log\|\mathbf{b}_i^*\|$, which means that the slope closer to 0, the better basis quality.

PnJBKZ in G6K. PnJBKZ (`Pump` and Jump BKZ) is a BKZ-type reduction algorithm that uses `Pump` as its SVP oracle. The main difference from `Pump` to the traditional SVP oracle is that it can insert more than one vector into the inputting basis of sublattice and will output a nearly HKZ-reduced context. Thus, PnJBKZ can perform such an SVP oracle with an adjustable `jump` no less than 1. Specifically, running a PnJBKZ with blocksize β and `jump`$=J$ means that after executing SVP oracle on a certain block $\mathbf{B}_{[i:i+\beta]}$, the next SVP oracle will be executed on the $\mathbf{B}_{[i+J:i+\beta+J]}$ block with a `jump` $= J$ rather than $\mathbf{B}_{[i+1:i+\beta+1]}$.

3 ProPnJBKZ in Two-step mode for solving uSVP$_\gamma$

In solving the uSVPγ problem, we typically use either reduction algorithms like BKZ or search algorithms like Sieve and Enumeration, sometimes combining them. However, the best way to combine them effectively remains uncertain. We'll discuss the Two-step Mode, which first applies several reduction algorithms to improve the lattice basis quality and then uses a search algorithm to ensure finding the target vector. This comparison is detailed in Sec. 3.1. Then, we give a sketch of ProPnJBKZ in Sec. 3.2, where we use the Two-step mode with a strategy selection method for solving uSVP$_\gamma$.

3.1 Comparison of uSVP$_\gamma$ solving Mode

In this part, we introduce BKZ-only Mode and Default Mode in G6K for solving the uSVP$_\gamma$ in the literature and compare them with the Two-step mode. We also give an experiment to prove the comparison result. The main differences among the above three modes are as follows: (1) The reduction ability of a reduction algorithm (such as BKZ and its variants) or a special search algorithm (such as `Pump` as the blue boxes in Fig. 4 shown. `Pump` inserts vectors into lattice basis

more than once, so it can also be used to improve the quality of lattice basis. (2) The ability to find the short vector using a reduction algorithm or search algorithm as the orange boxes in Fig. 4 shown. (3) Rather than a Heuristic condition to call `Pump` in Default Mode in G6K, the Two-step mode gives the most suitable timing of when to call the `Pump` in the search step.

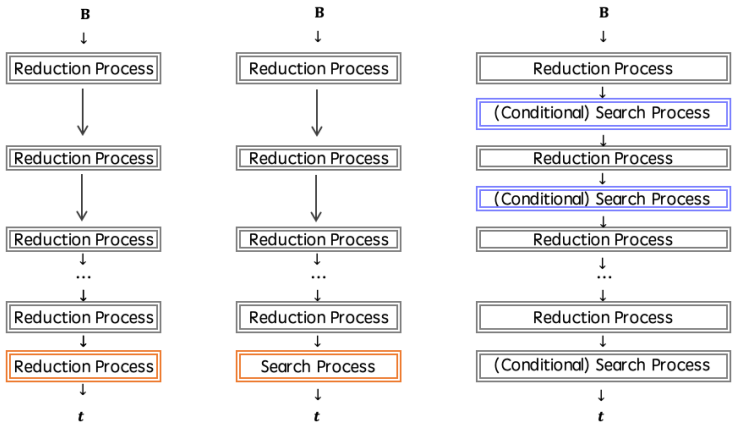


Fig. 4: Different mode to solve the uSVP$_\gamma$: BKZ-only (Left Column), Two-step (Middle Column), Default Mode in G6K (Right Column). t is the target vector of uSVP$_\gamma$.

BKZ-only Mode BKZ-only mode [14, 35] (Fig. 4, left column) implements multiple tours of lattice reduction algorithm (such as BKZ or its variants) until solving the uSVP$_\gamma$ problem. The BKZ-only mode is the mainstream method in the security estimation of LWE-based cryptosystems, such as in [25, 36].

Default uSVP$_\gamma$ solving Mode in G6K G6K [15, 24] solve the LWE problem by primal attack, i.e. reduce LWE to the uSVP$_\gamma$ and solve it by calling progressive PnJBKZ and a conditional `Pump` (The algorithm calls `Pump` only if the estimated time cost of `Pump` is shorter than an upper bound computed) repeatedly. We can extract a uSVP$_\gamma$ solving mode from their implement `lwe_challenge.py` and name it as the *Default uSVP$_\gamma$ solving Mode in G6K*, abbreviated as default Mode in G6K (Fig. 4, right column).

In the default uSVP$_\gamma$ solving mode of G6K, it will reduce the basis by a heuristic specific blocksize strategy S_0[4]. After each lattice reduction by PnJBKZ $\beta \in \mathsf{S}_0$, J=1, default Mode in G6K will record the time cost of the reduction process and determine whether a `Pump` will finish in the same or smaller time cost: $T_{\text{Pump}} \leq T_{\text{BKZs}}$. If it does, it will call a `Pump`; If not, it will skip to the next PnJBKZ. Therefore, the condition for calling `Pump` is a heuristic condition in the default uSVP$_\gamma$ solving Mode of G6K, which is not a condition set to solve uSVP$_\gamma$ within the minimum total time cost. Meanwhile, if a `Pump` is triggered but this `Pump` fails to find the target vector of LWE, G6K also goes back to reducing the

[4] S_0 = list(range(10, 50)) + [b−20, b−17] + list(range(b−14, b+25, 2)).

basis with progressive BKZ and resets the time recorder. The search of Pump may also succeed before reaching the Pump dimension. For more detail, see the "Implemented strategy and performance" part in Section 6.4 of [15].

The benefit of the default uSVP$_\gamma$ solving mode in G6K is that if we do not have an accurate simulator for BKZ or PnJBKZ and we are not sure of the solvability by a final Pump calling, then a Default uSVP$_\gamma$ solving Mode in G6K will make sure in outputting the required result in a reasonable time. However, the condition for calling Pump in G6K is heuristic, not to minimize the total time cost and G6K sometimes enters a Pump with solving failure (see Fig. 9) because of its over-optimistic dimension estimation and waste processing time heavily since a Pump call is costly. Here a failed Pump means that the dimension setting of the sieving in the Pump is over-optimistic, which makes the Pump fail to find the target vector (Fig. 9). Besides, it might enter a Pump late and waste the processing time of extra cost for several PnJBKZs with large blocksizes.

Two-step Mode The Two-step mode (Fig. 4, middle column) was firstly informally stated in [26] for solving BDD problem and LWE problem, which calls a series of BKZ (or its variant) first for lattice reduction and calls a search algorithm to find the target vector at last.

In this paper, we show that a Two-step mode adapted to PnJBKZ and Pump is more efficient in solving uSVP$_\gamma$, which calls a series of PnJBKZ for reduction first and at a good timing uses a Pump to search the unique shortest vector. Unlike the heuristic condition of calling Pump in the *default uSVP$_\gamma$ solving Mode in G6K*, here the main difference between our Two-step mode and *default uSVP$_\gamma$ solving Mode in G6K* is that to minimize the total uSVP$_\gamma$ solving time cost, based on our PnJBKZ simulator Alg. 2 and Pump sieving dimension estimation Alg. 3 in Sec. 4.1, we can give the most suitable timing of calling the Pump in search step.

Experiments of Comparison among BKZ-only Mode, Default Mode in G6K, and Two-step Mode In this part, we give an experimental result to illustrate that for solving uSVP$_\gamma$, the Two-step mode can solve an uSVP$_\gamma$ instance in less time than both BKZ-only mode, and the G6K default mode while it enters Pump at least twice. For ease of comparison, let the blocksize strategy used in different modes be the same and simply set jump=1. We assume all of the above three modes can solve the uSVP$_\gamma$ instance after running all the steps.

In our test the Two-step mode and the BKZ-only mode both run the same BKZ tours at the beginning with blocksize from 10 to 39. However, in their final stages to find the unique shortest vector, the Two-step mode calls a Pump, while the BKZ-only mode calls one or more BKZ tours (as orange boxes in Fig. 4).We call a Pump-$(\kappa, d_{\text{svp}}, f)$ and a PnJBKZ(β, J) $(\beta{<}d_{\text{svp}})$ separately on the same lattice basis, denote cost of Pump as T_{Pump}. Table 2 shows that the reduced shortest vector $\mathbf{b}_0$ after a Pump is shorter than that after a PnJBKZ$(55, 1)$ while $T_{\text{Pump}} \leq T_{\text{PnJBKZ}}$. Thus calling a Pump is more likely to find a shorter vector compared to the PnJBKZ in no more time cost than PnJBKZ. So the Two-step mode can solve an uSVP$_\gamma$ instance in less time than the BKZ-only mode.

On the other hand, both the Two-step mode and the G6K default mode ends with a `Pump` which outputs the unique shortest vector. Their differences are at the earlier stage, where the Two-step mode always calls BKZ, while the G6K default mode may call an early `Pump` without a solution and such a `Pump` is less beneficial than calling a BKZ in same cost to increase the quality of lattice basis.

Table 2: Norm of $\mathbf{b}_0$ after PnJBKZ and `Pump` with equal time limits.

$(n,\alpha)^\dagger$	Cost‡	$\ln(\|\mathbf{b}_0\|^2)$	
		PnJBKZ	`Pump`$(\mathcal{L}_{[\kappa:d]})$
(55,0.005)	23.9	15.82	10.55
(40,0.015)	6.6	14.76	11.27
(45,0.010)	18.0	15.04	11.12
(40,0.020)	17.1	14.76	12.05

† LWE Challenge Lattice basis with (n,α).

Table 3: PSC after PnJBKZ and `Pump` with equal time limits.

(n,α)	Cost‡	$\log_2(\text{PSC}^\S)$	
		PnJBKZ	`Pump`$(\mathcal{L}_{[\kappa:d]})$
(55,0.010)	6.0	7.94	8.44
(60,0.005)	10.3	11.69	12.01
(70,0.005)	12.6	16.01	17.34
(75,0.005)	14.9	21.68	22.76

§ Here the cost unit of PSC is second.

‡ The cost in minutes while calling the corresponding algorithm.

The comparison between the Two-step mode and G6K default mode when G6K enters `Pump` at least twice, we show that an early `Pump` is less helpful in solving uSVP$_\gamma$ than an early PnJBKZ if both of them run in same time cost. Let the time cost of `Pump` for finding the target norm on the specific lattice basis, i.e. PSC, be a standard of measuring the quality of the lattice basis. Lattice basis with low PSC can be regarded as better quality. Our experiments separately call a PnJBKZ or `Pump` on the same lattice basis with the same reduction of BKZ-β, β from 10 to 19. Table 3 shows that basis quality after a PnJBKZ$(60, 5)$ reduction is better than that after a `Pump` reduction while $T_{\text{PnJBKZ}} \leq T_{\text{Pump}}$. The latter basis quality is estimated by the PSC estimation Alg. 3. So, in the G6K-default mode, if it enters a `Pump` with no solution, the quality of the returned lattice basis will be worse than that after a PnJBKZ reduction under the same time limit. In conclusion, the G6K-default mode is slower than the Two-step mode.

In Fig. 5, we compared the time cost among these three modes under the same blocksize strategy[4]. The remaining time cost in Fig. 5 has different meanings in each mode. In the case of BKZ-only mode, the remaining time represents the time cost of BKZ-only mode will continue to run progressive BKZ with remaining strategy [4] until finding the target vector after finishing the reduction of Pre-BKZ as Tab. 4 shown. In the case of G6K mode, the remaining time is the time cost of G6K mode to continue to run remaining reductions as strategy[4] first and enters a conditional `Pump` (if `Pump` fails it continues the above process) until finding the target vector. As for Two-step mode, the remaining time means the time cost of running the final `Pump`. Here the terminal condition of the reduction in Two-step mode is considered to minimize the total time cost of solving uSVP$_\gamma$. In Fig. 5, the Two-step mode is 1.2∼2.9 times faster than that of default G6K mode and 3.1∼18.9 times faster than that of BKZ-only mode in tested LWE instances.

$(n,\alpha)^{\circledast}$	Pre-BKZ strategy [β]
(67,0.005)	[10,11,$\cdots$,49,48]
(55,0.010)	[10,11,$\cdots$,49]
(59,0.010)	[10,11,$\cdots$,49, 69,72,75,77]
(61,0.010)	[10,11,$\cdots$,49,71,74,77,79,81,85]

$\circledast$ Lattice basis from randomly generated LWE instance with (n, α).

Table 4: Pre-BKZ strategy[4] in Fig. 5.

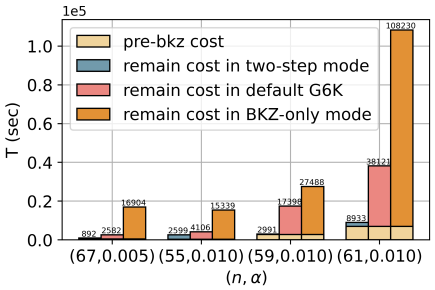


Fig. 5: Different Modes Comparison on Machine C with threads = 32 and gpus = 2.

3.2 Algorithm overview

Our *Improved Progressive BKZ with Sieve*(ProPnJBKZ) Algorithm is designed in Two-step mode and a strategy selection algorithm, which will be introduced in Sec. 5. It aims to speed up the efficiency for solving the uSVP$_\gamma$ and can be described as the following: input a lattice basis $\mathbf{B}$, and the distribution function $F(\star, \mathcal{D})$ of the (projected) target norm (see Sec. 4.2). Although $F(\star, \mathcal{D})$ can also be either simplified as a fixed value or a function solely related to the degree $\star$ of the distribution of (projected) target norm and the probability distribution $\mathcal{D}$ of each element in target vector (e.g. $F(\beta, N(0, \sigma^2)) = \sigma^2 \cdot \chi^2_\beta$). To accurately estimate the length of short vectors, we use a function to represent the actual distribution as input. This approach makes the estimation of the length of the shortest vector in the lattice more accurate, thereby enhancing the success rate of the final sieving process by setting an appropriate sieving dimension. To minimize the total cost, it first generates an optimized blocksize and jump strategy S using a strategy selection algorithm (EnumBS or BSSA, see Sec. 5) and reduce $\mathbf{B}$ by a series of PnJBKZ(β, J) according to S. Then it will call a `Pump` to find the target vector. The parameter selection of `Pump` follows Alg. 3, which leads to finding the unique shortest vector after `Pump`. The detailed process is as Alg. 1.

We improve the heuristic uSVP$_\gamma$ solver in G6K from two aspects: firstly, we use a Two-step mode with the optimized blocksize strategy. Besides, we can choose the optimized $\mathsf{jump}(\geq 1)$ strategy rather than always keeping the jump=1.

Although the experiments in [15] suggest that compared with the reduction strategy of jump=1, the reduction strategy of jump value with (1<jump<4) is not beneficial in small blocksize (around 60$\sim$80), we show that it is beneficial in large blocksize (from 80 to 134) and large jump (with 1, 4 and 9). More precisely, [15] shows that the reduction strategy of jump=3 requires similar running time to obtain the same quality of lattice basis reduced by the strategy of jump=1, with a larger memory consumption. However in our experiments, we give the experiment with a setting of larger jump strategy (with 1, 4, and 9), and find out that the wall time for reaching the same lattice basis quality decreases significantly. More details can be found in Fig. 6, which shows that the PnJBKZ with the

jump>1 has a smaller time cost (3∼6 times faster) while achieving the same reduction quality as that of the PnJBKZ with jump=1. Therefore, to find proper PnJBKZ reduction parameters, it is essential to construct a PnJBKZ simulator to handle the case for jump>1 which we will discuss in Sec. 4.1.

input : $\mathbf{B}$, $F(\star, \mathcal{D})$;
output: The unique shortest vector $\mathbf{v}$;
Function `ProPnJBKZ(`$\mathbf{B}$, $F(\star, \mathcal{D})$`)`**:**
  $\mathbf{B} = \texttt{LLL}(\mathbf{B})$;
  Generate Strategy S using EnumBS or BSSA;
  for $(\beta, J) \in \mathsf{S}$ **do**
    $\mathbf{B} \leftarrow$ Run a PnJBKZ(β, J) tour on basis $\mathbf{B}$;
  $d_{\text{svp}}, \_ \leftarrow \texttt{PumpDimEst}(\texttt{rr}(\mathbf{B}), F(\star, \mathcal{D}))$; $f \leftarrow$ d4f(d_{svp});
  $\mathbf{B} \leftarrow \texttt{Pump}(\mathbf{B}, d - d_{\text{svp}}, d_{\text{svp}}, f)$;
  return $\mathbf{v} \leftarrow \mathbf{b}_0$;

Algorithm 1: Improved Progressive PnJBKZ

4 Simulators in Two-step Mode of Solving uSVP$_\gamma$

In this section, we construct the simulators we need to design the optimized Blocksize and Jump Strategy for obtaining the expected minimum cost of the Two-step mode of solving uSVP$_\gamma$. Specifically, in Sec. 4.1, we first give the construction of the PnJBKZ simulator and show the validation experiments of the PnJBKZ simulator. Then we give the Pump cost model and the sieving dimension estimation of the last Pump in Two-step solving mode in Sec. 4.2.

Set Gram-Schmidt vectors reduced by one tour of BKZ-β and PnJBKZ(β, J) respectively as $l_i^{'}$=ln$(\|\mathbf{b}_i^{*'}\|)$ and $l_i^{''}$=ln$(\|\mathbf{b}_i^{*''}\|)$, for $i \in \{0, \cdots, d-1\}$. Denote BKZSim by the BKZ simulator proposed in [10]. Denote our PnJBKZ simulator as PnJBKZSim to simulate the change of lattice basis $\mathbf{B}$ after calling a PnJBKZ.

4.1 PnJBKZ Simulator

The first step in the Two-step solving mode is using a series of well-chosen PnJBKZ(β, J, t) to reduce the lattice basis. As we shown in Fig. 6, a reduction strategy of PnJBKZ with jump>1 can improve the reduction efficiency and how to properly choose the PnJBKZ reduction parameters (β, J, t) is the key. The reduction strategy of PnJBKZ with jump>1 shown in Fig. 6 is a heuristic reduction strategy that can be improved. Thus, it is necessary to construct a polynomial time-accurate PnJBKZ simulator that can accurately predict the reduction effort of the PnJBKZ algorithm. Since the time cost of a PnJBKZ is exponential according to β, we can not find the optimized reduction strategy by actually running all possible reduction strategies in practice. Especially, when trying to solve high dimension LWE challenges, the β needed is quite big.

The PnJBKZ Simulator Construction. Before we give the detailed construction of PnJBKZ simulator, let's briefly review the main idea of the BKZ simulator proposed in [10], which uses Gaussian Heuristic to predicate BKZ-β.

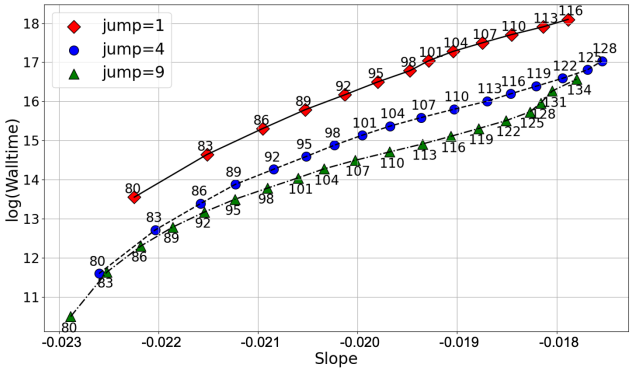


Fig. 6: Efficiency Speedup in Reduction by Jump strategy.Test on a 252-dimension lattice basis. The wall time and slope are averaged over 5 instances for each test. Each instance ran on machine C with 2 GPUs, and 32 threads. The points are labeled by β.

Let $\mathcal{L}(\mathbf{B}^{(i)})$ to represent the lattice basis after the first i blocks' reduction and $\mathcal{L}(\mathbf{B}^{(0)})$ be the initial lattice basis. For $\forall i \in [1, d]$, set $l_i = \ln \|\mathbf{b}_i^*\|$ and $l_i^{'}$ be the ln value of $\|\mathbf{b}_i^*\|$ after one tour reduction of BKZ-β. The BKZ simulator proposed in [10] first will calculate $\text{Sim}(l_0^{'}) = \ln\left(\text{GH}(\mathcal{L}(\mathbf{B}^{(0)}_{\pi[0:\beta-1]})\right) \approx \frac{1}{2}\ln(\beta/2\pi e) + \frac{1}{\beta}\ln\left(\text{Vol}(\mathcal{L}(\mathbf{B}^{(0)}_{\pi[0:\beta-1]}))\right)$ under Gaussian Heuristic (GH). Then it calculates $\text{Sim}(l_1^{'}) := \ln\left(\text{GH}(\mathcal{L}(\mathbf{B}^{(1)}_{\pi[1:\beta]}))\right) \approx \frac{1}{2}\ln(\beta/2\pi e) + \frac{1}{\beta}\left(\ln \text{Vol}(\mathcal{L}(\mathbf{B}^{(0)}_{\pi[0:\beta]})) - \text{Sim}(l_0^{'})\right)$ under GH and the information of $\text{Sim}(l_0^{'})$. Since the insertion of new $\mathbf{b}_0$ will lead to the changes of the length of Gram-Schmidt (GS) vectors, i.e. for $\forall i \in \{2, ..., d\}$, l_i changes to some unknown values. In particular, $\text{Vol}(\mathcal{L}(\mathbf{B}^{(1)}_{\pi[0:\beta]}))=\text{Vol}(\mathcal{L}(\mathbf{B}^{(0)}_{\pi[0:\beta]}))$, but l_0 changes to $l_0^{'}$ after the insert of new $\mathbf{b}_0$. So $\text{Vol}(\mathcal{L}(\mathbf{B}^{(0)}_{\pi[1:\beta]})) = \prod_{i=1}^{\beta}\|\mathbf{b}_i^*\|$, after the insert of new $\mathbf{b}_0$ it will change to $\text{Vol}(\mathcal{L}(\mathbf{B}^{(1)}_{\pi[1:\beta]})) = \left(\prod_{i=0}^{\beta}\|\mathbf{b}_i^*\|\right)/\|\mathbf{b}_0^{*'}\|$. Here $\text{Sim}(l_0^{'})$ is a simulated approximate value of $l_0^{'}$ by GH. Iteratively calculating all remaining unknown $\text{Sim}(l_i^{'})$ by $\text{Sim}(l_i^{'}) = \ln\left(\text{GH}(\mathcal{L}(\mathbf{B}^{(i)}_{\pi[i:i+\beta-1]}))\right) \approx \frac{1}{2}\ln(\beta/2\pi e) + \frac{1}{\beta}\left(\ln \text{Vol}(\mathcal{L}(\mathbf{B}^{(0)}_{\pi[0:i+\beta-1]})) - \sum_{j=0}^{i-1}\text{Sim}(l_j^{'})\right)$, for $\forall i \in [0, d-\beta]$. For the last β $\text{Sim}(l_i^{'})$ it gradually decrease the blocksize to 2. Then such a simulator can predict the value of each $l_i^{'}$ in $\mathbf{B}^{*'}$ which is reduced by one tour of BKZ-β.

However, the BKZ 2.0 simulator [10] cannot be used directly to simulate the behavior of PnJBKZ when jump>1. Set jump $= J$. Let $\mathcal{L}(\mathbf{B}^{(i)})$ to represent the lattice basis after the first i vector was inserted in i-th position of GS vectors and $\mathcal{L}(\mathbf{B}^{(0)})$ be the initial lattice basis. We observe that when $J > 1$, each time after a new $\mathbf{b}_i^*$ inserting at the first position of the block $\mathbf{B}^{(i)}_{\pi[i:k]}$, the $J-1$ norms of Gram-Schmidt vectors $\mathbf{b}_{i+1}^*, ..., \mathbf{b}_{i+J-1}^*$ will change and remain unknown. These unknown norms prevent the BKZ 2.0 simulator [10] from predicting the norm of the first Gram-Schmidt vector in the next block. Our idea is that when $J > 1$, we

use the property of HKZ reduced lattice basis to predict these unknown norms between adjacent blocks. Because as the description of `Pump` given in Section 4.1 of G6K [15], it illustrates that when turning on sieving during the pump-down stage, one can ideally obtain an HKZ reduced lattice basis after the reduction of a `Pump`. Meanwhile, turning on sieving during the pump-down stage already be the default operation in the implementation of the GPU version of G6K [24]. We conclude this result in Heuristic 1. Later our experiments will show that in practice, Heuristic 1 indeed holds under the property reduction parameter.

Heuristic 1 (Pump outputs HKZ reduced basis) *Given a d-dimensional lattice basis B with reduction quality that slope equals s. Under reduction parameter $J \leq \mathrm{d4f}_{slope}(s) \ll \beta \leq d$. For $\kappa \equiv 1 \mod J$, $\kappa < d-1$, $\mathrm{B}_{\pi[\kappa,\min\{\kappa+\beta,d\}]}$ reduced by a `Pump`$(\mathrm{B}_{\pi[\kappa,\min\{\kappa+\beta,d\}]}, \kappa, \beta, f \leq \mathrm{d4f}_{slope}(s))$, the first J vectors in each block $\mathrm{B}_{\pi[\kappa,\min\{\kappa+\beta,d\}]}$ are almost HKZ reduced, i.e. under Heuristic 4, for $i \in [\kappa, \kappa+J-1]$, the expected norms of $\|\mathbf{b}_i^*\| \approx \mathrm{GH}(\mathcal{L}(\mathbf{B}_{\pi[i:\min\{\kappa+\beta,d\}]}))$.*

Here $\mathrm{d4f}_{slope}(s)$ is an upper bound of the d4f value estimation function based on the quality of the current lattice basis proposed in [34]. $\mathrm{d4f}_{slope}(s) := \ln_\delta \sqrt{4/3} \approx \ln(4/3)/(-s)$. More detail about $\mathrm{d4f}_{slope}(s)$, see the Eq. (5) in [34].

Let l_i'' be the logarithm of each Gram-Schmidt norm after the reduction of one tour of PnJBKZ$(\beta,\ J)$. Under GH, we can simulate each l_i'' and the simulation values denoted as $\mathrm{Sim}(l_i'') = \ln\left(\mathrm{GH}\left(\mathcal{L}(\mathbf{B}^{(i)}_{\pi[i:k]})\right)\right)$. Here $k = i - (i \mod J) + \beta$ when $i \in [0, d-\beta-1]$. $k = d$, when $i \in [d-\beta, d-1]$.

The key is how to calculate the volume of $\mathcal{L}(\mathbf{B}^{(i)}_{\pi[i:k]})$. Same as BKZ during the process of PnJBKZ reduction the volume of $\mathcal{L}(\mathbf{B}^{(i)}_{\pi[0:k]})$ equals $\mathcal{L}(\mathbf{B}^{(0)}_{\pi[0:k]})$. Suppose we already know $\mathrm{Sim}(l_j'')$, for $\forall j \in \{0, ..., i-1\}$, then we calculate $\ln\left(\mathcal{L}(\mathbf{B}^{(i)}_{\pi[i:k]})\right) := \ln\left(\mathrm{Vol}\left(\mathcal{L}(\mathbf{B}^{(0)}_{\pi[0:k]})\right)\right) - \ln\left(\mathrm{Vol}\left(\mathcal{L}(\mathbf{B}^{(i)}_{\pi[0:i-1]})\right)\right) = \sum_{j=0}^{k} l_j - \sum_{j=0}^{i-1} \mathrm{Sim}(l_j'')$. Here $k = i-(i \mod J)+\beta$ when $i \in [0, d-\beta-1]$, and $k = d$ for rest cases. Under GH, for $i \in [0, d-1]$, we can iteratively calculate $\mathrm{Sim}(l_i'') :=$

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

In other words, we only need to input the initial Gram-Schmidt norms $l_i = \ln(\|\mathbf{b}_i^*\|)$, $i \in \{0, \cdots, d-1\}$ of the lattice basis. Without performing PnJBKZ reduction, we can simulate l_i'' by Eq. (1), which describes the change of lattice basis after each tour of PnJBKZ$(\beta,\ J)$. Here l_i'' are these actual Gram-Schmidt vector norms of lattice base after reducing by one tour of PnJBKZ$(\beta,\ J)$. We give a detailed algorithm description of the PnJBKZ simulator in the Alg. 2.

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

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

Algorithm 2: PnJBKZ Simulator

Besides, we need to remind that in simulating the length value of GS vectors, when $i \equiv 1 (\mathrm{mod}\ J)$, the index i of $\mathrm{GH}\left(\mathcal{L}\left(\mathbf{B}^{(i)}_{\pi[i:i+\beta-(i \bmod J)]}\right)\right)$ in our simulator is the same as that of $\mathrm{GH}\left(\mathcal{L}\left(\mathbf{B}^{(i)}_{\pi[i:i+\beta]}\right)\right)$ in the BKZ-2.0 simulator, the calculation form looks same in two simulators. However, the simulated volumes of projected sublattice are different in the two simulators. Because in BKZ 2.0 simulator [10] $\mathrm{Vol}(\mathcal{L}'\left(\mathbf{B}^{(i)}_{\pi[i:i+\beta]}\right) = \prod_{j=0}^{i+\beta-1}\|\mathbf{b}_j^*\| / \prod_{j=0}^{i-1}\|\mathbf{b}_j^{*'}\|$ 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 $\mathrm{Vol}(\mathcal{L}''\left(\mathbf{B}^{(i)}_{\pi[i:i+\beta]}\right) =$

$\prod_{j=1}^{i+\beta-1}\|\mathbf{b}_j^*\|/\prod_{j=1}^{i-1}\|\mathbf{b}_j^{*'}\|$ and $\|\mathbf{b}_j^{*'}\|$ obtained from Eq. (1). So when $i \equiv 1 (\mathrm{mod}\ J)$ the calculation of GH $\left(\mathcal{L}\left(\mathbf{B}_{\pi[i:i+\beta]}^{(i)}\right)\right)$ is different in different simulators.

Considering the influence of optimistic d4f used in PnJBKZ in practice The above discussion did not consider the influence of using dimension for free (d4f) technology. The implementation of PnJBKZ used in [15] and [24] default use d4f technology to improve the efficiency of reduction. However, compared with the theoretical d4f estimation in [23], the default d4f function used in the implementation of PnJBKZ (both [15] and [24]) is an optimistic heuristic setting. See Section 2.2. Such an optimistic d4f setting in the implementation of G6K leads to the actual reduction effort of a PnJBKZ(β, J) with the optimistic d4f setting is more closed to a PnJBKZ(β', J) with the theory d4f estimation value rather than a PnJBKZ(β, J) with the theory d4f estimation value. Here $\beta' \leq \beta$.

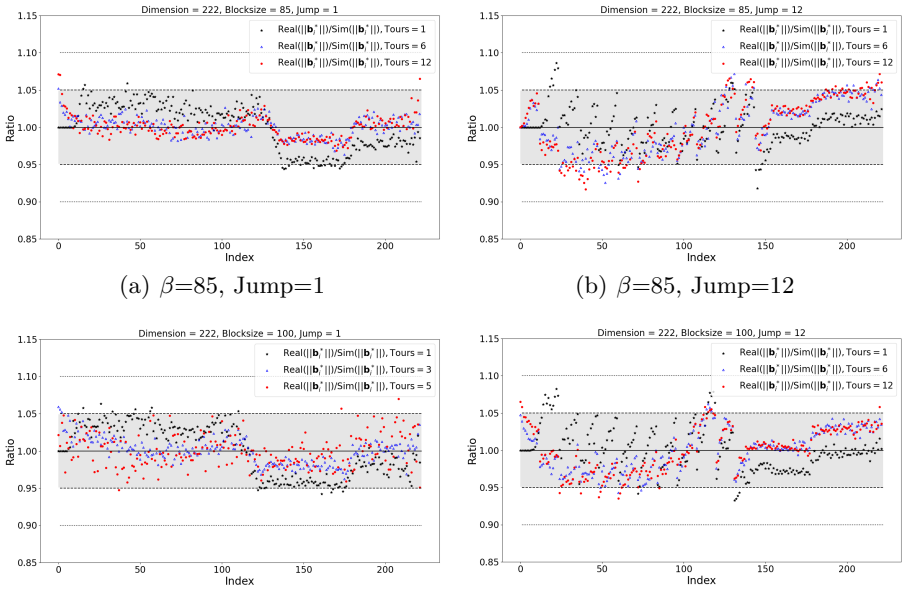
However, the PnJBKZ(β, J) with the optimistic d4f setting is quite efficient in practice. Meanwhile, our PnJBKZ simulator is designed for predicting the behavior of PnJBKZ(β, J) with the theory d4f estimation value. Therefore, to more accurately predicate the behavior of the PnJBKZ which uses the optimistic d4f function, we give the following simulation strategy. Specifically, for each blocksize β, we calculate $\beta_{sim} = \beta - d4fgap(\beta)$, here $d4fgap(\beta) = 0$, if β<40; $d4fgap(\beta) = \lfloor \beta - 40/2 - \beta\ln(4/3)/\ln(\beta/2\pi e)\rfloor$, if $40 \leq \beta \leq 75$; $d4fgap(\beta) = \lfloor 11.5 + 0.075\beta - \beta\ln(4/3)/\ln(\beta/2\pi e)\rfloor$, if β>75. The function $d4fgap(\beta)$ aims to calculate the gap between the optimistic d4f setting in the implementation of G6K and the theory d4f estimation in [23]. Then let $\beta_{sim} = \beta - d4fgap(\beta)$ as the blocksize input of Alg. 2 when using PnJBKZ simulator. Such a simulation strategy is based on the theoretical d4f estimation function proposed in [23] to adjust the over-optimistic d4f used in the default implement of PnJBKZ [15].

Furthermore, when predicting the behavior of the PnJBKZ with a quite big jump value which close to the upper bound of the d4f value under the quality of current lattice basis, by using the more refined d4f value estimation proposed in [34], we can further give more refined PnJBKZ simulator to predicate the behavior PnJBKZ which uses the optimistic d4f function. Specifically, using the information of the quality of the current lattice basis, like the slope value s, [34] calculates the refined d4f value estimation by $\mathrm{d4f}_{slope}(s) = \ln(4/3)/(-s)$. See Section 2.2 or [34] for more details about this estimation. In this case $J \leq \mathrm{d4f}_{slope}(s)$, we calculate $\beta_{sim} = \beta - \mathrm{d4f}_{slope}(s)$, and replace each blocksize β by β_{sim} when calling Alg. 2. Finally, the comparison of PnJBKZ estimations under different simulation strategies can be seen in Section 3.2 of [34]. The comparison results indicate that using the simulation strategy we mention above, can be more accurate in predicting the behavior of PnJBKZ which uses the optimistic d4f function. In the rest part of this paper, we default use the above simulation strategies to predicate the practical reduction effect of the PnJBKZ(β, J) which uses the optimistic d4f setting in practice.

Verification experiments of Heuristic 1 and PnJBKZ Simulator In this part, we show that Heuristic 1 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 particular, we set the upper bound of jump value J to be $\mathrm{d4f}_{slope}(s)$. First of all, in the implementation of `Pump`, `Pump` inserts the short vector only in the first $\mathrm{d4f}(\beta)$ index in each block. Here $\mathrm{d4f}(\beta)$ actually is the optimistic d4f value. Since $\mathrm{d4f}_{slope}(s)$ is the upper bound of d4f value under the quality of current lattice basis with slope value s, any jump value J bigger than this value will lead to that there are always $J - \mathrm{d4f}_{slope}(s) > 0$ vectors in the front of each block reduced by a `Pump` no longer satisfied the property of HKZ reduced. Therefore, only when $J \leq \mathrm{d4f}_{slope}(s)$ we can ensure Heuristic 1 is held and our PnJBKZ simulator is accurate. Next, we give numerous experiments to verify this.



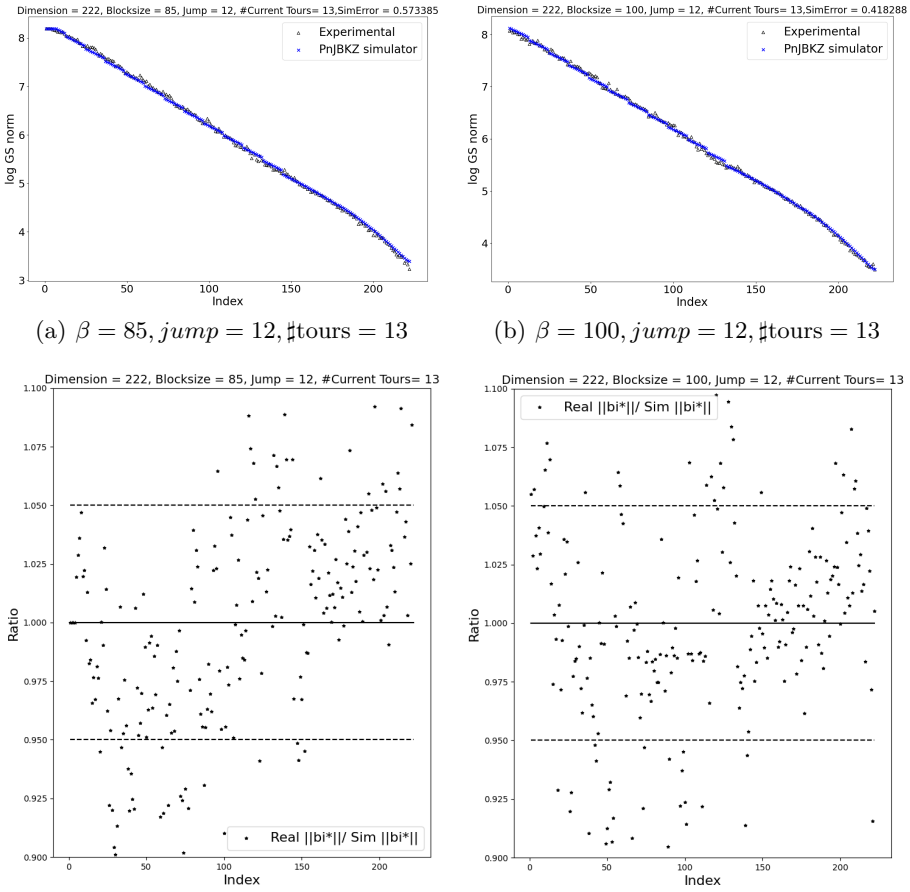
(a) β=85, Jump=1 (b) β=85, Jump=12

(c) β=100, Jump=1 (d) β=100, Jump=12

Fig. 7: Ratio $l_i''/\mathrm{Sim}(l_i'')$. Run PnJBKZ$(\beta, J)$ reduction on a 222-dimension LWE lattice basis (n=60,α=0.010) and record the ratio values. We test 20 times for each reduction parameter.

In this part, our experiments tested on the TU Darmstadt LWE Challenge lattice basis with parameter (n=60, α=0.010), and before running the PnJBKZ simulator we did small block reduction to remove the influence of q-ary vectors in the LWE Challenge lattice basis. After our pre-processing, we obtain a 222-dimension lattice basis which has a few q-ary vectors in the front of the lattice basis with a slope value equal to -0.0248 (the walltime of such a pre-processing within a few minutes). Then we calculate the ratio $l_i''/\mathrm{Sim}(l_i'')$ for $i \in [0, d-1]$ in each tour of PnJBKZ's reduction, see Fig. 7. Here l_i'' is the average logarithms of these Gram-Schmidt vector lengths obtained from 20 independent reduction experiments that use the same reduction parameter $(\beta, J, tours)$ do 20 times Pn-

JBKZ reduction respectively, and $\mathrm{Sim}(l_i'')$ is the simulated logarithm of lengths of Gram-Schmidt vector which are calculated by Eq. (1).



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

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

Fig. 8: Overall Prediction effect of PnJBKZ simulator. Ratio $l_i''/\mathrm{Sim}(l_i'')$. We perform the experiments on a reduced lattice basis of LWE Challenge (n=60,α=0.010) with slope value s=−0.0248 when jump increasing to the minimum theoretical upper bound $\lceil \mathrm{d4f}_{slope}(s) \rceil$=12. We test also 20 times for each reduction parameter and show the average value of the experiments.

We calculate the $\mathrm{Sim}(l_i'')$ strictly according to the property of the HKZ reduction basis and Heuristic 4. Therefore, in addition to being one of the criteria for measuring the accuracy of the PnJBKZ simulator, this ratio $l_i''/\mathrm{Sim}(l_i'')$ can also be used as a criterion for judging whether Heuristic 1 is held. In particular, it can be seen from Fig. 7 that for β from 85 increasing to 100 and jump from 1 increasing to 12, which is the minimum theoretical upper bound value under the current quality of lattice basis: $\mathrm{d4f}_{slope}(s = -0.0248) \approx 11.6 \leq 12$. When $\mathsf{jump} \leq \lceil \mathrm{d4f}_{slope}(s = -0.0248) \rceil = 12$, even the tours increase to 12, all most

of the ratios $l_i''/\mathrm{Sim}(l_i'')$ are within range: [0.95,1.05] (the rest ratios are also within range [0.90,1.10]), indicating that Heuristic 1 is held when $J \leq \mathrm{d4f}_{slope}(s)$. Moreover, this also is verified by (e) and (f) in Fig. 8 when the tours increase to 13. Meanwhile, the PnJBKZ simulator using Eq. (1) as the approximate estimate of the actual value l_i'' can already reflect how the average of the norms of Gram-Schmidt vectors change during each tour's reduction of PnJBKZ(β, J) which uses the optimistic d4f setting in practice.

We set $\mathrm{SimError}(\sharp\mathrm{tours}) = \sum_{i=0}^{d-1} \left(\|\mathbf{b}_i^*\|_{(\sharp\mathrm{tours})} - \mathrm{Sim}(\|\mathbf{b}_i^*\|)_{(\sharp\mathrm{tours})}\right)^2$, where $\sharp$tours represents the number of current tours and $\mathrm{Sim}(\|\mathbf{b}_i^*\|)_{(\sharp\mathrm{tours})}$ are the lengths of Gram-Schmidt vectors predicted by PnJBKZ simulator with $\sharp$tours. Then we give Fig. 8 which shows that the overall prediction error of the PnJBKZ simulator with different jumps is similar to that of jump=1. More verification experiment results with different reduction parameters on different lattice bases can be seen in Appendix C. In addition, Table 6 and Appendix H also show that reduction on the LWE challenge lattice basis under completely different reduction parameters (different blocksize and jump size strategies), the quality of lattice basis: slope value predicted based on our PnJBKZ simulator is very close to that of obtained through actual reduction. Therefore, Table 6 and Appendix H also indicate that the PnJBKZ simulator can accurately predicate the actual average reduction effect of PnJBKZ with $1 \leq J \leq \mathrm{d4f}_{slope}(s)$. For selecting the optimized reduction strategy PnJBKZ simulator is accurate enough.

4.2 Pump Cost Model and Expected Cost for Last Pump

Pump Cost Model Given sieve dimension β, we set $T_{\mathrm{Pump}}(\beta)$ as the time cost of a `Pump` since the main cost of a β-dimensional `Pump` can be regarded as a β-dimensional progressive sieve, i.e. $T_{\mathrm{Pump}}(\beta)=\sum_{j=\beta_0}^{\beta} T_{\mathrm{sieve}}(j) \leq 2^{c\cdot\beta_0} \cdot 2^{c\cdot(\beta+1)+o(\beta+1)}/(1-2^c) \approx 2^{c\cdot\beta+c_1}$. β_0 is the dimension of initial sieving in `Pump` (β_0 set as 30 in G6K, 50 in G6K-GPU), $T_{\mathrm{sieve}}(j)$ is the cost of one j-dimensional sieve, c and c_1 are the coefficients of time cost related to sieve algorithm.

The G6K (or its GPU version) implementation requires $O(\beta)$ memory and $O(\beta)$ computational cost to generate the SimHash value, which is used to find the nearest neighbor of each vector. Thus, an $O(2^{c\beta})$-time and $O(2^{c_2\beta})$-space algorithm actually requires $O(2^{c\beta} + n \cdot 2^{c_2\beta})$. Set $c = 0.367$ and $c_2 = 0.2075$ according to Fig. 7 in [24] and construct the practical `Pump` time cost model as $T_{\mathrm{Pump}}(\beta) = a_1 \cdot 2^{c\beta+c_1} + a_2 \cdot 2^{c_2\beta+c_3}$, then we can obtain the practical `Pump` time cost model (as Fig. 23 shown) through the curve fitting method. More detail of our practical `Pump` cost model shown in the Appendix D.

Sieving Dimension Estimation and Expected Cost for Last Pump. Except for PnJBKZ, solving uSVP_γ requires determining the sieving dimension and expected PSC in the final `Pump` step. We propose PumpDimEst (Alg. 3) for estimating the sieving dimension and PSC to solve general uSVP_γ with arbitrary target vector distributions.

Specifically, $F(\star, \mathcal{D})$ as the input value of Alg. 3 is a distribution function, which describes the probability distribution of the norm of the target vector

projected on the β-dimensional projected sub-lattice. Let $\star = \beta$, β represents the dimension β of projected lattice $\mathcal{L}_{\pi[d-\beta:d]}$ and $\mathcal{D}$ is the distribution of the squared length $\|\pi_{d-\beta}(\mathbf{t})\|$ of the projected target vector $\pi_{d-\beta}(\mathbf{t}) \in \mathcal{L}_{\pi[d-\beta:d]}$. The idea, which considers the norm of the projected target vector on the sub-lattice as a random variable rather than an expected value, was first proposed in [36].

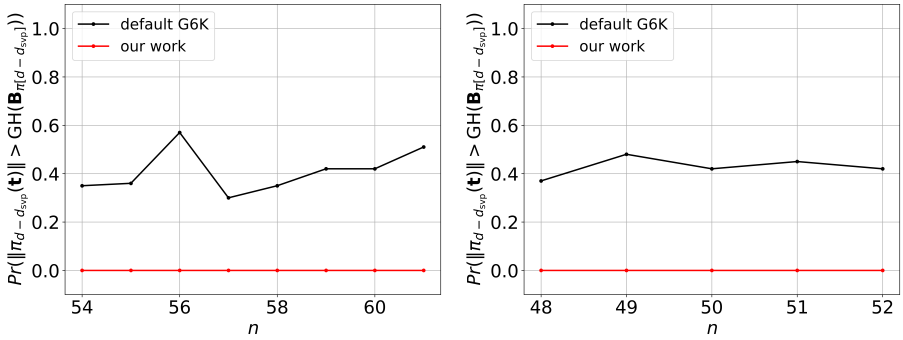
input : rr, $F(\star, \mathcal{D})$;
output: d_{svp}, PSC;
Function `PumpDimEst(rr, F(⋆, D))`:
  for $d_{\text{svp}} \leftarrow d_{\text{start}}$ **to** d **do**
    $P_{\text{suc}}(d_{\text{svp}}) \leftarrow \Pr\left[x \leftarrow F(d_{\text{svp}}, \mathcal{D}) : \ x \leq \left(\text{GH}(\text{rr}_{[d-d_{\text{svp}}:d]})\right)^2\right]$;
    if $P_{\text{suc}}(d_{\text{svp}}) \geq 0.999$ **then**
      return d_{svp}, $\text{PSC}(d_{\text{svp}})$;

Algorithm 3: Dimension and PSC Estimation for `Pump` on solving uSVP_γ.

In Alg. 3, the expected `Pump` Solvable Cost $\text{PSC}(d_{\text{svp}})$ is the estimated expected cost of the final `Pump`. Considering the progressive sieve in `Pump`, we account for both failure and success probabilities. The success probability of a β-dimension progressive sieve denoted as $P_{\text{suc}}(\beta)$. The event E_β means finding the target vector precisely at dimension β during sieving with success probability $\Pr(\text{E}_\beta)$=$P_{\text{suc}}(\beta) - P_{\text{suc}}(\beta-1)$. The expected time cost of E_β is $T_{\text{Pump}}(\beta - \text{d4f}(\beta)) \cdot \Pr(\text{E}_\beta)$. Iterating β from β_0 to d_{svp}, then $\text{PSC}(d_{\text{svp}})$=$\sum_{\beta=\beta_0}^{d_{\text{svp}}}[T_{\text{Pump}}(\beta - \text{d4f}(\beta)) \cdot (P_{\text{suc}}(\beta) - P_{\text{suc}}(\beta-1))]$, where $P_{\text{suc}}(d_{\text{svp}}) \geq$0.999.

In the case of solving standard form LWE, the fastest solving algorithm for it is the primal attack. The primal attack solves LWE problem by transforming it to uSVP_γ Problem and then call a uSVP_γ solver to solve it. Its target vector $\mathbf{t} = (\mathbf{e}, \pm 1)$ is a combination of ± 1 and the noise vector $\mathbf{e} \in \mathbb{Z}_q^m$ from a discrete Gaussian distribution $N(0, \sigma^2)$ with standard deviation σ. Then, the probability distribution of squared norm of target vector in β-dimensional sub-lattice can be described as $F(\beta, N(0, \sigma^2)) = \sigma^2 \cdot \chi^2_\beta$.

As mentioned in Sec. 6.4 in G6K paper [15], the estimation of d_{svp} in the default G6K uses the estimation originally given in [14] and experimentally justified in [32]. It computes the expected norm of the projected target vector $\|\pi_{d-d_{\text{svp}}}(\mathbf{t})\| \approx \sigma\sqrt{d_{\text{svp}}}$. It declared that it satisfies the condition $\sigma\sqrt{d_{\text{svp}}} \approx \|\pi_{d-d_{\text{svp}}}(\mathbf{t})\| \leq \text{GH}(\mathbf{B}_{\pi[d-d_{\text{svp}}]})$, then the projected shortest vector may be in the projected sub-lattice. It outputs the minimum value d_{svp} such that the inequality $\sigma\sqrt{d_{\text{svp}}} \leq \text{GH}(\mathbf{B}_{\pi[d-d_{\text{svp}}]})$ holds and take it as the upper bound of sieving in the `Pump`. Fig. 9 shows that even if $\sigma\sqrt{d_{\text{svp}}} \leq \text{GH}(\mathbf{B}_{\pi[d-d_{\text{svp}}]})$, $\|\pi_{d-d_{\text{svp}}}(\mathbf{t})\|$ is possibly larger than $\text{GH}(\mathbf{B}_{\pi[d-d_{\text{svp}}]})$, i.e. estimating the upper bound of `Pump` by the expected value is over-optimistic. The red line shows that the norm of projected vector of our estimated dimension value satisfies the condition $\|\pi_{d-d_{\text{svp}}}(\mathbf{t})\| \leq \text{GH}(\mathbf{B}_{\pi[d-d_{\text{svp}}]})$ by testing 100 trials of LWE instances. Because $\|\mathbf{e}\|^2$ is a randomly positive variable following chi-squared distribution rather than a fixed value. It is more reasonable to consider a high success probability ($\geq$0.999) for recovering the target vector with a suitable sieving dimension by setting $F(\beta, \mathcal{D}) = \sigma^2\chi^2_\beta$ in Alg. 3 to solve LWE problem.



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

Fig. 9: The failure probability of the estimated dimension for last `Pump` using LWE instances. For each (n, α), 100 randomly LWE instances are generated. The black line shows that using the estimated dimension [14,32] in Sec. 6.4 in G6K paper [15] has a non-negligible probability s.t. $\|\pi_{d-d_{\text{svp}}}(\mathbf{t})\| > \text{GH}(\mathbf{B}_{\pi[d-d_{\text{svp}}]}) \geq \sigma\sqrt{d_{\text{svp}}}$, which makes it fail to find the projection of the target vector because the projected norm of $\mathbf{t}$ is larger than the norm of the shortest vector in $\mathcal{L}_{\pi[d-d_{\text{svp}}]}$, i.e. $\Pr(\|\pi_{d-d_{\text{svp}}}(\mathbf{t})\| > \text{GH}(\mathbf{B}_{\pi[d-d_{\text{svp}}]}) : d_{\text{svp}}$ estimated from [14] $) > 0$. The red line shows that using the estimated dimension computed by Alg. 3 in our work, the condition $\|\pi_{d-d_{\text{svp}}}(\mathbf{t})\| \leq \text{GH}(\mathbf{B}_{\pi[d-d_{\text{svp}}]})$ can always be satisfied, i.e. $\Pr(\|\pi_{d-d_{\text{svp}}}(\mathbf{t})\| > \text{GH}(\mathbf{B}_{\pi[d-d_{\text{svp}}]}) : d_{\text{svp}}$ estimated from Alg.3 $) = 0$.

5 Blocksize and Jump Strategy Optimization for ProPnJBKZ

In this section, we detailly present the strategy selection algorithm EnumBS. Details about BSSA are available in Appendix E. It is a novel approach for deriving a blocksize and jump strategy with minimum simulated time cost. We provide both formal proofs (based on the Estimator Stability Assumption proposed in Sec. 5.1) and experimental results demonstrating its superior performance compared to the ProBKZ-based algorithm in solving uSVP$_\gamma$ instances in less time.

5.1 Estimator Stability Assumption

In this part, we give some heuristic assumptions to support the strategy generation algorithms which will be discussed in Sec. 5. Since the theoretical time cost of the sieving algorithm is related only to the lattice dimension, we assume that our time cost model fits well for almost all lattices. Thus, we give Heuristic 2.

Heuristic 2 *The time cost of the practical time cost model (Appendix D) is close to the actual time cost of the* `Pump` *algorithm and PnJBKZ algorithm.*

In addition to the stability of the time cost, the reduction effect of a fixed PnJBKZ(β, J) is also stable. Same as BKZ-β, the quality of the lattice basis is

not always improved by using a fixed PnJBKZ(β, J) to do the reduction. Based on the analysis of using the dynamical system, [37] already demonstrated that using a fixed PnJBKZ(β, J) tour repeatedly for reduction can only gradually approach a certain reduction basis. In particular, after running enough reduction tours of a fixed PnJBKZ(β, J), the norm values of each vector in Gram Schmidt lattice basis gradually approaches PnJBKZ(β, J) fully reduced fixed values which related to the volume of lattice basis, more detail see Section 3.2 of [37]. Besides Section 4 of [37] also proves that the reduction effect of a fixed PnJBKZ(β, J) on the norm of each Gram Schmidt vector of lattice basis does converge to some certain values which relate to the volume of lattice basis and [37] gave an estimation of the maximum number of tours required for convergence. Then we give the following proposition. For convenience, here we use slope as the standard to measure the quality of a lattice basis reduced by the lattice reduction algorithm (other standards to measure the quality of a lattice basis are also held).

Proposition 1. *For a fully PnJBKZ(β, J) reduced lattice basis* $\mathbf{B}$ *with current slope* s*. Then, the slope of* $\mathbf{B}$ *can not be further improved better than* s *if one only uses PnJBKZ(β_{new}, J_{new}) tour to do reduction for any* $\beta \geq \beta_{new}$, $J_{new} \geq J$.

5.2 Blocksize and Jump Strategy Enumeration Algorithm

The advantage of BSSA is that the algorithm runs in polynomial time. However, there are also some disadvantages, we list two of them.

First, the reduction effect of a BSSA strategy is not estimated accurately. Each intermediate result of lattice basis is fitted into one of the nodes, which are some BKZ-β reduced basis, but the actual basis may have better quality than the node. Therefore, a BSSA strategy may overestimate the time cost to reach a BKZ-β reduced basis and we could explore ways to develop a more efficient reducing strategy. Second, BSSA also misses many potential optimized strategies. For example, BSSA only considers improving a BKZ-β reduced basis to a BKZ-(β+1) reduced basis by using a fixed blocksize of BKZ-β' tour(s) reduction, β'>(β+1). However, the reduction with minimum expected time cost between the two nodes might be a progressive reduction that blocksize progressive increased or jump progressive decreased rather than blocksize fixed reduction, which cannot be found by BSSA. The problems can be solved by using a blocksize and jump strategy enumeration algorithm abbreviate as EnumBS to enumerate all possible blocksize and jump reduction strategies, so we can find the strategy with minimum simulated time cost among them. We use the "simulated time cost" since we generate the strategy through the practical time cost (Appendix D), which remains a slight disparity from the actual cost.

However, an enumeration algorithm is inefficient, and cannot be used in practice since it costs an exponential time overhead. Based on Heuristic 3, we improve the algorithm by pruning unnecessary strategies in the branch and bound.

Heuristic 3 *Given two sufficiently reduced lattice bases* $\mathbf{B}, \mathbf{B}'$ *with Gram-Schmidt basis lengths* $\mathsf{rr}, \mathsf{rr}'$. *If* $\mathrm{PSC}(\mathsf{rr}) \geq \mathrm{PSC}(\mathsf{rr}')$, *then* $\mathrm{PSC}(\mathrm{PnJBKZSim}(\mathsf{rr}, \beta, J, 1)) \geq \mathrm{PSC}(\mathrm{PnJBKZSim}\ (\mathsf{rr}', \beta, J, 1))$ *also holds for any* β *and* $J \leq \mathrm{d4f}_{slope}(s)$.

Heuristic 3 is a variant of the Heuristic used in [12] based on for their strategy selection method. The main difference between the Heuristic 3 used in [12] and Heuristic 3 is that we use PSC rather than FEC for comparison.

input : rr_0, $F(\star, \mathcal{D})$, $J_{\max}(\star) \leftarrow \mathrm{d4f}_{slope}(\star)$;
output: $T_{\min}$, $\mathsf{S}_{\min}$;
Function `EnumBS`(rr_0, $F(\star, \mathcal{D})$, $J_{\max}(\star) \leftarrow \mathrm{d4f}_{slope}(\star)$)**:**
  $k \leftarrow 1$; $d \leftarrow \mathrm{len}(\mathsf{rr}_0)$; $\_, \mathrm{PSC}^{(0)} \leftarrow$ `PumpDimEst`(rr_0, $F(\star, \mathcal{D})$);
  $\mathsf{bs}^{(0)} \leftarrow (\mathsf{rr}_0, [\,], 0, \mathrm{PSC}^{(0)})$; $\mathsf{BS} \leftarrow \{\mathsf{bs}^{(0)}\}$;
  while $k \leq \sharp\mathsf{BS}$ **do**
    $\mathsf{bs} \leftarrow \mathsf{BS}\,[k]$; $(\beta, J) \leftarrow$ the last element of $\mathsf{BS}[k].\mathsf{S}$;
    $(\beta, J) \leftarrow$ next (β, J) s.t. $\mathrm{PSC}($`PnJBKZSim` $(\mathsf{bs.rr}, \beta, J, 1)) < \mathsf{bs.PSC}$;
    while (β, J) is not None **do**
      $\mathsf{bs}^*.\mathsf{S} \leftarrow \mathsf{S} \cup [(\beta, J)]$, update bs^* under $\mathsf{bs}^*.\mathsf{S}$;
      $\mathsf{BS} \leftarrow \mathsf{BS} \cup \{\mathsf{bs}^*\}$;
      if $\exists\ \mathsf{bs} \in \mathsf{BS}$ s.t. $\mathsf{bs}^*.\mathrm{PSC} \geq \mathsf{bs.PSC}$ **and** $\mathsf{bs}^*.T_{\mathrm{PnJBKZs}} \geq \mathsf{bs}.T_{\mathrm{PnJBKZs}}$ **then**
        $\mathsf{BS} \leftarrow \mathsf{BS} \setminus \{\mathsf{bs}^*\}$;
      else
        for $\forall \mathsf{bs} \in \mathsf{BS}$ s.t. $\mathsf{bs}^*.\mathrm{PSC} \leq \mathsf{bs.PSC}$ **and** $\mathsf{bs}^*.T_{\mathrm{PnJBKZs}} \leq \mathsf{bs}.T_{\mathrm{PnJBKZs}}$ **do**
          $\mathsf{BS} \leftarrow \mathsf{BS} \setminus \{\mathsf{bs}\}$;
      $(\beta, J) \leftarrow$ next (β, J) s.t. $\mathrm{PSC}($`PnJBKZSim` $(\mathsf{bs.rr}, \beta, J, 1)) < \mathsf{bs.PSC}$;
      $k \leftarrow k + 1$;
  $\mathsf{bs}_{\min} \leftarrow \min_{\mathsf{bs}.T_{\mathrm{PnJBKZs}} + \mathsf{bs.PSC}} \mathsf{BS}$;
  return $T_{\min} \leftarrow \mathsf{bs}_{\min}.T_{\mathrm{PnJBKZs}} + \mathsf{bs}_{\min}.\mathrm{PSC}$, $\mathsf{S}_{\min} \leftarrow \mathsf{bs}_{\min}.\mathsf{S}$;

Algorithm 4: EnumBS

The detailed description of EnumBS with pruning is shown in Alg. 4. In EnumBS, we use BS to store the information of each reduction strategy which might be the final strategy with minimum time cost or will become the final strategy after adding more (β, J) nodes. BS is a list and each element bs in the BS is a tuple of values $\mathsf{bs} = (\mathsf{rr}, \mathsf{S}, T_{\mathrm{PnJBKZs}}, \mathrm{PSC})$. It is important to note that each bs in BS should be in order that increases by its T_{PnJBKZs} value and decreases by its PSC value. In bs, S is a list storing the blocksize and jump strategy used for calling PnJBKZ, T_{PnJBKZs} is the time cost for calling such a series of PnJBKZ, rr stores the current simulated gs-lengths after analogically calling PnJBKZ following strategy S by PnJBKZ simulator (Alg. 2). PSC output from the `Pump` dimension estimation method (Alg. 3), it estimates the expected time cost for last `Pump`. Each element in S is a tuple (β, J), where β is the blocksize value of PnJBKZ and J is the jump value of PnJBKZ.

For the sake of narrative simplicity, we will use $\mathsf{bs}.\star$ to denote each element in bs, e.g. $\mathsf{bs.S}$. Let $\sharp\mathsf{S}$ and $\sharp\mathsf{BS}$ be the size of S and BS. At the start of EnumBS,

there is only one tuple $\mathsf{bs}^{(0)}$ in BS, where $\mathsf{bs}^{(0)}.\mathsf{S} = [\ ]$ denotes a no PnJBKZ blocksize and jump strategy with a pure `Pump` sieve. The total cost of $\mathsf{bs}^{(0)}$ is the `Pump` cost. Then, to generate more strategies and try to find the strategy with minimum expected time cost, we can regard $\mathsf{bs}^{(0)}$ as the root node and expand the strategy list from $\mathsf{bs}^{(0)}$ using a breadth-first search.

For a node bs in the tree, each of its children bs^* satisfies that $\mathsf{bs}^*.\mathsf{S} = \mathsf{bs}.\mathsf{S} \cup [(\beta, J)]$, where $\mathrm{PSC}(\mathrm{PnJBKZSim}(\mathsf{bs}.\mathsf{rr}, \beta, J, 1)) < \mathsf{bs}.\mathrm{PSC}$, which means that a (β, J) tour can further improve the basis quality. As the claim proposed in Proposition 1, we can begin the enumeration search from $\beta_0 + 1$ or $J_0 - 1 \geq 1$, where $\mathsf{bs}.\mathsf{rr}$ has equal or higher quality than a PnJBKZ(β_0, J_0) reduced basis.

Considering each child strategy $\mathsf{bs}^*.\mathsf{S}$ of $\mathsf{bs}.\mathsf{S}$ for all possible (β, J), compute the other values in bs^*, i.e. $\mathsf{bs}^*.T_{\mathrm{PnJBKZs}}$, $\mathsf{bs}^*.\mathsf{rr}$and $\mathsf{bs}^*.\mathrm{PSC}$. When we try to add a bs^* into BS, we should first determine whether it exists a $\mathsf{bs}' \in \mathsf{BS}$ so that $\mathsf{bs}^*.\mathrm{PSC} \geq \mathsf{bs}'.\mathrm{PSC}$ and $\mathsf{bs}^*.T_{\mathrm{PnJBKZs}} \geq \mathsf{bs}'.T_{\mathrm{PnJBKZs}}$. If so, we cannot add such bs^* into BS, because the child strategies generated by bs^*(including bs^* itself) will not have a shorter time overhead than which generated by the corresponding bs'. If not, then we should first add bs^* and then delete the bad strategy in BS whose PSC value and T_{PnJBKZs} value are both larger than bs^*. Iterate each $\mathsf{BS}[k]$ and its child nodes sequentially, and we will end up with a BS containing the blocksize and Jump strategy with minimum simulated time cost. Finally, we search through BS and return the blocksize and Jump reduction strategy which has the minimum simulated time cost in the end.

In addition, Theorem 1 proves that EnumBS can find the blocksize and jump strategy with minimum simulated cost to solve uSVP$_\gamma$ in Two-step mode.

Theorem 1. *Let $\mathcal{S}$ be the set of all sequences consisting of $(\beta, J \leq \mathrm{d4f}_{slope}(s))$, $\mathcal{S}$ is the set of all possible blocksize and jump strategies. If Heuristic 3 holds, the algorithm EnumBS always returns the reduction strategy in $\mathcal{S}$ with minimum simulated time cost to solve uSVP$_\gamma$ instance.*

Proof. Let rr_0 be the input Gram-Schmidt Lengths of a random lattice basis, Suppose that the strategy in $\mathcal{S}$ with minimum simulated cost is $\mathsf{S} = [(\beta_1, J_1), ..., (\beta_k, J_k)]$. We write the sub-strategy $[(\beta_1, J_1), ..., (\beta_i, J_i)]$ of S, $i \leq k$ as S_i.

$\mathrm{PSC}(\mathrm{PnJBKZSim}(\mathsf{rr}_0, \mathsf{S}_{i-1})) \geq \mathrm{PSC}(\mathrm{PnJBKZSim}(\mathsf{rr}_0, \mathsf{S}_i))$ for all $i \leq k$, otherwise removing (β_i, J_i) from S can get a strategy that can solve an uSVP$_\gamma$ instance in less time by Heuristic 3. From the description of EnumBS, either S is inside the final strategy set BS, or there is a sub-strategy S_i such that S_i is removed from BS (then S won't appear in BS anymore). Since S has a minimum time cost among all strategies in BS, S must be the final output strategy and meets the first case. Now we show that the second case cannot occur.

If S_i is removed from BS, then there must be another strategy S' such that $\mathrm{PSC}(\mathrm{PnJBKZSim}(\mathsf{rr}_0, \mathsf{S}_i)) \geq \mathrm{PSC}(\mathrm{PnJBKZSim}(\mathsf{rr}_0, \mathsf{S}'))$, and the PnJBKZ time cost $\mathsf{S}_i.T_{\mathrm{PnJBKZs}} \geq \mathsf{S}'.T_{\mathrm{PnJBKZs}}$. If we append the sequence $(\beta_{i+1}, J_{i+1}), ..., (\beta_k, J_k)$ into S' and get a new strategy S^*, it infers that $\mathsf{S}.T_{\mathrm{PnJBKZs}} \geq \mathsf{S}^*.T_{\mathrm{PnJBKZs}}$.

Heuristic 3 implies $\mathrm{PSC}(\mathrm{PnJBKZSim}(\mathsf{rr}_0, \mathsf{S})) \geq \mathrm{PSC}(\mathrm{PnJBKZSim}(\mathsf{rr}_0, \mathsf{S}^*))$, contradicting the expectation that S^* has a smaller time cost than S. □

6 Apply ProPnJBKZ to LWE

Based on Heuristic 2 and Proposition 1, we use BSSA and EnumBS to generate optimized strategies for blocksize and jump. This section focuses on demonstrating ProPnJBKZ's performance in solving LWE using the Two-step mode. Sec. 6.1 applies ProPnJBKZ to solve LWE and compares it with the LWE solving algorithm implemented in G6K-GPU-Tensor. In Sec. 6.2, we present the optimized reduction strategies generated by EnumBS which are used in Sec. 6.1 for solving the TU Darmstadt LWE Challenge. Besides, we give the simulated accuracy test in Sec. 6.3. Additionally, Sec. 6.4 showcases new records in solving TU Darmstadt LWE Challenges. Appendix A provides a condensed security estimation of LWE in NIST schemes based on Two-step mode and EnumBS.

6.1 Efficiency of ProPnJBKZ for LWE and Compare with G6K

The default LWE solving algorithm in G6K is the script `lwe_challenge.py` in the implementation of G6K's GPU version [24], we have discussed it in Sec. 3.1. Besides, for more detail about the default LWE solving algorithm in G6K's GPU version[1]. Fig. 2 gives the experimental result of different LWE-solving algorithms. The cream bars show the experimental time or memory cost of the default strategy in G6K. The remaining bars give the experimental time or memory cost of ProPnjBKZ using the strategy generated by EnumBS(Alg. 4) (BSSA(Alg. 5)). From the result of Fig. 2, we can see that using the strategy selected by EnumBS (BSSA) significantly decreased the wall time cost by about 7.2∼17.0 (5.2∼10.2) times compared to that of the default LWE solving strategy in G6K when all LWE solvers use the same float type "dd" to calculate. One can refer to the log files of Fig. 2 in the folders `lwechal-test` and `lwe-instance-test`. It can also be reproduced by running the test code `implement_lwechal_forall.sh` and `implement_lwe_instance_forall.sh` in source code[2].

6.2 Optimized strategy generated by ProPnJBKZ for Solving TU Darmstadt LWE Challenge

Table 5: Blocksize and Jump strategy generated by EnumBS (threads = 10).

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

We use EnumBS (Alg. 4) with the practical cost model mentioned in Appendix D and tested on machine C to select the blocksize and jump strategy for some instances of TU Darmstadt LWE Challenges, we list the selected strategies in Table 5. Besides, Table 5 shows that the time cost of generating the reduction strategy

by EnumBS is acceptable. Also, we upload the open source code for blocksize and jump strategy generation on any LWE instances in folder "strategy_gen" from source code[2]. We solved the TU Darmstadt LWE Challenge instances with $(n, \alpha) \in \{(40, 0.035), (40, 0.040), (50, 0.025), (55, 0.020), (90, 0.005)\}$ successfully by the selected strategies in Appendix G.

6.3 Simulated Accuracy of ProPnJBKZ for LWE

To show the accuracy of our optimized blocksize and jump-selected strategy, we compare the predicted quality of lattice basis and wall time with that of actual experiments in each middle node in the reduction step. Table 6 and Appendix H illustrate that both the quality of the actual lattice basis and the actual wall time of each tour of PnJBKZ(β, J) are close to our prediction.[5] It also implies that Heuristic 1 and Heuristic 2 established.

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

Table 6: Quality and log(T) during reduction of LWE $(n, \alpha) = (40, 0.030)$.

(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
(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

Table 7: Actual running time, RAM cost for LWE Challenge.

6.4 New LWE Records

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

By our new algorithm, i.e. ProPnJBKZ, we have solved the LWE instances $(n, \alpha) \in \{(80, 0.005), (40, 0.035), (90, 0.005), (50, 0.025), (55, 0.020), (40, 0.040)\}$ in TU Darmstadt LWE Challenge website[3]. See Fig. 1 for more details. Specifically we denoted a service with AMD EPYC^{TM} 7002 Series 128@2.6GHz, NVIDIA 3090 * 8, 1.5T RAM as Machine A, and denoted a service with AMD EPYC^{TM} 7002 Series 64@2.6GHz, a100 * 4, 512 GB RAM as Machine B. A workstation with Intel Xeon 5128 16c 32@2.3GHz, 1.48T RAM and NVIDIA RTX 3090 * 2, denoted as machine C. Then we listed the walltime and RAM cost in solving the above LWE Challenges in Table 7. The units of T in Tables 6 and 7 are seconds and hours, respectively.

[5] The data in Table 6 is extracted from a test in Fig. 2 for comparing the quality and wall time between our simulations and actual experiments. For more experiment results on different LWE lattice bases please see Appendix H.

References

1. L. Ducas, T. L. Eike Kiltz, V. Lyubashevsky, P. Schwabe, G. Seiler, and D. Stehlé, *Dilithium(Round 3)*. NIST PQC project, 2020.
2. R. Avanzi, J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, P. Schwabe, G. Seiler, and D. Stehlé, "Kyber(Round 3)," p. 42, 2021.
3. O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," *Journal of the ACM*, vol. 56, pp. 34:1–34:40, Sept. 2009.
4. V. Lyubashevsky, C. Peikert, and O. Regev, "On Ideal Lattices and Learning with Errors over Rings," in *Advances in Cryptology – EUROCRYPT 2010* (H. Gilbert, ed.), Lecture Notes in Computer Science, (Berlin, Heidelberg), pp. 1–23, Springer, 2010.
5. S. Bai and S. D. Galbraith, "An Improved Compression Technique for Signatures Based on Learning with Errors," in *Topics in Cryptology – CT-RSA 2014* (J. Benaloh, ed.), (Cham), pp. 28–47, Springer International Publishing, 2014.
6. R. Kannan, "Improved algorithms for integer programming and related lattice problems," in *Proceedings of the fifteenth annual ACM symposium on Theory of computing*, STOC '83, (New York, NY, USA), pp. 193–206, Association for Computing Machinery, Dec. 1983.
7. C. P. Schnorr and M. Euchner, "Lattice basis reduction: Improved practical algorithms and solving subset sum problems," in *Fundamentals of Computation Theory* (L. Budach, ed.), Lecture Notes in Computer Science, (Berlin, Heidelberg), pp. 68–85, Springer, 1991.
8. A. K. Lenstra, H. W. Lenstra, and L. Lovász, "Factoring polynomials with rational coefficients," *Mathematische Annalen*, vol. 261, pp. 515–534, Dec. 1982.
9. N. Gama, P. Q. Nguyen, and O. Regev, "Lattice Enumeration Using Extreme Pruning," in *Advances in Cryptology – EUROCRYPT 2010* (D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, J. C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, M. Sudan, D. Terzopoulos, D. Tygar, M. Y. Vardi, G. Weikum, and H. Gilbert, eds.), vol. 6110, pp. 257–278, Berlin, Heidelberg: Springer Berlin Heidelberg, 2010. Series Title: Lecture Notes in Computer Science.
10. Y. Chen and P. Q. Nguyen, "BKZ 2.0: Better Lattice Security Estimates," in *Advances in Cryptology – ASIACRYPT 2011* (D. H. Lee and X. Wang, eds.), Lecture Notes in Computer Science, (Berlin, Heidelberg), pp. 1–20, Springer, 2011.
11. M. R. Albrecht, S. Bai, J. Li, and J. Rowell, "Lattice Reduction with Approximate Enumeration Oracles," in *Advances in Cryptology – CRYPTO 2021* (T. Malkin and C. Peikert, eds.), Lecture Notes in Computer Science, (Cham), pp. 732–759, Springer International Publishing, 2021.
12. Y. Aono, Y. Wang, T. Hayashi, and T. Takagi, "Improved Progressive BKZ Algorithms and Their Precise Cost Estimation by Sharp Simulator," in *Advances in Cryptology – EUROCRYPT 2016* (M. Fischlin and J.-S. Coron, eds.), Lecture Notes in Computer Science, (Berlin, Heidelberg), pp. 789–819, Springer, 2016.
13. M. R. Albrecht, F. Göpfert, F. Virdia, and T. Wunderer, "Revisiting the Expected Cost of Solving uSVP and Applications to LWE," in *Advances in Cryptology – ASIACRYPT 2017* (T. Takagi and T. Peyrin, eds.), vol. 10624, pp. 297–322, Cham: Springer International Publishing, 2017. Series Title: Lecture Notes in Computer Science.
14. E. Alkim, L. Ducas, T. Pöppelmann, and P. Schwabe, "Post-quantum Key Exchange—A New Hope," pp. 327–343, 2016.

15. M. R. Albrecht, L. Ducas, G. Herold, E. Kirshanova, E. W. Postlethwaite, and M. Stevens, "The General Sieve Kernel and New Records in Lattice Reduction," in *Advances in Cryptology – EUROCRYPT 2019* (Y. Ishai and V. Rijmen, eds.), vol. 11477, pp. 717–746, Cham: Springer International Publishing, 2019. Series Title: Lecture Notes in Computer Science.
16. T. Laarhoven and A. Mariano, "Progressive Lattice Sieving," in *Post-Quantum Cryptography* (T. Lange and R. Steinwandt, eds.), vol. 10786, pp. 292–311, Cham: Springer International Publishing, 2018. Series Title: Lecture Notes in Computer Science.
17. D. Micciancio and P. Voulgaris, "Faster exponential time algorithms for the shortest vector problem," in *Proceedings of the 2010 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, Proceedings, pp. 1468–1480, Society for Industrial and Applied Mathematics, Jan. 2010.
18. R. Fitzpatrick, C. Bischof, J. Buchmann, Ö. Dagdelen, F. Göpfert, A. Mariano, and B.-Y. Yang, "Tuning gausssieve for speed," in *Progress in Cryptology - LATINCRYPT 2014* (D. F. Aranha and A. Menezes, eds.), (Cham), pp. 288–305, Springer International Publishing, 2015.
19. P. Q. Nguyen and T. Vidick, "Sieve algorithms for the shortest vector problem are practical," *Journal of Mathematical Cryptology*, vol. 2, Jan. 2008.
20. G. Herold and E. Kirshanova, "Improved Algorithms for the Approximate k-List Problem in Euclidean Norm," in *Public-Key Cryptography – PKC 2017* (S. Fehr, ed.), Lecture Notes in Computer Science, (Berlin, Heidelberg), pp. 16–40, Springer, 2017.
21. G. Herold, E. Kirshanova, and T. Laarhoven, "Speed-Ups and Time–Memory Trade-Offs for Tuple Lattice Sieving," in *Public-Key Cryptography – PKC 2018*, pp. 407–436, Springer, Cham, Mar. 2018.
22. A. Becker, N. Gama, and A. Joux, "Speeding up lattice sieving without increasing the memory, using sub-quadratic nearest neighbor search," 2015.
23. L. Ducas, "Shortest Vector from Lattice Sieving: A Few Dimensions for Free," in *Advances in Cryptology – EUROCRYPT 2018* (J. B. Nielsen and V. Rijmen, eds.), (Cham), pp. 125–145, Springer International Publishing, 2018.
24. L. Ducas, M. Stevens, and W. van Woerden, "Advanced Lattice Sieving on GPUs, with Tensor Cores," in *Advances in Cryptology – EUROCRYPT 2021* (A. Canteaut and F.-X. Standaert, eds.), Lecture Notes in Computer Science, (Cham), pp. 249–279, Springer International Publishing, 2021.
25. A. Becker, L. Ducas, N. Gama, and T. Laarhoven, "New directions in nearest neighbor searching with applications to lattice sieving," in *Proceedings of the twenty-seventh annual ACM-SIAM symposium on Discrete algorithms*, SODA '16, (USA), pp. 10–24, Society for Industrial and Applied Mathematics, Jan. 2016.
26. M. Liu and P. Q. Nguyen, "Solving BDD by Enumeration: An Update," in *Topics in Cryptology – CT-RSA 2013* (E. Dawson, ed.), Lecture Notes in Computer Science, (Berlin, Heidelberg), pp. 293–309, Springer, 2013.
27. W. Xia, L. Wang, G. Wang, D. Gu, and B. Wang, "A Refined Hardness Estimation of LWE in Two-Step Mode," in *Public-Key Cryptography – PKC 2024* (Q. Tang and V. Teague, eds.), (Cham), pp. 3–35, Springer Nature Switzerland, 2024.
28. M. Walter, "The convergence of slide-type reductions," in *Public-Key Cryptography – PKC 2021* (J. A. Garay, ed.), (Cham), pp. 45–67, Springer International Publishing, 2021.
29. J. Li and P. Q. Nguyen, "A complete analysis of the bkz lattice reduction algorithm," *IACR Cryptol. ePrint Arch.*, vol. 2020, p. 1237, 2020.

30. G. Hanrot, X. Pujol, and D. Stehlé, "Analyzing Blockwise Lattice Algorithms Using Dynamical Systems," in *Advances in Cryptology – CRYPTO 2011* (P. Rogaway, ed.), Lecture Notes in Computer Science, (Berlin, Heidelberg), pp. 447–464, Springer, 2011.
31. S. Bai, D. Stehlé, and W. Wen, "Measuring, Simulating and Exploiting the Head Concavity Phenomenon in BKZ," in *Advances in Cryptology – ASIACRYPT 2018* (T. Peyrin and S. Galbraith, eds.), Lecture Notes in Computer Science, (Cham), pp. 369–404, Springer International Publishing, 2018.
32. M. R. Albrecht, F. Göpfert, F. Virdia, and T. Wunderer, "Revisiting the Expected Cost of Solving uSVP and Applications to LWE," in *Advances in Cryptology – ASIACRYPT 2017* (T. Takagi and T. Peyrin, eds.), (Cham), pp. 297–322, Springer International Publishing, 2017.
33. N. Gama and P. Q. Nguyen, "Predicting lattice reduction," in *Advances in Cryptology – EUROCRYPT 2008* (N. Smart, ed.), (Berlin, Heidelberg), pp. 31–51, Springer Berlin Heidelberg, 2008.
34. L. Wang, Y. Wang, and B. Wang, "A trade-off svp-solving strategy based on a sharper pnj-bkz simulator," in *Proceedings of the 2023 ACM Asia Conference on Computer and Communications Security*, ASIA CCS '23, (New York, NY, USA), p. 664–677, Association for Computing Machinery, 2023.
35. Y. Chen and P. Q. Nguyen, *Réduction de réseau et sécurité concrète du chiffrement complètement homomorphe*. PhD Thesis, 2013.
36. D. Dachman-Soled, L. Ducas, H. Gong, and M. Rossi, "LWE with Side Information: Attacks and Concrete Security Estimation," in *Advances in Cryptology – CRYPTO 2020: 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17–21, 2020, Proceedings, Part II*, (Berlin, Heidelberg), pp. 329–358, Springer-Verlag, Aug. 2020.
37. L. Wang, "Analyzing pump and jump bkz algorithm using dynamical systems," in *Post-Quantum Cryptography – PQCrypto 2024*, 2024. https://eprint.iacr.org/2024/713, https://www.maths.ox.ac.uk/system/files/inline-files/pqc24programme.pdf.
38. I. T. L. C. S. R. CENTER, "Post-quantum cryptography pqc selected algorithms 2022." https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022.
39. MATZOV, "Report on the Security of LWE: Improved Dual Lattice Attack," Apr. 2022.
40. M. R. Albrecht, V. Gheorghiu, E. W. Postlethwaite, and J. M. Schanck, "Estimating quantum speedups for lattice sieves," in *Advances in Cryptology – ASIACRYPT 2020* (S. Moriai and H. Wang, eds.), (Cham), pp. 583–613, Springer International Publishing, 2020.
41. L. Ducas, "leaky-LWE-Estimator."
42. A. Leon-Garcia, *Probability, statistics, and random processes for electrical engineering*. Upper Saddle River, NJ: Pearson/Prentice Hall, 3. ed ed., 2008.
43. P. Q. Nguyen, "Hermite's Constant and Lattice Algorithms," in *The LLL Algorithm* (P. Q. Nguyen and B. Vallée, eds.), pp. 19–69, Berlin, Heidelberg: Springer Berlin Heidelberg, 2009. Series Title: Information Security and Cryptography.
44. V. Lyubashevsky and D. Micciancio, "On Bounded Distance Decoding, Unique Shortest Vectors, and the Minimum Distance Problem," in *Advances in Cryptology - CRYPTO 2009* (S. Halevi, ed.), vol. 5677, pp. 577–594, Berlin, Heidelberg: Springer Berlin Heidelberg, 2009. Series Title: Lecture Notes in Computer Science.
45. M. R. Albrecht, R. Player, and S. Scott, "On the concrete hardness of Learning with Errors," *Journal of Mathematical Cryptology*, vol. 9, Jan. 2015.

46. C. Peikert, "A Decade of Lattice Cryptography," *Found. Trends Theor. Comput. Sci.*, vol. 10, pp. 283–424, Mar. 2016. Place: Hanover, MA, USA Publisher: Now Publishers Inc.
47. K. Xagawa, "Cryptography with Lattices," p. 244, 2010.
48. T. Laarhoven, "Sieving for Shortest Vectors in Lattices Using Angular Locality-Sensitive Hashing," in *Advances in Cryptology – CRYPTO 2015* (R. Gennaro and M. Robshaw, eds.), Lecture Notes in Computer Science, (Berlin, Heidelberg), pp. 3–22, Springer, 2015.
49. A. Becker and T. Laarhoven, "Efficient (Ideal) Lattice Sieving Using Cross-Polytope LSH," in *Progress in Cryptology – AFRICACRYPT 2016* (D. Pointcheval, A. Nitaj, and T. Rachidi, eds.), (Cham), pp. 3–23, Springer International Publishing, 2016.
50. L. Babai, "On Lovász' lattice reduction and the nearest lattice point problem," *Combinatorica*, vol. 6, pp. 1–13, Mar. 1986.

A Security Estimation for NIST schemes

In this part, we estimate the security bit of LWE-based NIST schemes [38] under consideration of the influence of the two-step mode and the blocksize and jump strategy selection. Our new concrete hardness estimation of LWE [6] answers Question 7 in Section 5.3 of [2] and narrows the security estimation error interval. For more details about the construction of our new concrete hardness estimator of the Two-step mode of solving LWE, please refer to the citation [27]. Now our evaluation code is available at open source[6].

Table 8: Security Upper bound Estimation results of different estimators for NIST schemes with different blocksize and jump solving strategies.♮

	$\log_2$ G/$\log_2$(gates)			$\log_2$ B/$\log_2$(bit)			$\Delta \log_2$ G	
	Previous	Two-step		Previous	Two-step			
		S_0	S_{op}		S_0	S_{op}	S_0	S_{op}
Kyber512	146	142.6	141.4	93.97	99.1	98.1	3.4	4.6
Kyber768	208.9	205.5	204.4	138.73	144.0	143.2	3.4	4.5
Kyber1024	281.07	277.7	276.9	189.78	195.4	194.6	3.3	4.2
Dilithium-II	152.85	150.8	150.6	97.95	104.3	104.4	2.1	2.3
Dilithium-III	210.23	207.9	207.9	138.8	145.3	145.3	2.3	2.3
Dilithium-V	279.17	277.0	277.0	187.52	194.1	194.1	2.2	2.2

♮ The column "Previous" is the security estimation in the statement of Kyber and Dilithium. Strategy "S_0" uses a trial progressive BKZ+Pump in Two-step mode to estimate security. Strategy "S_{op}" uses a progressive BKZ+Pump with the optimized strategy selected by EnumBS in Two-step mode to estimate security. $\Delta \log_2$ G is the difference between "Previous" and "Two-step" under the RAM model in strategy S_0 and S_{op} in the logarithm of gate count with base 2. The gate count of all estimations in this Table uses the same improved list-decoding technique proposed by MATZOV [39].

Under the RAM model, i.e. assume that access into even exponentially large memory is free, the estimated security bit of LWE in NIST schemes [38] can be reduced by 2.2∼4.6 bit compared to the estimation generated by Leaky-LWE-Estimator[7] in [36] under gate-count model which adopts the improved list-decoding technique proposed in [39]. It fixed the estimate done in [40] of the list-decoding technique proposed in [25]. See Table 8 for details. Here G and B in Table 8 respectively represent the total number of logic circuits and the maximum memory needed for solving these LWE instances in NIST schemes [38] being solved, that both are calculated under same gate-count model.

Without considering the RAM model, the Two-step mode of using larger `Pump` dimension will indeed lead to an extra cost of accessing exponentially large

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

memory, which will somewhat offset the above-claimed decreasing of security hardness. But since the time cost (Number of gates G) is far larger than the memory cost B, the impact of memory growth can be ignored with such a significant decrease in time cost. Specifically, in Table 8 even though ΔB is negative (Two-step mode use bigger memory), $\Delta \log_2(\mathrm{G}+\mathrm{B})$ same as $\Delta \log_2 \mathrm{G}$ between 2.2∼4.6 bits is still positive. It means that the increase in memory will partially offset the decrease in the number of gates. However, in general, the time cost is still decreasing even considering the extra increase in memory.

Last but not least, although we only compare our security estimation with Leaky-LWE-Estimator [41] in this section, for other LWE estimators that consider only the BKZ-only mode, using our two-step mode along with the time-cost models in these estimators will also lead to better security estimation.

B Preliminaries

B.1 Notations and Basic Definitions

We write a matrix $\mathbf{B}$ as $\mathbf{B} = (\mathbf{b}_0, \cdots, \mathbf{b}_{d-1})$ where $\mathbf{b}_i$ is the $(i+1)$-th column vector of $\mathbf{B}$. The Euclidean norm of a vector $\mathbf{v}$ is denoted by $\|\mathbf{v}\|$. If $\mathbf{B} \in \mathbb{R}^{d\times d}$ has full rank d, the lattice $\mathcal{L}$ generated by the basis $\mathbf{B}$ is denoted by $\mathcal{L}(\mathbf{B}) = \{\mathbf{Bx} | \mathbf{x} \in \mathbb{Z}^d\}$. We denote $\mathbf{B}^* = (\mathbf{b}_0^*, \ldots, \mathbf{b}_{d-1}^*)$ as the Gram-Schmidt orthogonalization of $\mathbf{B}$, in which $\mathbf{b}_i^* = \mathbf{b}_i - \sum_{j=0}^{i-1} \mu_{i,j}\mathbf{b}_j^*$, $\mu_{i,j} = \langle \mathbf{b}_i, \mathbf{b}_j^* \rangle / \|\mathbf{b}_j^*\|^2$, for $i \in \{0, \cdots, d-1\}$. Let the orthogonal projection to the span of $(\mathbf{b}_0, \cdots, \mathbf{b}_{i-1})$ be π_i, i.e. $\forall \mathbf{v}$, $\pi_i(\mathbf{v}) = \mathbf{v} - \sum_{j=0}^{i-1} \omega_j \mathbf{b}_j^*$, where $\omega_j = \langle \mathbf{v}, \mathbf{b}_j^* \rangle / \|\mathbf{b}_j^*\|^2$. For $i, j \in \mathbb{Z}_d$ and $0 \le i < j \le d-1$, given an arbitrary d-dimensional vector $\mathbf{v} = (v_0, \cdots, v_{d-1})$, define $\mathbf{v}_{[i:j]}$ as $(v_i, \cdots, v_{j-1})$ with a size $j-i$. For a lattice basis $\mathbf{B}$, let $\mathbf{B}_{[i:j]} \leftarrow (\mathbf{b}_i, \cdots, \mathbf{b}_{j-1})$. Moreover, we denote $\mathbf{B}_{\pi[i:j]}$ by the local projected block $(\pi_i(\mathbf{b}_i), \cdots, \pi_i(\mathbf{b}_{j-1}))$, and call $\mathcal{L}_{\pi[i:j]}$ the lattice generated by $\mathbf{B}_{\pi[i:j]}$. We use $\mathbf{B}_{\pi[i]}$ and $\mathcal{L}_{\pi[i]}$ as shorthands for $\mathbf{B}_{\pi[i:d]}$ and $\mathcal{L}_{\pi[i:d]}$. The volume of a lattice $\mathcal{L}(\mathbf{B})$ is $\mathrm{Vol}(\mathcal{L}(\mathbf{B})) = \prod_{i=0}^{d-1} \|\mathbf{b}_i^*\|$, an invariant of the lattice. The first minimum of a lattice $\mathcal{L}(\mathbf{B})$ is the length of the shortest non-zero vector, denoted by $\lambda_1(\mathcal{L}(\mathbf{B}))$. We use the abbreviations $\mathrm{Vol}(\mathbf{B}) = \mathrm{Vol}(\mathcal{L}(\mathbf{B}))$ and $\lambda_1(\mathbf{B}) = \lambda_1(\mathcal{L}(\mathbf{B}))$.

Definition 1. *(The Gaussian Distribution [42]) Let $\sigma, u \in \mathbb{R}$ be the standard deviation and the mean value respectively, a continuous Gaussian Distribution denoted as $N(u, \sigma^2)$. Its probabilistic density function $\rho_{N(u,\sigma^2)} = e^{-\frac{(x-u)^2}{2\sigma^2}} / \sigma\sqrt{2\pi}$.*

Definition 2. *(Chi-Squared Distribution [42]) Given n random variables $X_i \sim N(0,1)$, the random variables $X_0^2 + \cdots + X_{n-1}^2$ follows a chi-squared distribution χ_n^2 over $\mathbb{R}^*$ of mean n and variance $2n$ with probabilistic density function $\rho_{\chi_n^2}(x) = x^{\frac{n}{2}-1} e^{-\frac{x}{2}} / 2^{\frac{n}{2}} \Gamma(n/2)$. Given n random variables $Y_i \sim N(0, \sigma^2)$, the random variables $Y_0^2 + \cdots + Y_{n-1}^2$ follows a scaled chi-squared distribution $\sigma^2 \cdot \chi_n^2$ over $\mathbb{R}^*$ of mean $n\sigma^2$ and variance $2n\sigma^2$.*

Heuristic 4 *(Gaussian Heuristic [23]) The expected first minimum of a lattice $\mathcal{L}$ (denoted as $\lambda_1(\mathcal{L}(\mathbf{B}))$) according to the Gaussian Heuristic denoted by $\mathrm{GH}(\mathcal{L})$ is*

given by $\lambda_1(\mathcal{L}(\mathbf{B})) \approx \mathrm{GH}(\mathcal{L}) = \left(\Gamma(\frac{d}{2}+1)\cdot \mathrm{Vol}(\mathcal{L})\right)^{\frac{1}{d}}/\sqrt{\pi} \approx \sqrt{d/(2\pi e)}\cdot \mathrm{Vol}(\mathcal{L})^{\frac{1}{d}}$ *Where* $\mathrm{V}_d(1)$ *is the volume of the d-dimensional unit sphere. We also write* $\mathrm{GH}(\mathbf{B}) = \mathrm{GH}(\mathcal{L}(\mathbf{B}))$ *and* $\mathrm{GH}(\mathsf{rr}_{[i:j]}) = \mathrm{GH}(\mathbf{B}_{\pi[i:j]})$.

Definition 3. *(Hermite-Korkine-Zolotarev and Block-Korkine-Zolotarev reductions [43]) The basis* $\mathbf{B}$ *of a lattice* $\mathcal{L}$ *is HKZ reduced if* $\mathbf{b}_i^* = \lambda_1(\mathcal{L}(\mathbf{B}_{\pi[i:d]}))$*, for all* $i < d$*.* $\mathcal{L}$ *is BKZ-*β *reduced if* $\mathbf{b}_i^* = \lambda_1(\mathcal{L}(\mathbf{B}_{\pi[i:\min\{i+\beta,d\}]}))$*, for all* $i < d$*.*

Definition 4. *(Root Hermite Factor [35]) For a basis* $\mathbf{B}$ *of d-dimensional lattice, the root Hermite factor is defined as* $\delta = \left(\|\mathbf{b}_0\|/\mathrm{Vol}(\mathbf{B})^{1/d}\right)^{1/d}$*, for estimating the quality of the output vector of BKZ. For larger blocksize, it follows the asymptotic formula* $\delta(\beta)^{2(\beta-1)} = \frac{\beta}{2\pi e}(\beta\pi)^{1/\beta}$.

Heuristic 5 *(Geometric Series Assumption [15]) Let* $\mathbf{B}$ *be a lattice basis after lattice reduction, then Geometric Series Assumption states that* $\|\mathbf{b}_i^*\| \approx \alpha\cdot\|\mathbf{b}_{i-1}^*\|$*,* $0 < \alpha < 1$*. Combine the GSA with root-Hermite factor (Definition 4) and* $Vol(\mathcal{L}(\mathbf{B})) = \prod_{i=0}^{d-1}\|\mathbf{b}_i^*\|$*, it infers that* $\alpha = \delta^{-\frac{2d}{d-1}} \approx \delta^{-2}$*. Let s be the slope value of the logarithm of GS norms* l_i *for* $\forall i \in \{1,\ldots,d\}$*,* $s \approx \ln\alpha$ *and* $\delta \approx e^{-\frac{s}{2}}$.

B.2 Lattice Hard Problems

Definition 5. *(unique Shortest Vector Problem(uSVP*$_\gamma$*) [44]) 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 6. *(*LWE_{m,n,q,D_σ} *Distribution [45–47]) Given some samples* $m \in \mathbb{Z}$*, a secret vector dimension* $n \in \mathbb{Z}$*, a modulo* $q \in \mathbb{Z}$ *, a probability distribution* D_σ*. Uniformly sample a matrix* $\mathbf{A} \in \mathbb{Z}_q^{m\times n}$ *and sample a secret vector* $\mathbf{s} \in \mathbb{Z}_q^n$ *from a specific distribution, randomly sample a relatively small noise vector* $\mathbf{e} \in \mathbb{Z}_q^m$ *from Gaussian distribution* D_σ *whose standard deviation is* σ*. The Learning with Errors (LWE) distribution* Ψ *is constructed by the pair* $(\mathbf{A}, \mathbf{b} = \mathbf{As}+\mathbf{e}) \in (\mathbb{Z}_q^{m\times n}, \mathbb{Z}_q^m)$ *sampled as above.*

Definition 7. *(Search* LWE_{m,n,q,D_σ} *problem [45–47]) Given a pair* $(\mathbf{A}, \mathbf{b})$ *sampled from* LWE_{m,n,q,D_σ} *distribution* Ψ *compute the pair* $(\mathbf{s}, \mathbf{e})$.

B.3 Sieving Algorithms

The first practical NV sieving algorithm uses a database of $N_0 \approx 2^{0.2075d+o(d)}$ vectors and runs in time $N_0^2 \approx 2^{0.415d+o(d)}$ by repeatedly checking all pairs $\mathbf{v}\pm\mathbf{w}$ [19]. To find the shortest vector, N_0 is the minimal number of vectors to ensure saturating the ball of radius $\mathrm{GH}(\mathcal{L})\sqrt{4/3}$ by short vector. In a line of works [22,25,48,49] the time complexity was gradually decreased to $2^{0.292d+o(d)}$ by nearest neighbor searching techniques.

B.4 Progressive Sieve

The progressive sieve [16] can save the cost of the classical sieve. It is realized by a right-to-left operation which first calls a sieve on a small dimension projected lattice, then uses Babai's nearest plane algorithm [50] to recover the vector to a higher dimension projected lattice. Repeat such a step until the short vectors onto a full dimensional lattice.

B.5 G6K and G6K-GPU-Tensor

G6K [15] is an abstract machine for running sieve and reduction algorithms, which is built on generalizing and extending the previous sieve algorithms. G6K-GPU-Tensor [24] as a state-of-art SVP solver improves the efficiency of G6K by GPU implementations and holds many records in TU Darmstadt SVP Challenges which is at least 400 times faster than the previous records.

C More experimental details about PnJBKZ simulator

In this part, we give more verification experiments of our PnJBKZ simulator on different LWE challenge lattice bases with different reduction parameters. Specifically, blocksize β increased from 55 to 100 and jump value increased from 1 to 12 and tours increased from 1 to 13. Meanwhile, for each reduction parameter, We did 20 times independent experiments to calculate the practical average value of the length of Gram-Schmidt vectors after the reduction of the PnJBKZ algorithm.

First of all, to remove the influence of q-ary vectors in LWE challenge initial lattice basis, we do pre-processing for all LWE challenge lattice basis by using small blocksize reduction which can be done within a few minutes wall time. For example $(n = 70, \alpha = 0.005)$ and $(n = 75, \alpha = 0.005)$, after pre-processing we can get LWE challenge lattice basis $(n = 70, \alpha = 0.005)$ with a slope equal to $-0.04921/2$ and LWE challenge lattice basis $(n = 75, \alpha = 0.005)$ with a slope equals to $-0.04339/2$. Then corresponding $\mathrm{d4f}_{slope}(s)$ between 11.7 to 13.7. As we need the maximum jump value $J \leq \mathrm{d4f}_{slope}(s)$ to ensure the accuracy of the PnJBKZ simulator (see section 4.1 for details), we set the maximum jump value $J \leq 12$ in our test experiments.

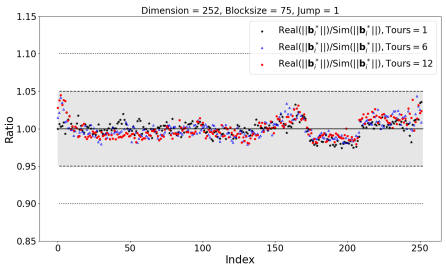
Then we give the results of verification experiments on four different lattice bases with $\beta \in [50, 70]$ and jump within $J \in [1, 12]$: $(n = 70, \alpha = 0.005)$, $(n = 75, \alpha = 0.005)$, $(n = 60, \alpha = 0.010)$ and $(n = 50, \alpha = 0.015)$. See Figure 12 $\sim$ 20 respectively. Verification experiments results indicate that our PnJBKZ simulator performs well in predicting the behavior of PnJBKZ which blocksize within $\beta \in [75, 100]$ and jump within $J \in [1, 12] \leq d4f_{slope(s)}$ on LWE challenge lattice basis on 4 different LWE challenge lattice bases.

Figures 10 $\sim$ 20 show that on different lattice basis with different reduction parameters, as long as the jump $\leq \mathrm{d4f}_{slope}(s)$, even the tours increase to 13, overall, the simulation values are closed to the actual values and most of ratios

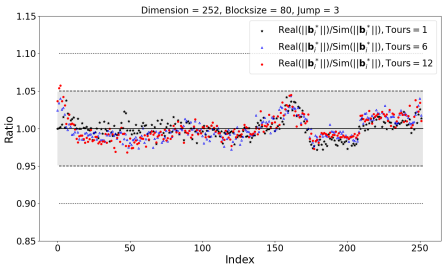
$\frac{l_i''}{\mathrm{Sim}(l_i'')}$ are within $[0.95, 1.05]$. Meanwhile, the reduction strategy shown in Section 5 will not run the same reduction parameter (β, J) for more than 6 tours, while our simulator is still accurate even if the tours increase to 13. The above results indicate that when $J \leq \mathrm{d4f}_{slope}(s)$ we can ensure Heuristic 1 is held and the PnJBKZ simulator calculates the estimation value of the actual value $\|\mathbf{b}_i''^*\|$ by Eq. (1) can already reflect how the average of the norms of Gram-Schmidt vectors change during each tour's reduction of PnJBKZ(β, J). Therefore our PnJBKZ simulator fits well with the actual PnJBKZ reduction result. In addition, Table 6 and Appendix H show that the practical slope of lattice Gram-Schmidt basis after each tour of the reduction of PnJBKZ with different blocksizes and different jump values is very close to that of our simulation, which also verified the accuracy of our PnJBKZ simulator. For selecting the optimized reduction strategy, our PnJBKZ simulator is accurate enough.

C.1 LWE challenge lattice basis ($n = 75, \alpha = 0.005$).

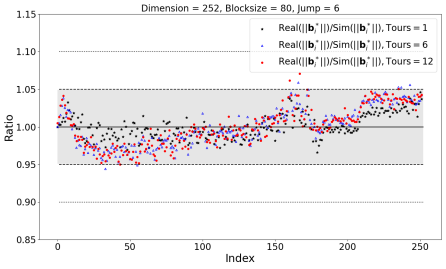
Figure 10 $\sim$ Figure 11.



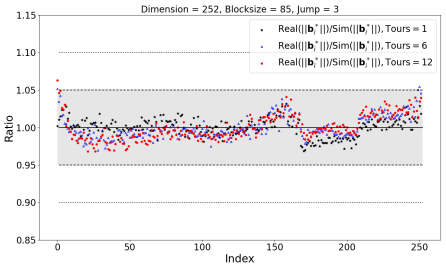
(a) β=75, Jump=1



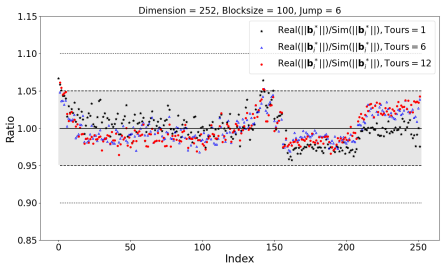
(b) β=80, Jump=3



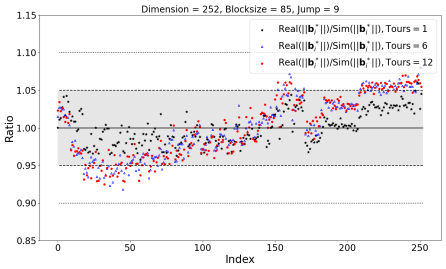
(c) β=80, Jump=6



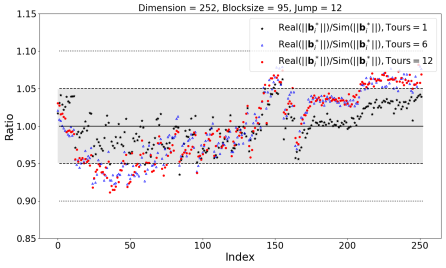
(d) β=85, Jump=3



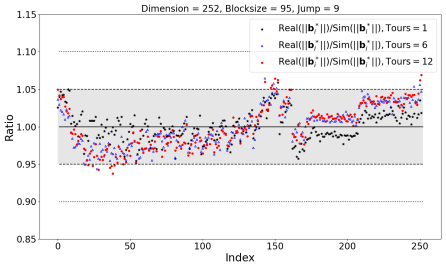
(e) β=100, Jump=6



(f) β=85, Jump=9

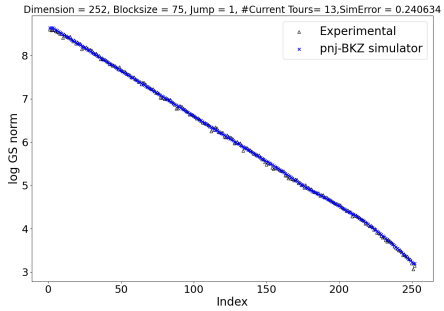


(g) β=95, Jump=12

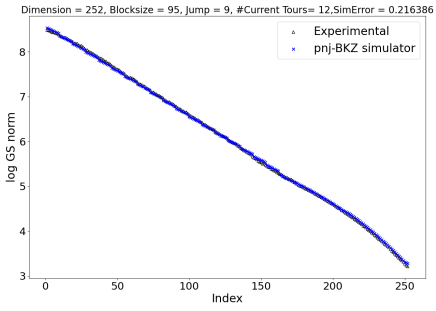


(h) β=95, Jump=9

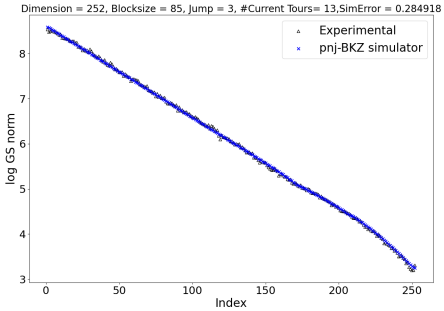
Fig. 10: Ratio $l_i''/\mathrm{Sim}(l_i'')$. Run 12 tours of PnJBKZ(β, J) reduction on a 252-dimension LWE lattice basis ($n = 75, \alpha = 0.005$), and record the ratio values. We test 20 times for each reduction parameters.



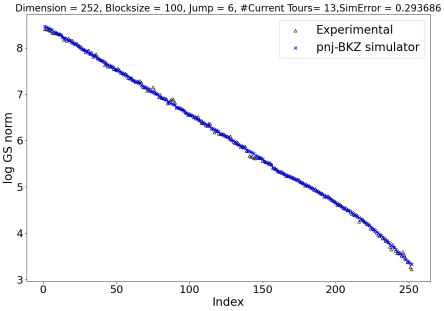
(a) $\beta = 75, jump = 1, \sharp\text{tours} = 13$



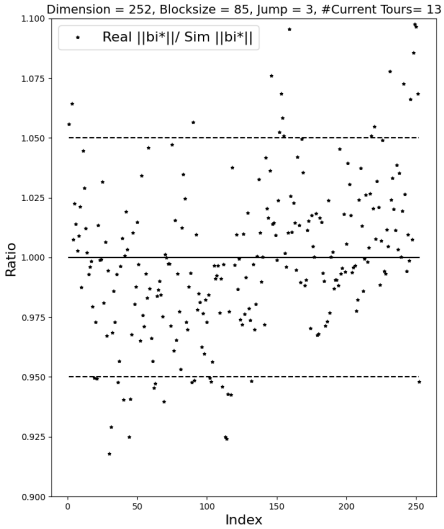
(b) $\beta = 95, jump = 9, \sharp\text{tours} = 12$



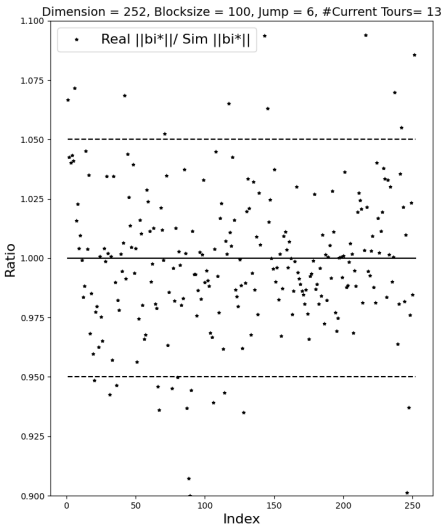
(c) $\beta = 85, jump = 3, \sharp\text{tours} = 13$



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



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

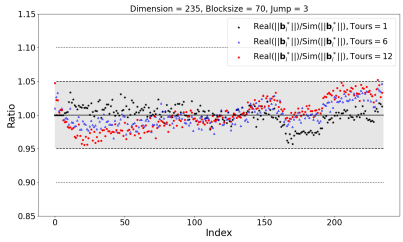


(f) $\beta = 100, jump = 6, \sharp\text{tours} = 13$

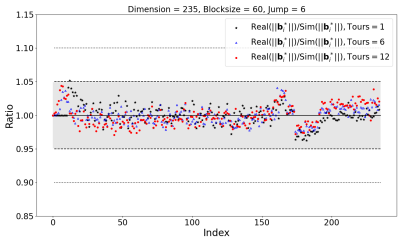
Fig. 11: Overall Prediction effect of PnJBKZ simulator. Ratio $l_i''/\text{Sim}(l_i'')$. We perform the experiments by reducing the lattice basis of LWE Challenge ($n = 75, \alpha = 0.005$). We test also 20 times for each reduction parameters.

C.2 LWE challenge lattice basis ($n = 70, \alpha = 0.005$).

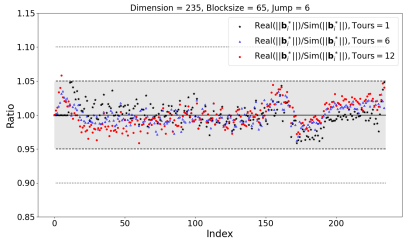
Figure 12 $\sim$ Figure 14.



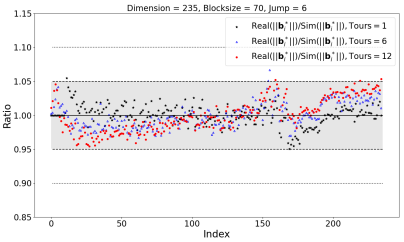
(a) $\beta = 70, jump = 3$



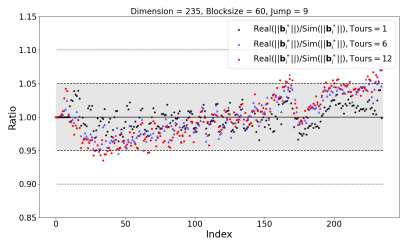
(b) $\beta = 60, jump = 6$



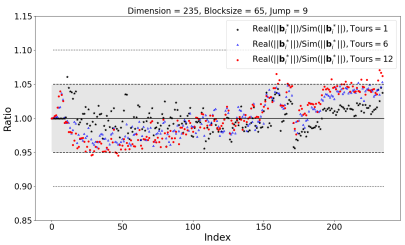
(c) $\beta = 65, jump = 6$



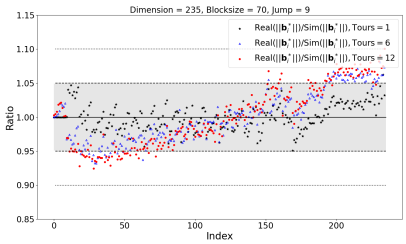
(d) $\beta = 70, jump = 6$



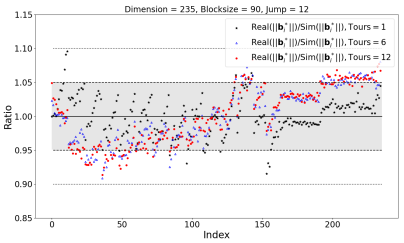
(e) $\beta = 60, jump = 9$



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

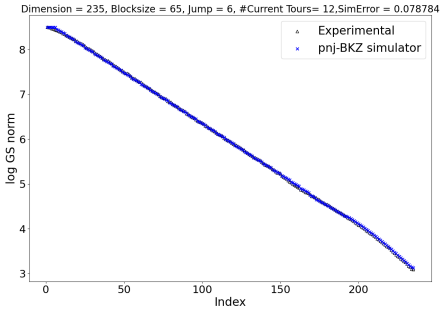


(g) $\beta = 70, jump = 9$

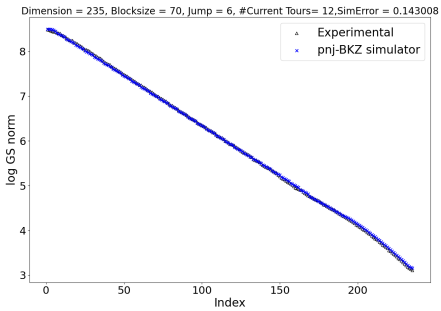


(h) $\beta = 90, jump = 12$

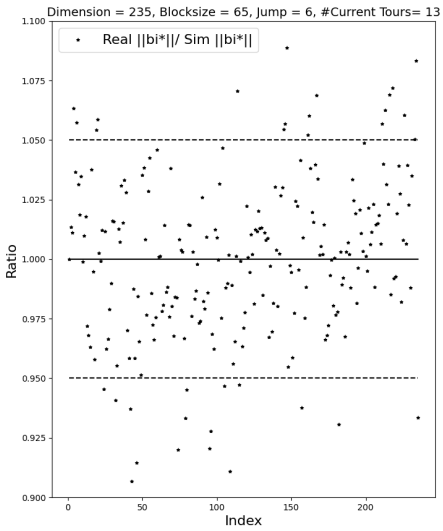
Fig. 12: Ratio $l_i''/\mathrm{Sim}(l_i'')$. Run 12 tours of PnJBKZ$(\beta, J)$ reduction on a 235-dimension LWE lattice basis ($n = 70, \alpha = 0.005$), and record the ratio values. We test 20 times for each reduction parameter.



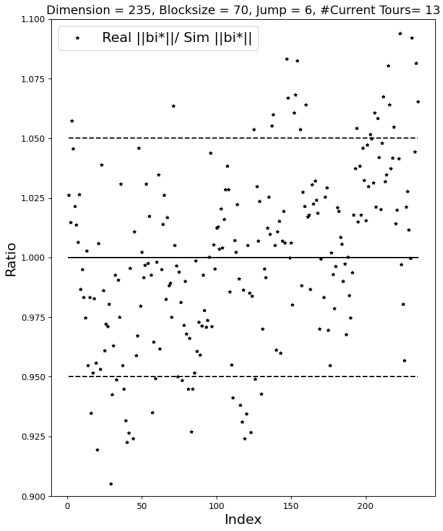
(a) $\beta = 65, jump = 6, \sharp\text{tours} = 12$



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

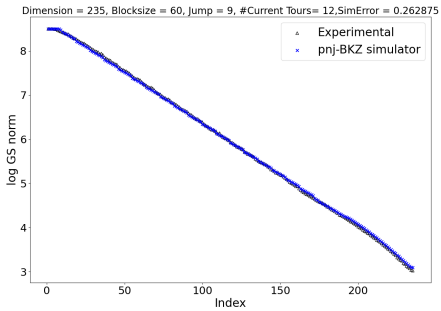


(c) $\beta = 65, jump = 6$

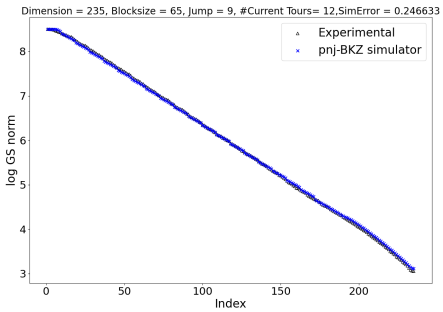


(d) $\beta = 70, jump = 6$

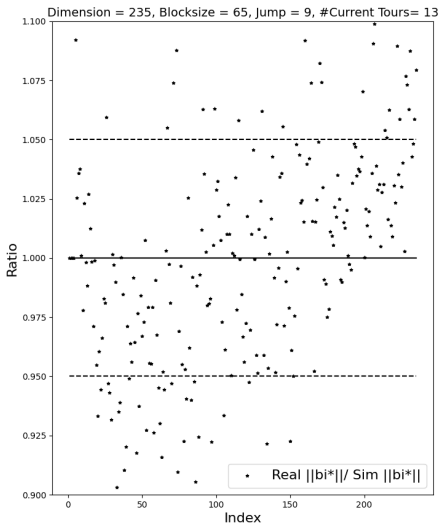
Fig. 13: Ratio $l_i''/\text{Sim}(l_i'')$. Run 13 tours of PnJBKZ(β, J) reduction on a 235-dimension LWE lattice basis ($n = 70, \alpha = 0.005$), different β with $J = 6$, and record the ratio values. We test 20 times for each reduction parameter.



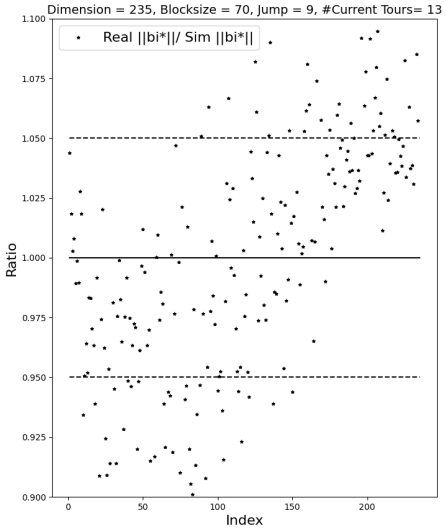
(a) $\beta = 60, jump = 9, \sharp\text{tours} = 12$



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



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

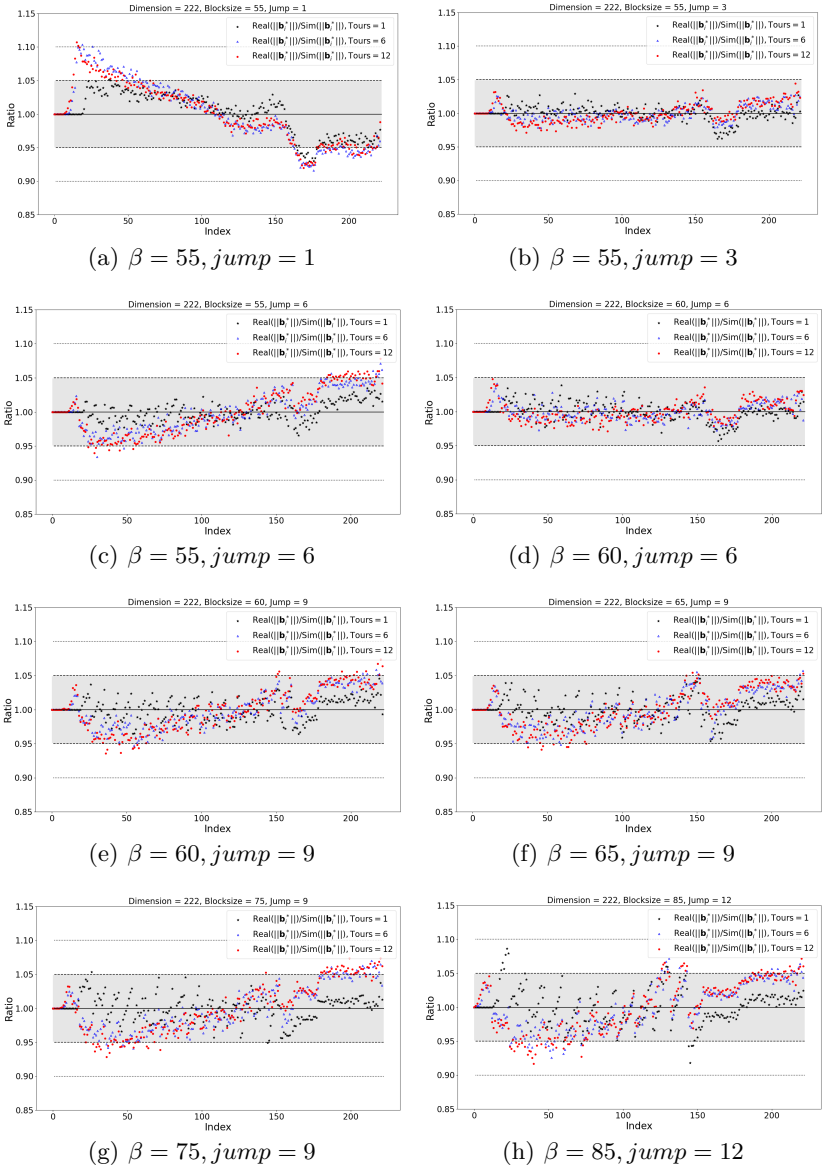


(d) $\beta = 70, jump = 9$

Fig. 14: ratio $l_i''/\text{Sim}(l_i'')$. Run 13 tours of PnJBKZ(β, J) reduction on a 235-dimension LWE lattice basis $(n = 70, \alpha = 0.005)$, different β with $J = 9$, and record the ratio values. We test 20 times for each reduction parameter.

C.3 LWE challenge lattice basis ($n = 60, \alpha = 0.010$).

Figure 15 $\sim$ Figure 17.



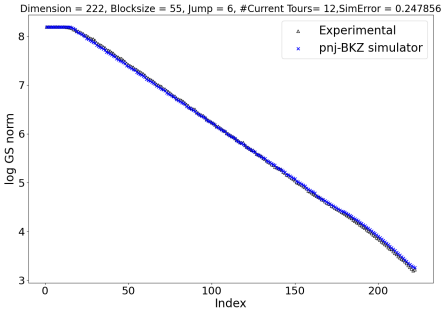
(a) $\beta = 55, jump = 1$ (b) $\beta = 55, jump = 3$

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

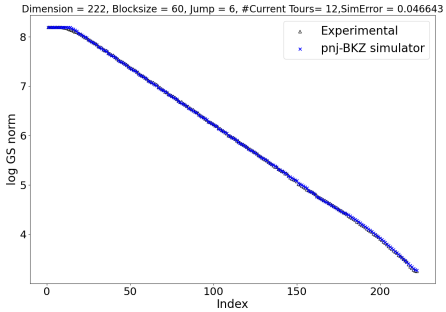
(e) $\beta = 60, jump = 9$ (f) $\beta = 65, jump = 9$

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

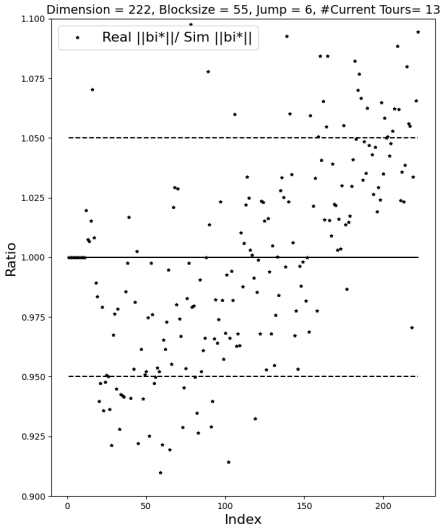
Fig. 15: Ratio $l_i''/\mathrm{Sim}(l_i'')$. Run 12 tours of PnJBKZ$(\beta, J)$ reduction on a 222-dimension LWE lattice basis $(n = 60, \alpha = 0.010)$, and record the ratio values. We test 20 times for each reduction parameter.



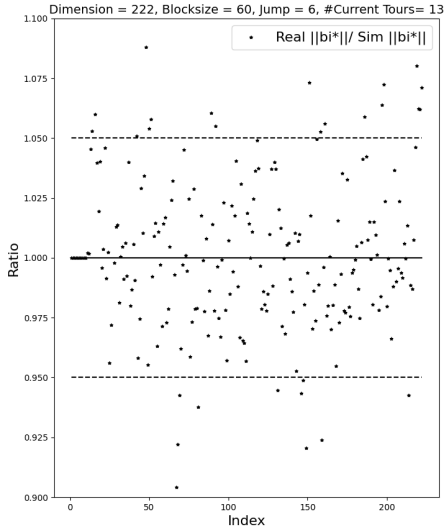
(a) $\beta = 55, jump = 6, \sharp\text{tours} = 12$



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

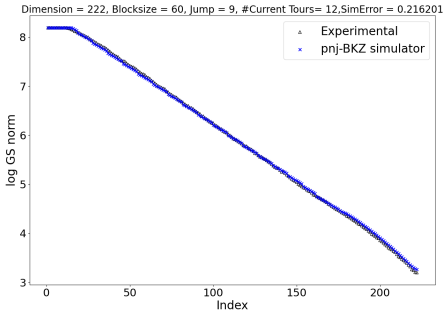


(c) $\beta = 55, jump = 6$

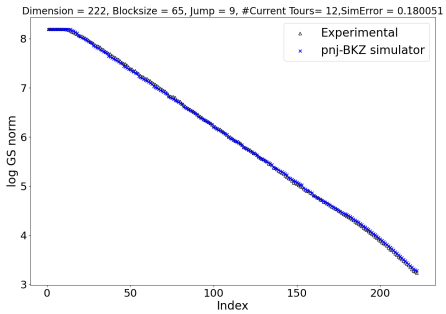


(d) $\beta = 60, jump = 6$

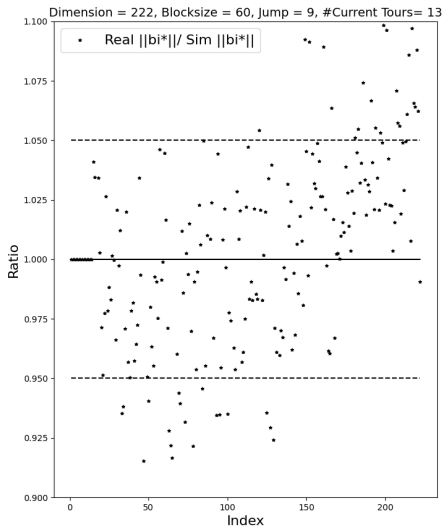
Fig. 16: Ratio $l_i''/\text{Sim}(l_i'')$. Run 13 tours of PnJBKZ(β, J) reduction on a 222-dimension LWE lattice basis $(n = 60, \alpha = 0.010)$, different β with $J = 6$, and record the ratio values. We test 20 times for each reduction parameter.



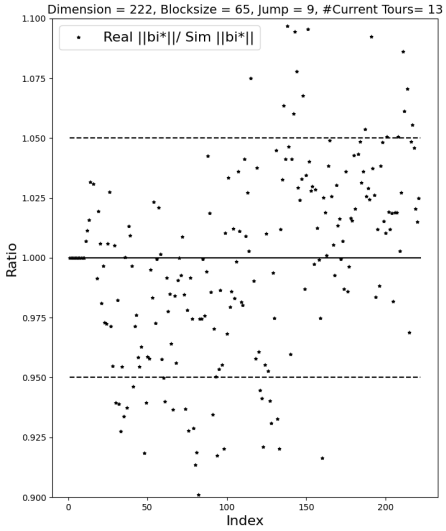
(a) $\beta = 60, jump = 9, \sharp\text{tours} = 12$



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



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

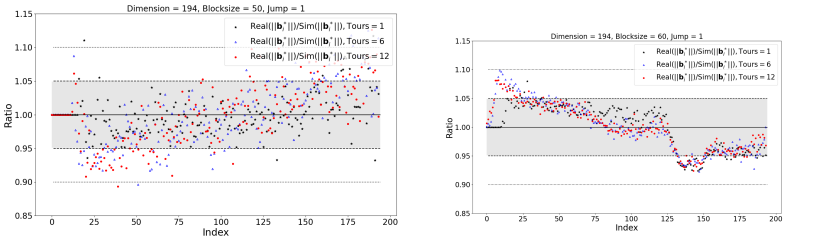


(d) $\beta = 65, jump = 9$

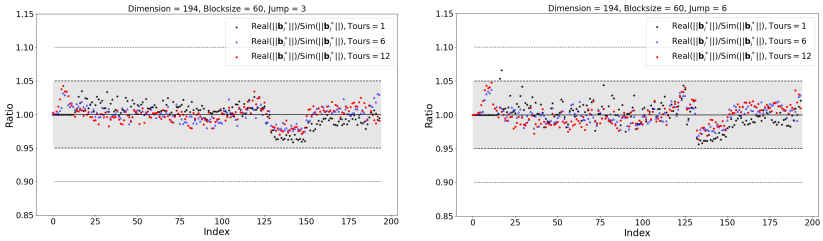
Fig. 17: Ratio $l_i''/\text{Sim}(l_i'')$. Run 13 tours of PnJBKZ(β, J) reduction on a 222-dimension LWE lattice basis ($n = 60, \alpha = 0.010$), different β with $J = 9$, and record the ratio values. We test 20 times for each reduction parameter.

C.4 LWE challenge lattice basis ($n = 50, \alpha = 0.015$).

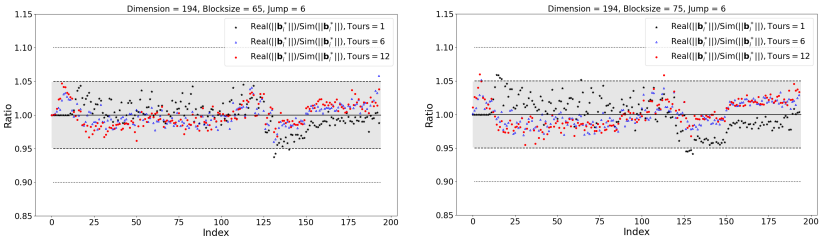
Figure 18 $\sim$ Figure 20.



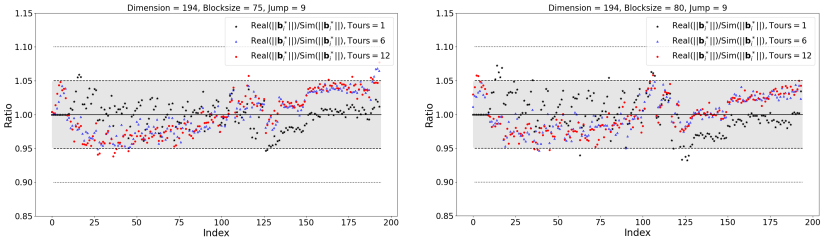
(a) $\beta = 50, jump = 1$ (b) $\beta = 60, jump = 1$



(c) $\beta = 60, jump = 3$ (d) $\beta = 60, jump = 6$

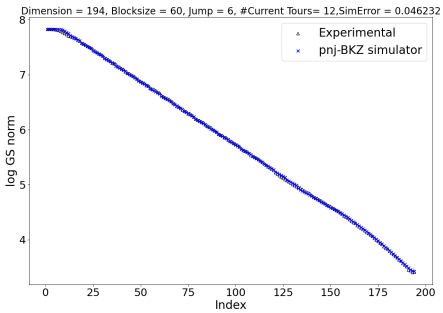


(e) $\beta = 65, jump = 6$ (f) $\beta = 75, jump = 6$

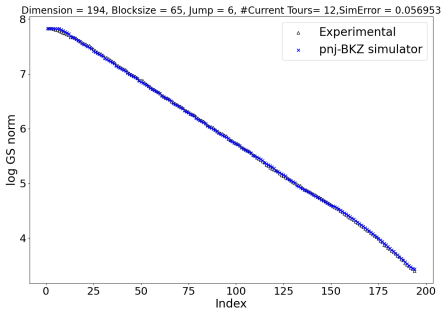


(g) $\beta = 75, jump = 9$ (h) $\beta = 80, jump = 9$

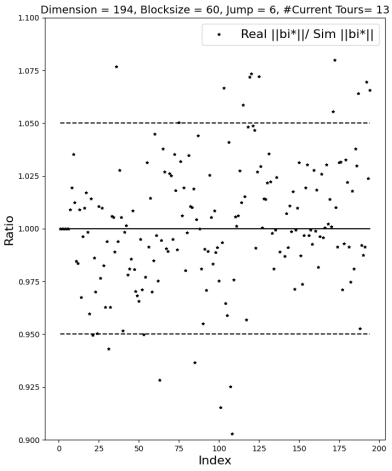
Fig. 18: Ratio $l_i''/\mathrm{Sim}(l_i'')$. Run 12 tours of PnJBKZ$(\beta, J)$ reduction on a 194-dimension LWE lattice basis $(n = 50, \alpha = 0.015)$, and record the ratio values. We test 20 times for each reduction parameter.



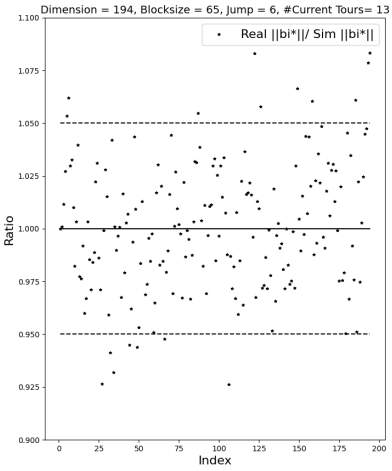
(a) $\beta = 60, jump = 6, \sharp\text{tours} = 12$



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

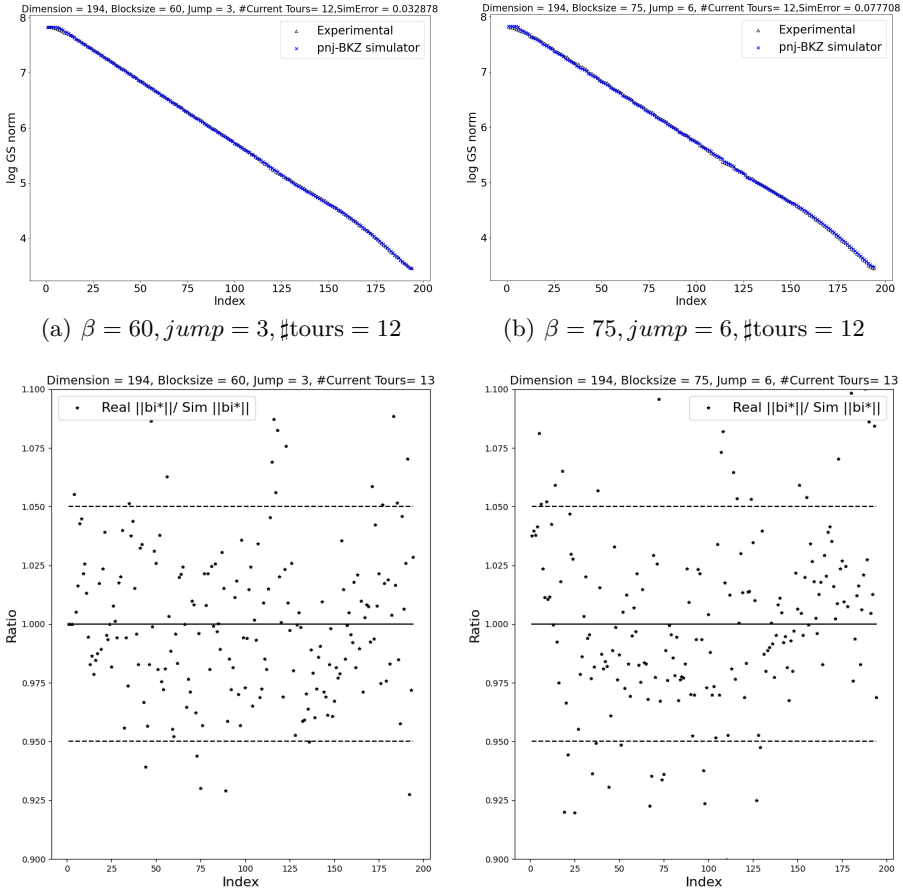


(c) $\beta = 60, jump = 6$



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

Fig. 19: Ratio $l_i''/\text{Sim}(l_i'')$. Run 13 tours of PnJBKZ(β, J) reduction on a 194-dimension LWE lattice basis $(n = 50, \alpha = 0.015)$, different β with $J = 6$, and record the ratio values. We test 20 times for each reduction parameter.



(a) $\beta = 60, jump = 3, \sharp\text{tours} = 12$ (b) $\beta = 75, jump = 6, \sharp\text{tours} = 12$

(c) $\beta = 60, jump = 3$ (d) $\beta = 75, jump = 6$

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

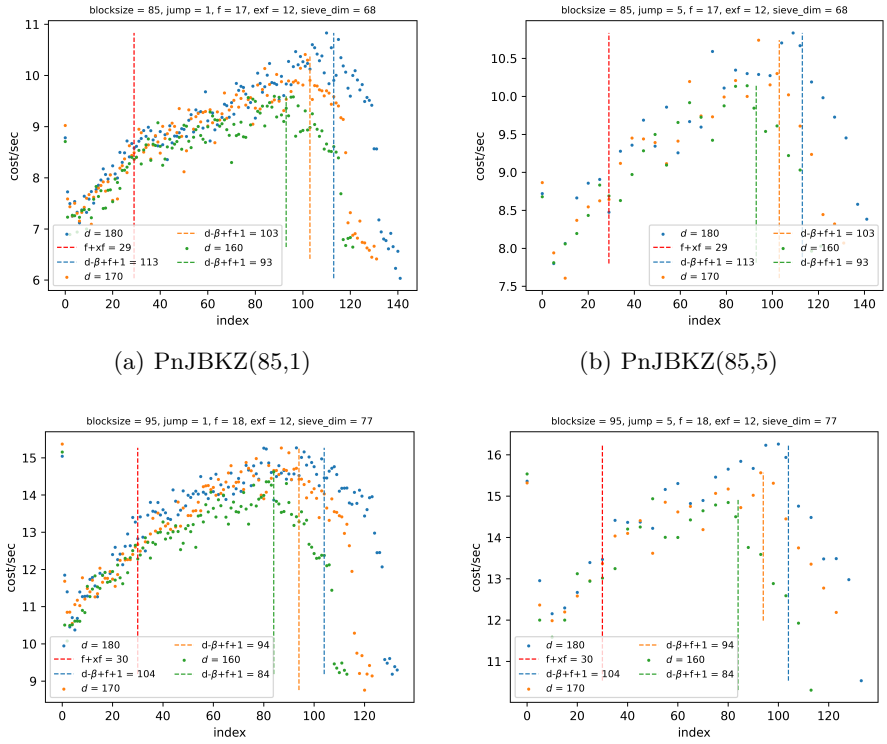
D Practical time cost model of Pump and PnJBKZ

To find the progressive blocksize and jump size selection strategy with minimal expected time cost for solving TU Darmstadt LWE challenges, it is necessary to construct PnJBKZ and `Pump` time cost models. However, the asymptotic complexity of the sieving does not match the actual cost well in the low-dimensional case[5] (dimension $\leq$ 128). The multi-threading technology used in `Pump` will

[5] While dimension exceeds 128, the time cost for `Pump` and PnJBKZ fits the theoretical value well, we can directly use the time cost model of triple_gpu sieve declared in [24].

balance part of the time cost increases when the dimension of sieving increases. Therefore, we construct a practical time cost model by using the experimental method to test the running time of the `Pump` in Appendix D.1 on a different lattice basis for finding the optimized reduction parameters of solving TU Darmstadt LWE challenges in shorter time cost.

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



(a) PnJBKZ(85,1)

(b) PnJBKZ(85,5)

(c) PnJBKZ(95,1)

(d) PnJBKZ(95,5)

Fig. 21: Cost for each `Pump` under different index in a PnJBKZ tour by testing SVP Challenge with different dimension d using Machine C with threads = 32 and GPUs = 2.

Besides, when we construct the actual time cost model by testing the time cost of PnJBKZ on the specific machine, we find that each `Pump` in PnJBKZ takes a different time cost as Fig. 21 shown. Especially, the time cost of the first Pump is higher than subsequent `Pump`s and it increases under the incremental index from 2^{nd} to $(d-\beta+f+1)^{\text{th}}$ and decreases after $d-\beta+f+1$ indices. It infers that for a fixed blocksize β, the average `Pump` cost in PnJBKZ will increase with the growth of dimension d. It means that the simplified model of treating each SVP oracle inside BKZ as having the same time cost no longer applies in the context of PnJBKZ. Therefore, in Appendix D.2, we propose a new time cost model for PnJBKZ to more accurately reflect its time cost performance in practical applications.

D.1 Practical Cost Model of Pump

We can regard T_{Pump} as a computational cost model of the d_{svp}-dimensional progressive sieve. T_{Pump} in [15] is considered as

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

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

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

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

Take β as the independent variable, $\log_2(T_{\text{Pump}})$ can be obtained from the experimental test as the dependent variable, and we use the least squares fitting to find c and c_1. We use R^2 to denote the coefficient of determination (R squared) value above the linear regression model. The coefficient of determination (R^2 or R squared) is a statistical measure in a regression model that determines the proportion of variance in the dependent variable that can be explained by the independent variable. Generally, the range of R^2 is $[0,1]$ and when R^2 closer is to 1, the better the model fits the data.

From Figure 22, we can see that R^2 is close to 1. It means that the fitting effect is good. Figure 22 also shows that the logarithm of the computational cost of `Pump` is linearly correlated to d_{svp} under both float type "dd" and "qd".

Since the "qd" float type is more precise than "dd", it is slower than "dd". So we suggest setting "dd" float type.

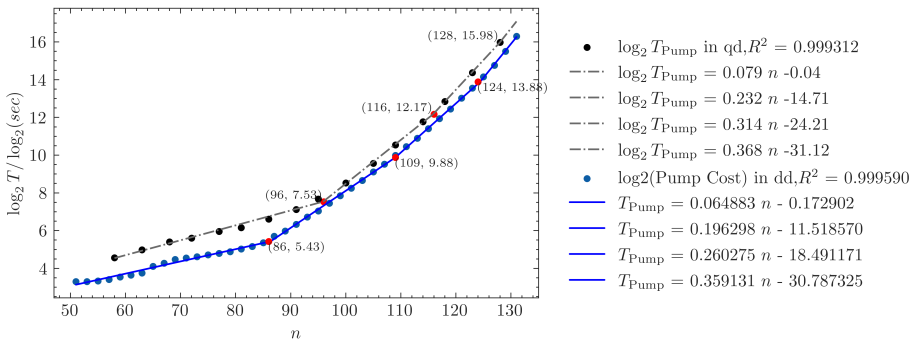


Fig. 22: `Pump` Cost Figure while $d = 180$, Sieve used in `Pump` is gpu_sieve, and it's running on Machine C with 2 GPUs and 32 threads: Relation between $\log_2(T_{\text{Pump}})$ and sieve dimension n.

Consider Cost of SimHash Generation. Moreover, given sieve dimension β, the G6K (or its GPU version) implementation requires $O(\beta)$ memory and $O(\beta)$ computational cost to generate the SimHash value, which is used to find the nearest neighbor of each vector. Thus, an $O(2^{c\beta})$-time and $O(2^{c_2\beta})$-space algorithm actually requires $O(2^{c\beta} + \beta * 2^{c_2\beta})$. Set $c = 0.367$ and $c_2 = 0.2075$ according to Fig. 7 in [24] and construct the practical `Pump` model as

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

then we can obtain the practical `Pump` cost model (as Fig. 23 shown) through the curve fitting method.

D.2 Practical Cost Model of PnJBKZ

PnJBKZ consists of a series of Pumps. If we regard PnJBKZ as a combination of Pumps with equal cost, the computational cost of PnJBKZ can be calculated by the sum cost of $\frac{d+2f-\beta}{J}$ progressive sieves on the $(\beta - f)$-dimension projected sublattice with jump J. However, as Fig. 21 shows, each `Pump` in PnJBKZ has a different cost. Especially, the `Pump` cost increases from the 2^{nd} to the $(d-\beta+f+1)^{\text{th}}$ index and decreases afterward. Here, in Figure 21, we can observe that the growth rate in the range of $[0, f+f_{\text{extra}}]$ differs from that of $[f+f_{\text{extra}}, d-\beta+f+1]$, where f_{extra} represents the extra dimension-for-free value set in G6K to enhance the efficiency of PnJBKZ and it is 12 in the default setting. So we depart the PnJBKZ cost into 4 parts: first index of `Pump`, preceding indices in range of

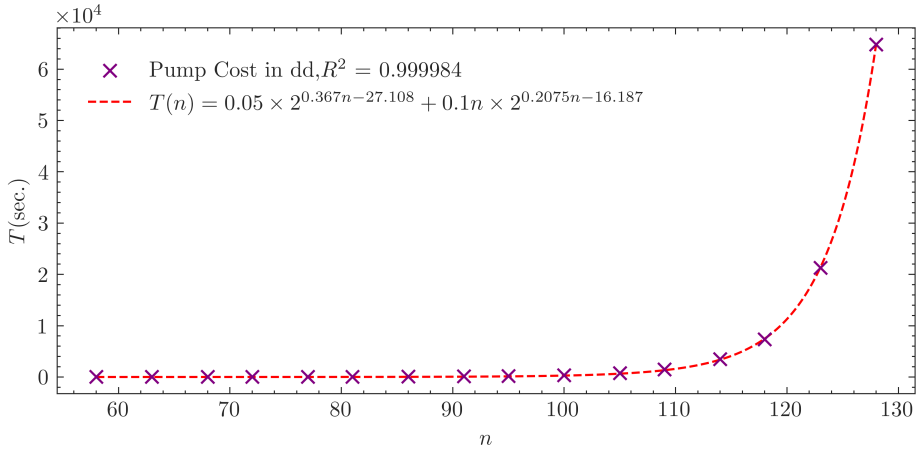


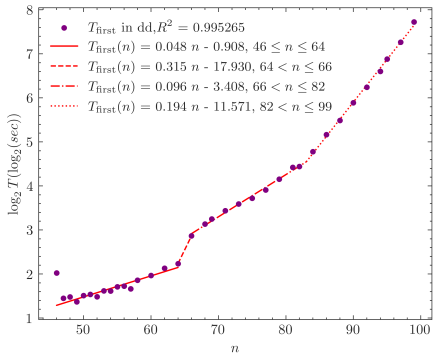
Fig. 23: `Pump` cost model considers the cost to generate the hash value: The vertical axis represents T_{Pump} and the horizontal axis represents the sieve dimension n.

$[0, f + f_{\text{extra}})$, middle indices in range of $[f + f_{\text{extra}}, d - \beta + f + 1)$ and later indices in range of $[d - \beta + f + 1, d)$. Let the cost of each range be T_{first}, T_{pre}, T_{mid} and T_{former}. Let $T_{f+f_{\text{extra}}}$ and $T_{d-\beta+f+1}$ be the `Pump` cost at the index $f + f_{\text{extra}}$ and $T_{d-\beta+f+1}$. We have tested T_{first}, T_{pre}, T_{later} and the coefficients A and B in $T_{\text{mid}}(d, \beta, J, f, f_{\text{extra}}) = \frac{T_{f+f_{\text{extra}}} + T_{d-\beta+f+1}}{2} \cdot (d - \beta - f_{\text{extra}} + 1) = \frac{(A \cdot (f+f_{\text{extra}})+B)+(A \cdot (d-\beta+f)+B)}{2} \cdot (d - \beta - f_{\text{extra}} + 1)$ in dimension $d = 180$, jump $J = 1$ and "dd" float type, then we obtain the simulated cost model as Fig. 24. Then, we can get that

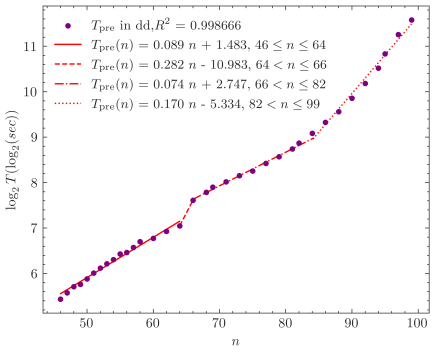
$$\begin{aligned} T_{\text{PnJBKZ}}(d, \beta, J, f, f_{\text{extra}}) = T_{\text{first}} &+ T_{\text{pre}} \cdot \frac{\lceil \frac{f+f_{\text{extra}}}{J} \rceil - 1}{f + f_{\text{extra}} - 1} \\ &+ T_{\text{mid}}(d, \beta, J, f, f_{\text{extra}}) \cdot \frac{\lceil \frac{f+f_{\text{extra}}}{J}) \rceil}{f + f_{\text{extra}}} \\ &+ T_{\text{later}} \cdot \frac{\lceil \frac{d-\beta-f_{\text{extra}}}{J} \rceil + 1}{d - \beta - f_{\text{extra}} + 1}, \end{aligned} \tag{3}$$

where f is the dimension for free value of β.

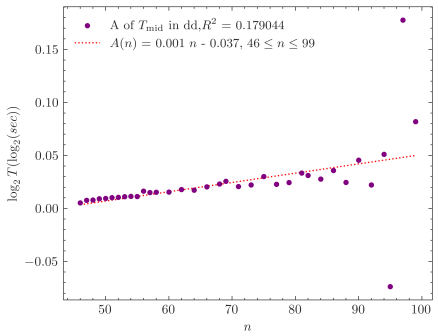
We've also used the Eq. 3 to simulate the PnJBKZ cost of other dimensions (such as $d = 150, 160, 170$) with blocksize from 51 to 119 and jump $J \geq 1$, and find it fits well in simulation as Figure 25.



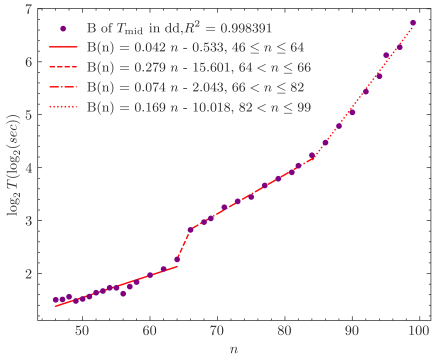
(a) T_{first}



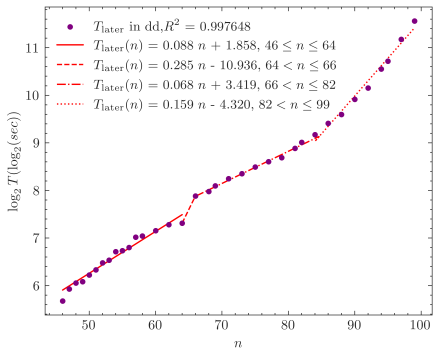
(b) T_{pre}



(c) A of T_{mid}

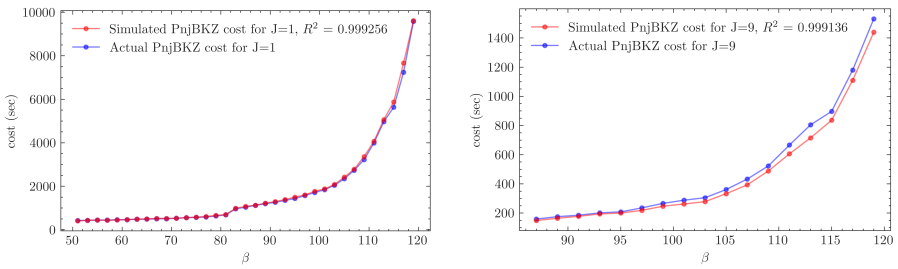


(d) B of T_{mid}



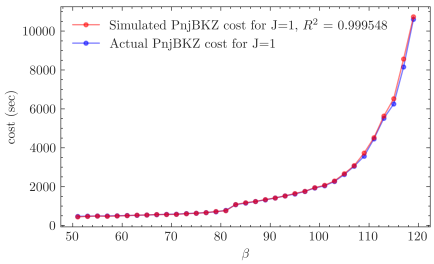
(e) T_{later}

Fig. 24: 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.

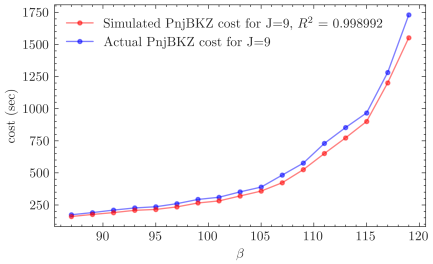


(a) $(d, J) = (160, 1)$

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



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



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

Fig. 25: Simulate each PnJBKZ Cost using Eq. (24 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.

E Blocksize and Jump Strategy Selection based on ProBKZ

The *blocksize and jump strategy selection algorithm based on ProBKZ* (BSSA, Fig. 26) applies the *Shortest Path Algorithm* to strategy selection.

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

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

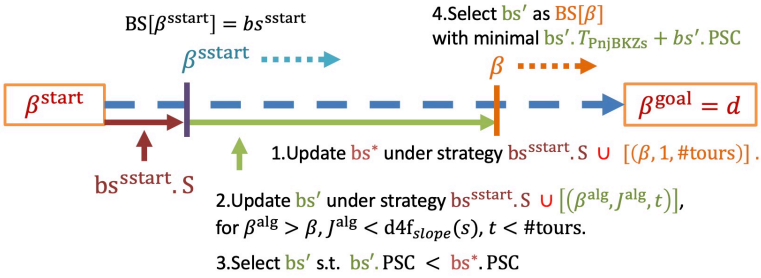


Fig. 26: BSSA Process.

BKZ-β^{goal} reduced, which is the combination of $(\beta^{\text{alg}}, J^{\text{alg}}, t, T_{\text{PnJBKZ}})$ stored on each edge in the shortest path from node β^{start} to β^{goal} with respect to the sum of simulated BKZ cost $T_{\text{PnJBKZs}} = \sum_{\beta^{\text{alg}}, J^{\text{alg}}, t} T_{\text{PnJBKZ}}(\beta^{\text{alg}}, J^{\text{alg}}, t)$, while the shortest path can be found using Dijkstra algorithm. PSC is one of the output from the `Pump` dimension estimation method (Alg. 3), which means the estimated time cost for uSVP$_\gamma$ to be solved by processing `Pump` on the BKZ-β^{goal} reduced basis.

By setting different final β^{goal}, we can get different reduction strategy BS that improves the quality of lattice basis from β^{start} to β^{goal} and different sieving dimension of the last `Pump` corresponding to the different quality of the lattice that is fully β^{goal} reduced. Then we set multiple different final β^{goal} to choose the Two-step solving strategy whose total time cost is minimum. Here, the total time cost includes the time cost of improving the quality of lattice by a series of PnJBKZ$(\beta, J) \in$ S and the time cost of final `Pump`. See Alg. 5 for more details about BSSA.

F Choosing the number of LWE Samples

BKZ-only mode is the mainstream method for estimating the security of an LWE-based cryptosystem at the current. It uses Kannan's Embedding technique to reduce the LWE problem to the uSVP$_\gamma$ problem and uses the GSA assumption to simulate the change after a BKZ-β reduction. Its evaluation method was firstly proposed by Erdem Alkim *et al.*. in [14] and has been proved the correctness in [32], which has both given a lower bound of LWE samples and a blocksize β. We renamed it "2016 Estimation from GSA for LWE" (referred to as 2016 Estimate).

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

instance, then

$$\min_{\beta\in\mathbb{N}}\left\{T_{\mathrm{BKZ}}(\beta):\sigma\sqrt{\beta}\leq\delta\left(\beta\right)^{2\beta-d-1}\cdot q^{\frac{d-n-1}{d}}\right\}. \tag{4}$$

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

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

Algorithm 5: BSSA

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

$$\min_{\beta,d_{\mathrm{svp}}\in\mathbb{N}}\left\{2\cdot T_{\mathrm{BKZ}}(\beta)+\mathrm{PSC}(d_{\mathrm{svp}}):\|\pi_{d-d_{\mathrm{svp}}}(\mathbf{v})\|\leq\mathrm{GH}(\mathcal{L}_{\pi[d-d_{\mathrm{svp}}]})\right\}, \tag{5}$$

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

However, we have explained in Sec. 4.2 that the 2016 Estimate still has a probability of failing to find the target vector through its estimation. Thus, our

strategy for solving the LWE problem considers simulating a two-stage strategy using our PnjBKZ simulator and new `Pump` sieve dimension and PSC estimation scheme (as described in Alg. 3) In the first stage, it will call the PnJBKZ simulator to simulate the basis after a series of PnJBKZ. In the second stage, it tries to find the approximate shortest vector by Pump. Based on the estimation scheme in the default G6K described above, we modify the time cost of two PnJBKZs and a progressive sieve to the time cost of serial PnJBKZs following the blocksize strategy and a progressive sieve. Besides, we use the new `Pump` estimation scheme to simulate the norm of the target vector. Let $P(d_{\text{svp}}) = \Pr\left[y \leftarrow \sigma^2\chi^2_{d_{\text{svp}}} \middle| y \leq \left(\text{GH}\left(\mathcal{L}_{\pi[d-d_{\text{svp}}:d]}\right)\right)^2\right]$. Thus, the formula becomes

$$\min_{\beta, d_{\text{svp}} \in \mathbb{N}} \left\{T_{\text{PnJBKZs}}\left(\mathbf{B}\right) + \text{PSC}\left(d_{\text{svp}}\right) : P(d_{\text{svp}}) \geq P_{\text{success}}\right\}, \tag{6}$$

where δ is the basis quality after PnJBKZs. $T_{\text{PnJBKZs}}\left(\mathbf{B}\right)$ will respectively call BSSA or EnumBS to calculate the corresponding computational cost. To minimize the number of attempts, we narrow the range of m to $[m_0 - \tau, m_0 + \tau]$, where m_0 is the number of samples chosen in the estimation of default G6K and set a maximum search field range $\tau \in \mathbb{Z}^*$. We use dichotomization to find an m with minimum β and d_{svp} satisfying the inequality (6). Furthermore, the concrete process is as the Algorithm 6.

input: n, q, α, m_{all}, β_{bound}, $d^{(\text{svp})}_{\text{bound}}$, τ, $\mathbf{A}^{m_{all}\times n}$, $\mathbf{b}^{m_{all}\times 1}$;
output: $S_{\min}$, $T_{\min}$, m;
$\sigma, T_{\min}, \mathsf{mRange} \leftarrow \alpha q, +\infty, \{\}$;
$m_0 \leftarrow$ LWE samples estimation in G6K as formula (5);
$m_{\min} \leftarrow \min\{m$ **satisfies** equation (6)$\}$; $S_{\min}, T_{\min} \leftarrow$ None, None;
while $\tau > 0$ **do**
  Construct $\mathbf{B}$ by $\left(\mathbf{A}^{m_0\times n}, \mathbf{b}^{m_0\times 1}, q\right)$;
  $m_1 \leftarrow m_0$;
  for $m \in \{\max\{m_{\min}, m_0 - \tau\}, m_0, \min\{m_{all}, m_0 + \tau\}\}$ **do**
    $d \leftarrow m + 1$, $M \leftarrow \sigma^2 m + 1$;
    Construct $\mathbf{B}$ by $\left(\mathbf{A}^{m\times n}, \mathbf{b}^{m\times 1}, q\right)$;
    $T_{\text{total}}, \mathsf{S} \leftarrow \mathtt{EnumBS}(\mathtt{rr}(\mathbf{B}), \sigma^2\chi^2_\star)$;
    if $T_{\min}$ is None or $T_{\min} < T_{\text{total}}$ **then**
      $\mathsf{S}_{\min}, T_{\min}, m_1 \leftarrow \mathsf{S}, T_{\text{total}}, m$;
  if $m_1 = m_0$ **then**
    $\tau \leftarrow \lfloor \frac{\tau}{2} \rfloor$;
  $m_0 \leftarrow m_1$;
return $S_{\min}$, $T_{\min}$, m_0;

Algorithm 6: Our LWE Samples Number Selection Algorithm

Using the optimization strategy for LWE instance number selection, we can solve challenges faster than the G6K default strategy, although its efficiency improvement is not significant (at most 2.2% in the test). See the Table 9.

Table 9: LWE samples improvement simulated result generated by EnumBS with no RAM limit and $\tau = 10$.

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

G The Optimized Strategy for the LWE Challenge

In Table 10, we give the optimized blocksize and jump strategy generated by EnumBS for solving TU Darmstadt LWE Challenge with

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

successfully by running "implement_unsolved_lwechal.sh" in source code [2].

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

In this part, we give the slope and cost comparison of two LWE Challenges under qd float type in Table 11, Table 12 and Table 13, which show the simulated slope and cost are close to the real slope and cost. They also indicate that our PnJBKZ simulator can already reflect how the average of the norms of Gram-Schmidt vectors change during the reduction of PnJBKZ(β, J) on different LWE lattice basis.

From Table 7, Table 11, Table 12 and Table 13 all show that although, at the first round of reduction, the gap between the slope value of simulated GS norms and the slope of real reduced GS norms is slightly bigger due to the influence of the q-ary vector in the initial LWE lattice basis, as the reduction proceeds, in the rounds of reduction before finally entering the Pump, the gap between the slope value calculated by simulation and the slope obtained by real reduction has been sufficiently small. For selecting the optimized blocksize and jump strategy, our PnJBKZ simulator is accurate enough.

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

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

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

Table 11: Quality and wall time (T in seconds) during reduction of LWE Challenge $(n, \alpha) = (45, 0.020)$.

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

Table 12: Quality and log(walltime) (log(T) in seconds) during reduction of LWE Challenge $(n, \alpha) = (50, 0.015)$.

Table 13: Quality and log(walltime) (log(T) in seconds) during reduction of LWE Challenge $(n, \alpha) = (40, 0.025)$.

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