Measuring, simulating and exploiting the head concavity phenomenon in BKZ

Shi Bai[1], Damien Stehlé[2], and Weiqiang Wen[2]

[1] Department of Mathematical Sciences, Florida Atlantic University. Boca Raton.
shih.bai@gmail.com
[2] ENS de Lyon and Laboratoire LIP (U. Lyon, CNRS, ENS de Lyon, INRIA, UCBL).
damien.stehle@ens-lyon.fr, weiqiang.wen@ens-lyon.fr

Abstract. The Blockwise-Korkine-Zolotarev (BKZ) lattice reduction algorithm is central in cryptanalysis, in particular for lattice-based cryptography. A precise understanding of its practical behavior in terms of run-time and output quality is necessary for parameter selection in cryptographic design. As the provable worst-case bounds poorly reflect the practical behavior, cryptanalysts rely instead on the heuristic BKZ simulator of Chen and Nguyen (Asiacrypt'11). It fits better with practical experiments, but not entirely. In particular, it over-estimates the norm of the first few vectors in the output basis. Put differently, BKZ performs better than its Chen–Nguyen simulation.

In this work, we first report experiments providing more insight on this shorter-than-expected phenomenon. We then propose a refined BKZ simulator by taking the distribution of short vectors in random lattices into consideration. We report experiments suggesting that this refined simulator more accurately predicts the concrete behavior of BKZ. Furthermore, we design a new BKZ variant that exploits the shorter-than-expected phenomenon. For the same cost assigned to the underlying SVP-solver, the new BKZ variant produces bases of better quality. We further illustrate its potential impact by testing it on the SVP-120 instance of the Darmstadt lattice challenge.

1 Introduction

A (full-rank) lattice $\mathcal{L}$ of dimension n can be generated by a basis $\mathbf{B}$ made of linearly independent vectors $\mathbf{b}_1, \cdots, \mathbf{b}_n \in \mathbb{R}^n$ via integer combinations: $\mathcal{L}(\mathbf{B}) = \sum_{i \leq n} \mathbb{Z}\mathbf{b}_i$. Lattice reduction aims to compute a basis made of relatively short vectors from an arbitrary input basis. Quantitatively, one measure of quality is the so-called Hermite factor $\mathrm{HF}(\mathbf{B}) = \|\mathbf{b}_1\|/|\det \mathbf{B}|^{1/n} = \|\mathbf{b}_1\|/(\det \mathcal{L})^{1/n}$. Understanding the practical behavior and limits of reduction algorithms is important for setting parameters in lattice-based cryptography. Indeed, the best known attacks against lattice-based schemes typically consist in finding short vectors/bases of lattices provided by publicly available data [APS15].

In [SE94], Schnorr and Euchner proposed a practical lattice reduction algorithm, named the Block Korkine-Zolotarev (BKZ) algorithm. It is parame-

terized by a block-size $\beta \geq 2$: the larger the block-size β, the more expensive in terms of running-time, but the smaller the output Hermite factor. This is because it internally relies on an algorithm that solves the Shortest Vector Problem (SVP) in dimension β, i.e., which can find a shortest non-zero vector in any β-dimensional lattice. Since then, several optimizations of BKZ have been investigated, such as early termination [HPS11] and progressive reduction [CN11,AWHT16]. In [HPS11] (see also [Neu17]), it was shown that in the worst case, BKZ_β (with early termination) achieves a Hermite factor of $\beta^{O(n/\beta)}$ within a polynomial number of calls to the SVP solver, for $\beta = o(n)$ and n growing to infinity. It was shown in [HS08] that there exist bases with such Hermite factors (up to a constant factor in the exponent) which are left unchanged when given as inputs to BKZ_β. Unfortunately, these worst-case bounds are quantitatively very far from experimental data.

The BKZ algorithm proceeds by improving the Gram–Schmidt orthogonalization $\mathbf{B}^* = (\mathbf{b}_1^*, \cdots, \mathbf{b}_n^*)$ of the current basis $\mathbf{B} = (\mathbf{b}_1, \cdots, \mathbf{b}_n)$. More precisely, it aims at updating $\mathbf{B}$ such that the norms $\|\mathbf{b}_1^*\|, \cdots, \|\mathbf{b}_n^*\|$ of the Gram–Schmidt vectors do not decrease too fast. In [Sch03], Schnorr presented a heuristic on the shape of the Gram–Schmidt norms of the output basis, named the Geometric Series Assumption (GSA). It states that there exists a constant $r > 1$ such that the output basis satisfies $\|\mathbf{b}_i^*\|/\|\mathbf{b}_{i+1}^*\| \approx r$ for all $i < n$. Among others, this implies that $\mathrm{HF}(\mathbf{B}) \approx r^{(n-1)/2}$. It was argued in [CN11] (see also [Che09, Ch. 4]) that for β small compared to n, one should have $r \approx (\frac{\beta}{2\pi \mathrm{e}}(\pi\beta)^{\frac{1}{\beta}})^{\frac{1}{\beta-1}}$. The latter value is derived by relying on the Gaussian heuristic[1] to estimate the smallest non-zero norm in a β-dimensional lattice $\mathcal{L}$ by $\mathrm{GH}(\mathcal{L}) := ((\det L)/v_\beta)^{1/\beta}$, where v_β denotes the volume of the β-dimensional unit ball. It was experimentally observed that the GSA is a good first approximation to the practical behavior of BKZ. Nevertheless, it does not provide an exact fit: for $\beta \gtrsim 30$, the typical BKZ output basis has its first few Gram–Schmidt norms and its last $\approx \beta$ Gram–Schmidt norms violate this assumption. These first and last Gram–Schmidt norms are respectively called the *head* and the *tail*, the rest being the *body*. In [CN11], Chen and Nguyen refined the sandpile model from [HPS11] and provided a BKZ simulator based on the Gaussian heuristic (with a modification for the tail, see Subsection 2.3). Their BKZ simulator captures the body and tail behaviors of the Gram–Schmidt norms very precisely [CN11,YD17]. However, as investigated in [CN11,AWHT16,YD17], the Chen–Nguyen simulator fails to capture the head phenomenon: the head almost follows the GSA in the simulations, whereas, in the experiments, the logarithmic Gram–Schmidt norms form a concave curve (instead of a line). Put plainly, BKZ finds shorter vectors than predicted by the Chen–Nguyen simulator. Refer to Figures 1–2 for an example: we run BKZ with block-size 45 on 100-dimensional lattices (generated by the Darmstadt lattice challenge generator)[2] and record the Gram–Schmidt norms of the reduced bases after 2000 tours (each data is averaged over 100 trials). This inaccuracy

[1] The Gaussian heuristic states that a measurable $\mathcal{S} \subseteq \mathbb{R}^n$ should contain $\approx \mathrm{vol}(\mathcal{S})/\det(\mathcal{L})$ points of $\mathcal{L}$.

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

may lead to overestimated security evaluations in cryptographic design. Understanding the head concavity phenomenon was put forward as an important open problem in [YD17], for assessing the bit-security of concrete lattice-based cryptosystems.

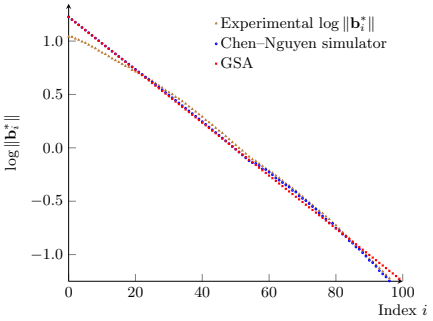

Fig. 1: Gram–Schmidt log-norms for BKZ_{45} at tour $2,000$.

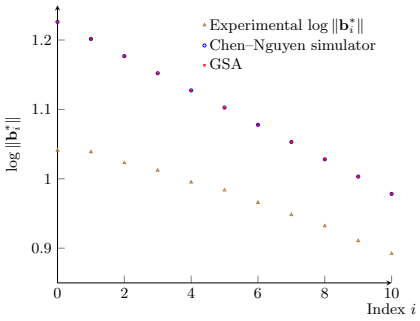

Fig. 2: Same as left hand side, but zoomed in.

Contributions. Our first main contribution is the design of a more accurate BKZ simulator, relying on a probabilistic version of the Gaussian heuristic. More precisely, we take into account the fact that the norm of a shortest non-zero vector of a random lattice is not a fixed quantity driven by the Gaussian heuristic, but a random variable. Concretely, we use a distribution derived from the result on the distribution of short vectors in random lattices by Södergren [Söd11]. We compare our probabilistic simulator and experimental BKZ, and observe that our simulator provides accurate predictions of the head region, while maintaining a good approximation on both body and tail regions. If we focus on the head region, the new simulator is always more precise than the Chen–Nguyen simulator, and similarly accurate for body and tail. Therefore, the Hermite factors estimated by the new simulator are more accurate and fit the experimental results more precisely. This is established through extensive experiments designed to measure the head concavity phenomenon. Such understanding also allows to efficiently assess how it scales for larger block-sizes: when β increases, the head phenomenon decreases, i.e., the GSA is followed more closely.

Our second main contribution is an algorithmic exploitation of the fact that BKZ performs better than the GSA for its first output vectors. We propose a new variant of BKZ, pressed-BKZ, that aims to exploit the head phenomenon everywhere in the graph of Gram–Schmidt norms. To do so, we proceed iteratively: we run BKZ between indices 1 and n, then we freeze the first basis vector and run BKZ between indices 2 and n (i.e., on the appropriately projected basis), then we freeze the first two basis vectors and run BKZ between 3 and n, etc. The bonus of being at the start of the basis is exploited at every position. The output basis tightly follows the GSA in the head and body regions. The gain is that the logarithmic Gram–Schmidt slope is better than that from the

original BKZ. Overall, for the same block-size as in BKZ, pressed-BKZ produces lattice bases of improved quality. We adapt our BKZ simulator to pressed-BKZ, and again, the simulation seems quite accurate, giving further confidence that our simulation correctly captures the head phenomenon.

Another way to exploit the head phenomenon was suggested in [AWHT16]. As the first BKZ blocks are more reduced, solving the corresponding SVP instances is easier. In [AWHT16], Aono *et al.* propose using a larger block-size in the head region than in the rest of the basis. The purpose is to make the head region even better, without increasing the overall cost significantly. We combine this "adaptive block-size" strategy with pressed-BKZ. This allows to accelerate the convergence of pressed-BKZ towards its typical output quality.

Finally, we demonstrate the usefulness of the BKZ variant by testing it on an SVP-120 instance obtained with the Darmstadt lattice challenge generator. We also compare the quality of pressed-BKZ_{60} reduced bases with standard BKZ_β-reduced bases for various block-sizes β.

Impact. For concrete lattice-based cryptosystems with parameters set using the Chen–Nguyen simulator (or the corresponding GSA ratio), the head phenomenon is a potential security risk: as BKZ performs better than what has been taken into account while setting parameters, the parameters were potentially set too low for the targeted bit-security levels. Our simulator, which accurately predicts the head phenomenon, suggests that the head phenomenon vanishes when the block-size becomes large. We conjecture this is because the distribution of the first minimum in random lattices has a standard deviation that decreases to 0 relatively quickly when the lattice dimension increases (this lattice dimension corresponds to the BKZ block-size β). Quantitatively, the phenomenon has almost fully disappeared for $\beta \approx 200$. It is also less important when n is much larger than β. Concrete figures are provided at the end of Subsection 4.4.

The lattice-based submissions to the NIST post-quantum standardization process[3] use conservative security estimates. In particular, they rely on lower bounds for the cost of solving SVP in dimension β, which are significantly below what can currently be achieved in practice (we refer to [ACD+18] for concrete figures). Note that this seems unrelated to the head phenomenon: the BKZ block-sizes needed to break the scheme are often in the hundreds, a range of block-sizes for which the head phenomenon has already vanished. The NIST candidates most impacted are those that were more aggressive in setting their parameters, though the impact remains limited even for them.

Oppositely, for block-sizes that can be handled in practice (e.g., $\beta \lesssim 100$), the head concavity phenomenon is non-negligible, and can be exploited. Our work can then help make concrete cryptanalysis more accurate. By allowing one to solve SVP in larger dimensions β using pressed-$\mathrm{BKZ}_{\beta'}$ with $\beta' < \beta$ as a preprocessing, our work should allow one to perform BKZ in larger block-sizes β. It is well-known that for small block-sizes (say $\beta \lesssim 35$), the BKZ output quality is inaccurately predicted by the Gaussian Heuristic (see [CN11], for example). This phenomenon vanishes when the block-size becomes higher. But then BKZ_β ben-

[3] `https://csrc.nist.gov/Projects/Post-Quantum-Cryptography`

efits from the head concavity phenomenon. As a result, extrapolating concrete experiments in such block-sizes to draw conclusions in much larger block-sizes seems to amount to wild guessing. On the other hand, we have a better simulation of the head phenomenon for BKZ with practical block-sizes. By exploiting the head phenomenon, we can hope to reach higher block-sizes, for which such small block-size effects do not occur anymore. In this smoother regime, extrapolating experiments should become sounder.

Related works. The first simulator for predicting the Gram–Schmidt norms of a BKZ-reduced basis was proposed by Chen and Nguyen in [CN11]. It relies on the assumption that each SVP-solver in the projected local block finds a lattice vector whose norm exactly matches its Gaussian heuristic estimate for that local block, except for a few blocks at the end of the basis. It is a good first approximation, but remains inaccurate in two ways. First, it does not capture the head concavity phenomenon (which is reported nevertheless in the experiments of [CN11]). Second, it does not take into account that in practice it is preferable to use heuristic SVP-solvers which may miss the optimal solutions. The main such heuristic SVP-solver is pruned enumeration, introduced in [SE94] and refined and improved in [GNR10]. It consists of pruning the enumeration tree by keeping only the nodes that are most likely to lead to interesting leaves. As a result, only a subset of lattice points are enumerated within the required radius, and the optimal solution may be missed. Extreme pruning [GNR10] goes even further: it decreases the probability of finding a shortest non-zero vector to lower the time/probability ratio, and runs the process several times to boost the success probability. Each time, the lattice basis is re-randomized and reduced with a lower block-size to prepare for the enumeration. An alternative approach for solving SVP is lattice sieving. The fast sieving variants, such as [NV08,MV10,Duc18,BDGL16], are also not guaranteed to return a shortest non-zero lattice vector.

In [AWHT16], Aono *et al.* described the so-called progressive-BKZ. The main new ingredient is that the latter tries to avoid the re-randomization/preprocessing overheads by using a single enumeration in any SVP call. For this, the authors increase the search radius in the enumeration, aiming to find a short vector but not necessarily a shortest one by pruning the enumeration tree. This search radius is adaptively derived from the current basis quality. Since the authors are not in the regime of finding a shortest non-zero vector, to estimate the success probability, the authors model lattice points of norm below the search radius as random points in the ball of that radius (see [AWHT16, Lemma 1]). This pruned enumeration with increased search radius heuristically produces a non-zero vector in lattice Λ of norm $\frac{\beta}{\beta+1} \cdot \alpha \cdot \mathrm{GH}(\Lambda)$ for some $\alpha \geq 1$, using their random point model. For $\alpha = 1$, we obtain an expectation for the first minimum that is lower than the Gaussian heuristic. Aono *et al.* also adapted the Chen–Nguyen simulator by modifying the expected norm found by the SVP-solver using the random points model rather than the Gaussian heuristic value. Note that the updated simulator takes some probabilistic phenomenon into account but remains deterministic. In particular, it does not capture the head concavity phenomenon.

Finally, as mentioned earlier, Aono *et al.* also experimentally observed the head concavity phenomenon, and proposed to exploit it by using BKZ with larger block-size on the first few blocks.

Yu and Ducas [YD17] ran extensive experiments to assess the practical behavior of BKZ. They have two main experimental observations. First, the distribution of differences $v_i := \log \|\mathbf{b}_i^*\| - \log \|\mathbf{b}_{i+1}^*\|$ between two consecutive Gram–Schmidt log-norms, varies as a function of the index i when i belongs to the head and tail regions (and it does not in the body region). Second, the covariance between v_i and v_{i+2} is 0 for all i, but v_i and v_{i+1} are negatively correlated: in the head and tail regions, their covariance depends on both i and the block-size β, but in the body region only the block-size β contributes to their covariance. These observations quantify the head concavity phenomenon more precisely.

Software. The BKZ experiments were run using the `fplll` [dt16] (version 5.2.0) and `fpylll` [dt17] (version 0.4.0dev) open-source libraries. The efficiency of these libraries for large block sizes $\beta \geq 50$ was essential for obtaining useful statistics. Our simulator, coded in Python, and the BKZ variants, coded in C++, are freely available.

As mentioned earlier, we report experiments on pressed-BKZ with an adaptive block-size strategy, for the SVP-120 challenge. We expect our BKZ improvements to be useful in larger dimensions as well (e.g., SVP-150), if given sufficient computational resources. We want to stress that our primary goal is to model, predict and exploit the head phenomenon, the SVP-120 experiment being an illustration of its relevance.

Auxiliary material. We provide a significant amount of material to make our results reproducible and to report experimental observations in more details. Concretely, we provide codes (as mentioned above), experimental raw data, graphs that could not fit within the page limit and video files. Data and links are provided on the authors' webpages.

2 Preliminaries

In this section, we recall some basic facts on lattices and lattice reduction. We refer the reader to the survey [NV09] for more background. We first introduce the notations used throughout the paper.

Notations. We let lower-case bold letters denote (column) vectors and upper-case bold letters denote matrices. For a vector $\mathbf{x}$, we use $\|\mathbf{x}\|$ to denote its ℓ_2-norm. Similarly, a matrix $\mathbf{B} = (\mathbf{b}_1, \cdots, \mathbf{b}_n)$ is also parsed column-wise. For $n \geq 1$ and $r > 0$, we let $V_n(r)$ denote the volume of an n-dimensional ball with radius r and v_n the volume of an n-dimensional unit ball. Correspondingly, we let c_n denote the radius of an n-dimensional ball of unit volume. Logarithms are in base 2. For $\lambda > 0$, we let $\mathrm{Expo}(\lambda)$ denote the exponential distribution with density function proportional to $x \mapsto \lambda \mathrm{e}^{-\lambda x}$, up to a normalization factor. We let log denotes the natural logarithm with base e.

2.1 Euclidean lattices

Let $\mathbf{B} \in \mathbb{R}^{n \times n}$ be full rank. The lattice $\mathcal{L}$ generated by $\mathbf{B}$ is $\mathcal{L}(\mathbf{B}) = \{\mathbf{Bx} \mid \mathbf{x} \in \mathbb{Z}^n\}$, and the matrix $\mathbf{B}$ is called a basis of $\mathcal{L}$. We let $\mathbf{B}^* = (\mathbf{b}_1^*, \cdots, \mathbf{b}_n^*)$ denote the Gram–Schmidt orthogonalization of $\mathbf{B}$. The determinant of a lattice $\mathcal{L}$ with basis $\mathbf{B}$ is defined as $\det(\mathcal{L}) = \prod_{i \leq n} \|\mathbf{b}_i^*\|$. The norm of a shortest non-zero vector in $\mathcal{L}$ is denoted by $\lambda_1(\mathcal{L})$ and called the minimum of $\mathcal{L}$. Minkowski's theorem asserts that $\lambda_1(\mathcal{L}) \leq 2 \cdot v_n^{-1/n} \cdot \det(\mathcal{L})^{1/n}$. For $i \leq n$, we let π_i denote the orthogonal projection onto the linear subspace $(\mathbf{b}_1, \cdots, \mathbf{b}_{i-1})^{\perp}$. For $i < j \leq n$, we let $\mathbf{B}_{[i,j]}$ denote the local block $(\pi_i(\mathbf{b}_i), \cdots, \pi_i(\mathbf{b}_j))$, and $\mathcal{L}_{[i,j]}$ denote the lattice generated by $\mathbf{B}_{[i,j]}$.

Lattice reduction. A lattice basis $\mathbf{B}$ is called size-reduced, if it satisfies $|\mu_{i,j}| \leq 1/2$ for $j < i \leq n$ where $\mu_{i,j} = \langle \mathbf{b}_i, \mathbf{b}_j^* \rangle / \langle \mathbf{b}_j^*, \mathbf{b}_j^* \rangle$. A basis $\mathbf{B}$ is HKZ-reduced if it is size-reduced and further satisfies:

$$\|\mathbf{b}_i^*\| = \lambda_1(\mathcal{L}_{[i,n]}), \ \forall i \leq n.$$

A basis $\mathbf{B}$ is BKZ-β reduced for block size $\beta \geq 2$ if it is size-reduced and satisfies:

$$\|\mathbf{b}_i^*\| = \lambda_1(\mathcal{L}_{[i,\min(i+\beta-1,n)]}), \ \forall i \leq n.$$

Heuristics. Lattice reduction algorithms and their analyses often rely on heuristic assumptions. Let $\mathcal{L}$ be an n-dimensional lattice and $\mathcal{S}$ a measurable set in the real span of $\mathcal{L}$. The *Gaussian Heuristic* states that the number of lattice points in $\mathcal{S}$, denoted $|\mathcal{L} \cap \mathcal{S}|$, is about $\mathrm{vol}(\mathcal{S}) / \det(\mathcal{L})$. In particular, taking $\mathcal{S}$ as a centered n-ball of radius R, the number of lattice points contained in the n-ball is about $V_n(R) / \det(\mathcal{L})$. Furthermore, by setting $V_n(R) \approx \det(\mathcal{L})$, we see that $\lambda_1(\mathcal{L})$ is about $\mathrm{GH}(\mathcal{L}) := v_n^{-1/n} \cdot \det(\mathcal{L})^{1/n}$. Note that this is a factor of 2 smaller than the rigorous upper bound provided by Minkowski's theorem. In [Sch03], Schnorr introduced the *Geometric Series Assumption* (GSA), which states that the Gram-Schmidt norms $\{\|\mathbf{b}_i^*\|\}_{i \leq n}$ of a BKZ-reduced basis behave as a geometric series, i.e., there exists $r > 1$ such that $\|\mathbf{b}_i^*\| / \|\mathbf{b}_{i+1}^*\| \approx r$ for all $i < n$.

Random lattices. We use $\mathit{\Gamma}_n = \{\mathcal{L} \in \mathbb{R}^n \mid \mathrm{vol}(\mathcal{L}) = 1\}$ to denote the set of all full-rank lattices of rank n with unit volume. The distribution of short(est) vectors in random lattices uniformly chosen in $\mathit{\Gamma}_n$ was studied, among others, in [Rog56,Sch59,Söd11]. In [Che09], Chen proposed the following statement as a direct corollary of [Söd11, Thm. 1].

Theorem 1. *[Che09, Cor. 3.1.4] Sample $\mathcal{L}$ uniformly in $\mathit{\Gamma}_n$. The distribution of $v_n \cdot \lambda_1(\mathcal{L})^n$ converges in distribution to* $\mathrm{Expo}(1/2)$ *as $n \to \infty$.*

If we set $\lambda_1(\mathcal{L})$ as a random variable $Y = X^{1/n} \cdot \mathrm{GH}(\mathcal{L})$, with X sampled from $\mathrm{Expo}(1/2)$, then the expected value of λ_1 is

$$\mathbb{E}(\lambda_1(\mathcal{L})) = 2^{1/n} \cdot \mathit{\Gamma}\left(1 + 1/n\right) \cdot \mathrm{GH}(\mathcal{L}).$$

In lattices of unit volume, the $\mathrm{GH}(\mathcal{L})$ term can be replaced by $v_n^{-1/n}$. In the rest of this paper, we refer to this quantity as the minimum expectation. For large n, this is $\approx (1 + 0.116/n + o(1/n)) \cdot \mathrm{GH}(\mathcal{L})$. It can be also seen that the variance is

$$\mathbb{V}(\lambda_1(\mathcal{L})) = 2^{2/n} \cdot \left(\Gamma\left(1 + 2/n\right) - \left(\Gamma\left(1 + 1/n\right)\right)^2\right) \cdot \left(\mathrm{GH}(\mathcal{L})\right)^2,$$

which is $\approx \frac{\pi^2}{6n^2}(1 + o(1)) \cdot (\mathrm{GH}(\mathcal{L}))^2$ for large n.

2.2 The BKZ algorithm

The Schnorr-Euchner BKZ algorithm [SE94] takes as inputs a block-size β and a basis $\mathbf{B} = (\mathbf{b}_1, \cdots, \mathbf{b}_n)$ of a lattice Λ, and outputs a basis which is "close" to being BKZ_β-reduced (up to numerical inaccuracies, as the underlying Gram–Schmidt orthogonalization is computed in floating-point arithmetic, and up to the progress parameter $\delta < 1$). BKZ can be seen as a practical variant of Schnorr's algorithm from [Sch87]. BKZ starts by LLL-reducing the input basis, then calls an SVP-solver on consecutive local blocks $\mathbf{B}_{[k,\min(k+\beta-1,n)]}$ for $k = 1, \cdots, n-1$. This is called a *BKZ tour*. After each execution of the SVP-solver, if we have $\lambda_1(\Lambda_{[k,\min(k+\beta-1,n)]}) < \delta \cdot \|\mathbf{b}_k^*\|$, then BKZ updates the block $\mathbf{B}_{[k,\min(k+\beta-1,n)]}$ by inserting the vector found by the SVP-solver between indices $k-1$ and k, and LLL-reducing the updated block (in this case, the input is a generating set instead of a basis). Otherwise, we LLL-reduce the local block directly, without any insertion. The procedure terminates when no change occurs at all during a tour. We refer to Algorithm 1 for a complete description of the BKZ algorithm.

Algorithm 1 The Schnorr and Euchner BKZ algorithm

Input: A basis $\mathbf{B} = \{\mathbf{b}_1, \cdots, \mathbf{b}_n\}$, a block size $\beta \geq 2$ and a constant $\delta < 1$.
Output: A BKZ_β-reduced basis of $\Lambda(\mathbf{B})$.

repeat
  for $k = 1$ `to` $n-1$ **do**
    `Find any b such that` $\|\pi_k(\mathbf{b})\| = \lambda_1(\Lambda_{[k,\min(k+\beta-1,n)]})$
    if $\delta \cdot \|\mathbf{b}_k^*\| > \|\mathbf{b}\|$ **then**
      LLL-reduce$(\mathbf{b}_1, \cdots, \mathbf{b}_{k-1}, \mathbf{b}, \mathbf{b}_k, \cdots, \mathbf{b}_{\min(k+\beta,n)})$.
    else
      LLL-reduce$(\mathbf{b}_1, \cdots, \mathbf{b}_{\min(k+\beta,n)})$.
    end if
  end for
until no change occurs.

For practical reasons, there are diverse BKZ variants.

- Early-abort. The BKZ reduction aborts when a selected number of tours are completed or when a desired output quality has been reached [HPS11].
- SVP-solver. BKZ could be run with any SVP solver; in practice for typical block sizes, the fastest one is lattice enumeration [Kan83,FP83]; the latter can be significantly accelerated with tree pruning [SE94] and even further with extreme pruning [GNR10]. In the case of enumeration with pruning, the SVP solver is not guaranteed to return a shortest non-zero vector in

its input lattice. Furthermore, one can set the enumeration radius to either $\|\mathbf{b}_i^*\|$ or Gaussian heuristic (whichever is smaller). In the experiments of Sections 3 and 4, we use the enumeration radius $0.99 \cdot \|\mathbf{b}_i^*\|$, which is the default choice in the implementation of BKZ in `fplll` [dt16]. In the experiments of Section 5.4, we set the enumeration radius to be 1.05 times the Gaussian heuristic of the local block. In all these cases, the SVP solver is not guaranteed to return a shortest non-zero vector.

- Pre-processing and post-processing. Pre-processing is performed *before* the call to the SVP solver. In the pre-processing step, some strategies (e.g., BKZ with a smaller block size) are chosen to further improve the basis. Post-processing is executed *after* the call to the SVP-solver, e.g., running LLL on indices 1 to $\min(k + \beta - 1, n)$, in order to propagate the progress made at index k.

2.3 The Chen–Nguyen simulator

In [CN11], Chen and Nguyen proposed a simulator to capture the practical behavior of BKZ with relatively large block size (e.g., $\beta \geq 45$). The goal was to estimate the practical behavior of BKZ for hard-to-solve instances. Overall, the simulation proceeds closely to BKZ. It considers successive tours. For each tour, it computes new Gram-Schmidt log-norms $\hat{\ell}_1, \ldots, \hat{\ell}_n$ from current Gram-Schmidt log-norms $\ell_1, \ldots, \ell_n$. At the beginning of each BKZ tour, a boolean flag τ is initialized to be `true`. To update each local block, the simulator first (deterministically) compute the Gaussian heuristic value $\mathrm{GH}(\mathbf{B}_{[k,\min(k+\beta-1,n)]})$ as an estimation of first miminum, by looking at the Gram–Schmidt norms of the current local block (except for small blocks in the end). This corresponds to `Line 8` of Algorithm 2, we recall that the v_d denotes the volume of d-dimensional unit ball. The computed Gaussian heuristic value is then used to update the current local block, if it is smaller than the current $\|\mathbf{b}_k^*\|$ (as written in `Line 10--11`). Else, the local block is kept unchanged. To update, the first Gram–Schmidt norm is replaced by the selected value, and the boolean flag τ is flipped once such an update occurs (as written in `Line 12`). Once the boolean flag τ is changed, all the remaining Gram–Schmidt norms, of indices $k' \in [k+1, n-45]$ are updated one by one to $\mathrm{GH}(\mathbf{B}_{[k',\min(k'+\beta-1,n)]})$, independently of whether the current $\|\mathbf{b}_k^*\|$ is already small enough or not (as written in `Line 15`). At the end of each tour, there is an additional update of the tail block of length 45 (as written in `Line 19--21`). The Gram–Schmidt norms in this tail block are simulated with the experimental Gram–Schmidt norms of HKZ-reduced bases of 45-dimensional unit-volume lattices. The experimental Gram–Schmidt norms are prepared in `Line 1--2` of Algorithm 2. This length of 45 was chosen because the minimum of blocks of dimension ≥ 45 within BKZ follows the Gaussian heuristic quite well (as observed by the extensive experiments of [CN11]). This special treatment on the tail block also make the simulator more precise for capturing the practical behavior of BKZ compared to GSA assumption.

Algorithm 2 The Chen–Nguyen BKZ simulator

Input: The Gram–Schmidt log-norms $\{\ell_i = \log \|\mathbf{b}_i^*\|\}_{i \le n}$ and an integer $N \ge 1$.

Output: A prediction of the Gram–Schmidt log-norms $\{\widehat{\ell}_i = \log \|\mathbf{b}_i^*\|\}_{i \le n}$ after N tours of BKZ.

for $i = 1$ `to` 45 **do** $r_i \leftarrow \mathbb{E}[\log \|\mathbf{b}_k^*\| : \mathbf{B}$ HKZ-reduced basis of $\Lambda \leftarrow \Gamma_{45}]$
end for
for $j = 1$ `to` N **do**
  $\tau \leftarrow$ `true`
  for $k = 1$ `to` $n - 45$ **do**
    $d \leftarrow \min(\beta, n - k + 1)$; $e \leftarrow k + d - 1$
    $\log \mathrm{vol}(\Lambda_{[k,e]}) \leftarrow \sum_{i=1}^{e} \ell_i - \sum_{i=1}^{k-1} \widehat{\ell}_i$
    $g \leftarrow \left(\log \mathrm{vol}(\Lambda_{[k,e]} - \log v_d\right)/d$
    if τ `=` `true` **then**
      if $g < \ell_k$ **then**
        $\widehat{\ell}_k \leftarrow g$
        $\tau \leftarrow$ `false`
      end if
    else
      $\widehat{\ell}_k \leftarrow g$
    end if
  end for
  $\log \mathrm{vol}(\Lambda_{[k,e]}) \leftarrow \sum_{i=1}^{n} \ell_i - \sum_{i=1}^{n-45} \widehat{\ell}_i$
  for $k' = n - 44$ `to` n **do**
    $\widehat{\ell}_{k'} \leftarrow \frac{\log \mathrm{vol}(\Lambda_{[k,e]})}{45} + r_{k'+45-n}$
  end for
  $\{\ell_1, \cdots, \ell_n\} \leftarrow \{\widehat{\ell}_1, \cdots, \widehat{\ell}_n\}$
end for

3 Measuring the head concavity

In this section, we describe in detail the concavity phenomenon in the leading Gram–Schmidt log-norms. In particular, we report experiments on the quality of bases output by BKZ_β and on the evolution of Gram–Schmidt norms during the execution of the algorithm.

In our experiments, we consider the knapsack-type lattice bases generated by the Darmstadt lattice challenge generator. In dimension n, the generator selects a prime p of bitsize $10 \cdot n$ and sets the first basis vector as $(p, 0, \cdots, 0)$. For $i > 1$, the i-th basis vector starts with a uniformly chosen integer modulo p, and all other entries are 0 except the i-th entry which is 1. When using the generator file, the `seed` is from the set $\{0, \cdots, k-1\}$, where k is total number of samples in the experiment, which enables reproducibility. We always run LLL reduction before a BKZ reduction. We use the default LLL in `fplll` of parameter $\delta = 0.99$.

3.1 BKZ output quality

In our first set of experiments, we measure the output quality of the BKZ algorithm. We consider the final reduced basis, for increasing block sizes β. We let BKZ run until it fully stops and use a full enumeration as SVP-solver (without pruning), to avoid side-effects. We then measure the Gram–Schmidt log-norms $\{\log \|\mathbf{b}_i^*\|\}_{i \le n}$ of the reduced basis $\mathbf{B}$. In particular, when BKZ completes, the vector $\mathbf{b}_i^*$ is a shortest non-zero vector of the lattice $\Lambda_{[i,i+\min(i+\beta-1,n)]}$ (up to the 0.99 factor), for every i (see Subsection 2.2).

As we use BKZ until exhaustion with full enumeration, the experiments are quite lengthy. We restricted them to dimension $n = 100$, with selected block sizes β ranging from 4 to 40. For each choice of β, we conduct the experiment 100 times using input lattices generated with different seeds. For each experiment, we normalize the $\log \|\mathbf{b}_i^*\|$'s of the reduced basis by substracting one hundredth of the logarithmic determinants of its input lattice, such that the summation of the new logarithmic Gram–Schmidt norms is normalized to be 0. This step helps eliminate the small differences of determinants of all generated lattices. We then average $\log \|\mathbf{b}_i^*\|$ for each i over the 100 samples.

We plotted the results for the various block sizes in Figures 3–8. The x-axis corresponds to the basis vector at index i and the y-axis is the Gram–Schmidt log-norm. Each figure contains several plots: the red dots are the (averaged) experimental $\log \|\mathbf{b}_i^*\|$'s; the brown dots are obtained by applying the first Minkowski's theorem to each one of the experimentally obtained local blocks $\mathbf{B}_{[i,i+\min(i+\beta-1,n)]}$ of the output basis; the purple dots are the values obtained by replacing Minkowski's theorem by the Gaussian heuristic value $\mathrm{GH}(\Lambda_{[i,i+\min(i+\beta-1,n)]})$, and the blue dots are the expected $\lambda_1(\Lambda_{[i,i+\min(i+\beta-1,n)]})$ (see Subsection 2.1).

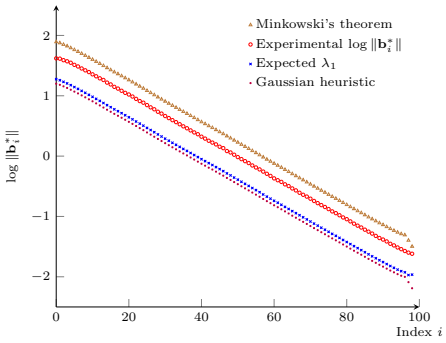

Fig. 3: Output of BKZ_4.

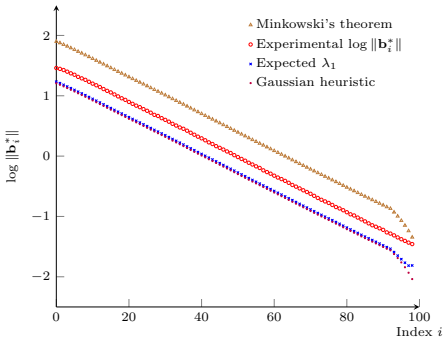

Fig. 4: Output of BKZ_8.

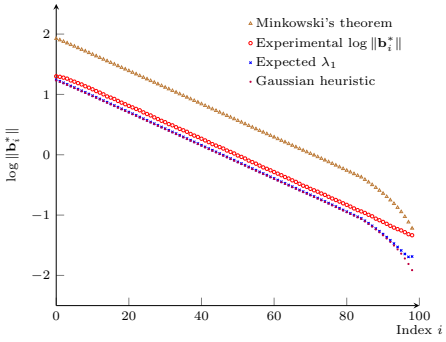

Fig. 5: Output of BKZ_{16}.

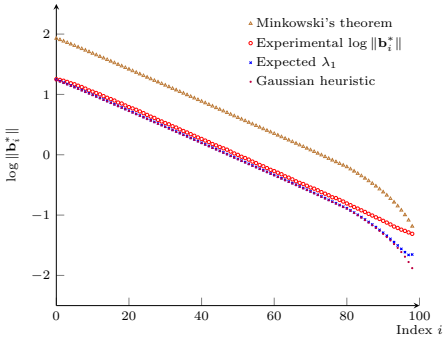

Fig. 6: Output of BKZ_{20}.

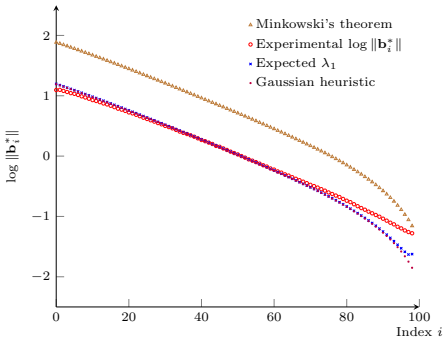

Fig. 7: Output of BKZ_{30}.

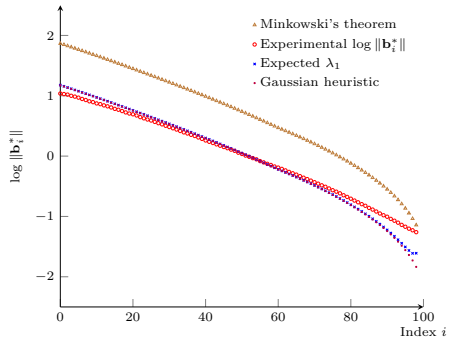

Fig. 8: Output of BKZ_{40}.

The experiments highlight that the Gaussian heuristic and expected value of first minimum are not very accurate for predicting the output of the BKZ algorithm (and neither is Minkowski's theorem, but that is less surprising). For small block sizes, the experimental Gram–Schmidt log-norms are above the Gaussian heuristic values. Notice this even appears to happen for the tail blocks for large β. When the block size increases, the Gaussian heuristic and first minimum expectation get closer to each other (except in the tail region), but still do not accurately predict the genuine BKZ output. In particular, the experimental Gram–Schmidt log-norms are concave in the head region (the other curves are also somewhat concave, but less so, as they are smoothed versions of the experimental curve). This is essentially the same as the phenomenon we already observed in Section 1, which we refer to as head concavity. It starts being quite noticeable with $\beta \approx 30$.

3.2 Enumeration costs in local blocks

The enumeration cost for SVP in each local block is also an interesting quantity for evaluating the extent of the head concavity of a BKZ-reduced basis. As explained in [HS07], under the Gaussian heuristic, the full enumeration cost (in terms of number of nodes enumerated, denoted by "# nodes") of a d-dimensional lattice using enumeration radius $\|\mathbf{b}_1^*\|$ can be estimated by

$$\sum_{k=1}^{d} \frac{1}{2} \cdot \frac{V_k(\|\mathbf{b}_1^*\|)}{\prod_{i=d-k+1}^{d} \|\mathbf{b}_i^*\|}. \tag{1}$$

We take the (averaged) BKZ_{40} preprocessed basis from the previous subsection and compute the local SVP costs of BKZ_{50} and BKZ_{60} on the BKZ_{40}-preprocessed basis. We also compute the local SVP costs of BKZ_{40} which can be considered as the cost for checking that the basis is indeed BKZ_{40}-reduced. In Figure 9, we plot the logarithm of the quantity above for each local block of SVP_{40}, SVP_{50} and SVP_{60}. It can be seen that the local SVP costs in the first few blocks are cheaper. The enumeration costs keep increasing until the last $\beta - 1$ blocks. These last blocks are of smaller dimensions, explaining why

their enumeration costs become lower. As cheaper enumeration reflects stronger reducedness, Figure 9 hints at a concavity of the $\log \|\mathbf{b}_i^*\|$'s in the head.

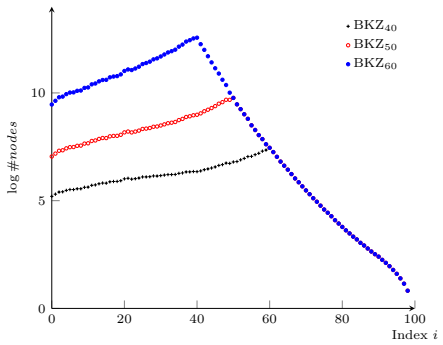

Fig. 9: Estimated enumeration costs (of each local block) for BKZ_{40}, BKZ_{50} and BKZ_{60} on a BKZ_{40} reduced basis.

3.3 Evolution of the Gram–Schmidt norms during the execution

The previous experiments suggest that the Gaussian heuristic may be inaccurate for a BKZ-reduced basis in the head region. Below, we further investigate the *evolution* of the accuracy of the Gaussian heuristic for each local block $\Lambda_{[i,\min(i+\beta,n)]}$ during the running of the BKZ algorithm. We focus on the evolution of the BKZ_{40} experiments from Subsection 3.1. After each BKZ tour, we record $\{\mathbf{b}_i^*\}_{i\in[n]}$ for each experiment. Again we use a full enumeration as SVP-solver (without pruning), to avoid side-effects and wait until BKZ completes.

In Figure 10, we plot the Gram–Schmidt log-norms after each tour (for the first 1000 tours, plus those of the initial LLL-reduced input). For each BKZ_{40} experiment, we normalize the log-norms after each tour as in Subsection 3.1. Furthermore, we take the average of the log-norms for the 100 experiments (one individual graph would be less smooth). Finally, we plot the log-norms for the first 1000 tours (one BKZ instance completes before 1000 tours; and after this one completes, for a given tour number, we take the average over the BKZ experiments that are running for more than 1000 tours). The dots corresponding to the earlier tours are colored in blue, and those corresponding to the later tours are colored in red (the color changes gradually).

The plot shows the evolution of the log-norms across tours. However, it does not clearly highlight the evolution of the relation between the $\|\mathbf{b}_i^*\|$'s and the Gaussian heuristic values. Hence we further compute the quantities

$$\frac{\|\mathbf{b}_i^*\|}{\mathrm{GH}(\Lambda_{[i,\min(i+\beta,n)]})} \quad \text{for} \quad i \leq n.$$

Note that this should be expected to be close to 1 for a random lattice, under the Gaussian heuristic. For each tour, for all indices i, we record all the (averaged)

quantities at i's across the 100 experiments. We plot a line for each tour and hence Figure 11 contains 1001 lines. In Figure 11, the x-axis corresponds to the index i; the y-axis corresponds to the quantities above. Note that for each i, there are 1001 dots vertically, corresponding to the number of BKZ tours plus the initial LLL-reduced input.

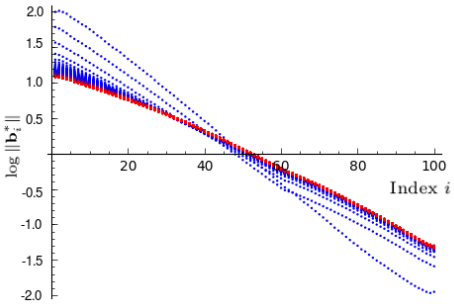

Fig. 10: Evolution of the Gram–Schmidt log-norms during BKZ_{40}'s execution.

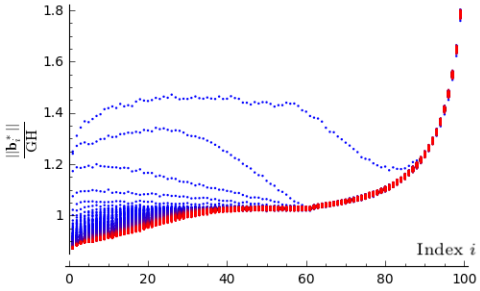

Fig. 11: Evolution of the $\|\mathbf{b}_i^*\|/\mathrm{GH}$'s during BKZ_{40}'s execution.

As observed in prior works, BKZ distorts the distribution of the projected lattices: the first projected sublattices are denser (the minimum is smaller) and the last projected sublattices are sparser. This can be seen from Figure 11 since the red points are significantly lower than 1 in the first indices. Further, this distortion occurs often quickly during the execution of BKZ, sometimes within a few tours.

3.4 Evolution of root Hermite factor of the basis

We now consider the asymptote of the root Hermite factor of the basis being BKZ-reduced, as the number of BKZ tours increases. A similar experiment was done in [HPS11]. We compare the experimental behavior to the Chen–Nguyen simulator. Note that the root Hermite factor only measures the head concavity phenomenon for the first basis vector. We fix the block size at $\beta = 45$ and run BKZ_β on 100 random instances. After each tour, we record the average root Hermite factor. In Figure 23 (we also duplicate it here for convenience), we plot the averaged root Hermite factors over all experiments. The root Hermite factor δ is computed as:

$$\delta = \left(\|\mathbf{b}_1^*\| / \left(\det \Lambda\right)^{1/n}\right)^{1/n}.$$

It can be seen that the evolution of the root Hermite factor does not match with its prediction by the Chen–Nguyen simulator. Indeed, the root Hermite factor

obtained with the Chen–Nguyen simulator does not further improve after the first few tours; while it keeps improving in the actual experiments. It should also be noted that in the first few tours, the root Hermite factors in the actual experiments are worse than those of the Chen–Nguyen simulation. In the experiments, we do not use pruned enumeration nor early-abort. One potential reason is that the local SVP solver used only attempts to find a vector slightly smaller than $\|\mathbf{b}_i^*\|$ (instead of the Gaussian heuristic value). As the number of tours increases, the root Hermite factors in experiments become smaller than those obtained by the Chen–Nguyen simulator.

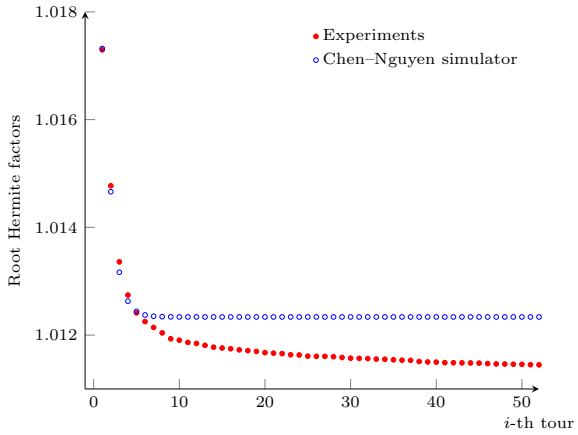

Fig. 12: Evolution of Root Hermite factors during the execution of BKZ_{45}.

3.5 BKZ with pruning

All of the previous experiments used BKZ without pruning to avoid side-effects on the quality. The purpose of this subsection is to show that using extreme pruning within the enumeration indeed affects the behavior of BKZ to some extent (this was also observed in [YD17]). Nevertheless, the head concavity phenomenon remains visible even in pruned-enumeration BKZ. Again we run the BKZ experiments until they fully complete (i.e., no early-abort).

We consider two experiments. First, we run BKZ_{40} with pruned enumeration and compare it with standard BKZ_{40}.We used the default pruning strategy of `fplll`. We note that there is no known canonically best way to prune, and the experimental results may vary a little across different pruning strategies. We see by comparing Figure 13 (with pruned enumeration) with Figure 8 (without) that the extent of the head concavity phenomenon is less than without pruned enumeration. In the second experiment, we run BKZ with pruned enumeration with larger block size, which is also more relevant to cryptanalysis. In particular, we run BKZ_{60} with pruned enumeration and then plot the evolution of the corresponding $\|\mathbf{b}_i^*\|$/GH's (for the first 500 tours) in Figure 14. This is to be

compared with Figure 11. We can conclude that the head concavity phenomenon still exists for practical versions of BKZ with larger block sizes.

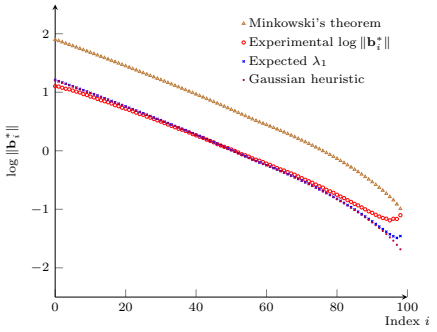

Fig. 13: Output Gram–Schmidt log-norms for BKZ_{40} with pruning.

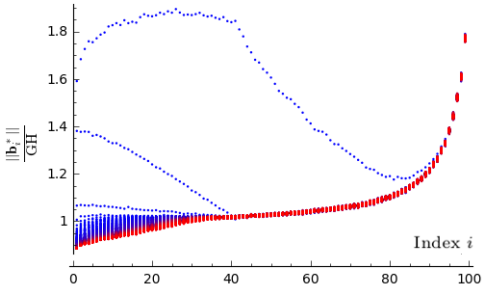

Fig. 14: Evolution of the $\|\mathbf{b}_i^*\|/\mathrm{GH}$'s during the execution of BKZ_{60} with pruning.

4 A refined BKZ simulator

In this section, we describe a refined BKZ simulator, and report on experiments indicating that the simulation is quite accurate, in particular in capturing the head concavity phenomenon.

4.1 The refined simulator

Our probabilistic BKZ simulator aims to provide a more accurate simulation of the experimental behavior of the BKZ algorithm. Our simulator has similar structure as the Chen–Nguyen simulator (refer to Subsection 2.2), in particular, we also consider the Gram–Schmidt log-norms. There are several differences with the Chen–Nguyen simulator.

The main difference is the emulation of the probabilistic nature of the minimum in random lattices. Theorem 1 provides the distribution of the minimum of a uniform unit-volume lattice. The new BKZ simulator, given as Algorithm 3, is probabilistic. It takes this distribution into consideration when updating each local block. In more detail, suppose that we are updating the local block $\Lambda_{[k,e]} = \Lambda(\mathbf{B}_{[k,e]})$ for some $k \leq n - 45$ and $e = \min(k + \beta - 1, n)$ with dimension $d = \min(\beta, n - k + 1)$. Let us assume this is a random lattice. By Theorem 1, we have that $\lambda_1(\Lambda_{[k,e]})$ is distributed as

$$\lambda_1(\Lambda_{[k,e]}) = \left(\frac{X \cdot \mathrm{vol}(\Lambda_{[k,e]})}{v_d}\right)^{1/d},$$

where X is sampled with distribution Expo[1/2]. Recall that v_d is the volume of n-dimensional unit ball. Now we can take the logarithm to obtain

$$\log \lambda_1(\Lambda_{[k,e]}) = \frac{\log X + \log \mathrm{vol}(\Lambda_{[k,e]}) - \log v_d}{d}.$$

This explains `Line 14` in Algorithm 3. In `Line 15`, the simulator checks whether a value sampled as above, e.g., $\log \|\mathbf{b}_k^*\|$, is smaller than the current $\log \|\mathbf{b}_k^*\|$. If it is indeed smaller, then the value of $\log \|\mathbf{b}_k^*\|$ is updated. Else the former one is kept. This is corresponding to the main step in the BKZ algorithm (`Line 4--7`): once the found vector (in the SVP call) is shorter than current $\mathbf{b}_i^*$ in current local block, then the found vector will be used to replace $\mathbf{b}_i$, thus $\mathbf{b}_i^*$ is updated. Otherwise the found vector will be discarded, and the simulation assumes that $\mathbf{b}_i^*$ is not changed during the LLL-reduction.

A further difference with the Chen–Nguyen simulator is the way we handle the remaining Gram–Schmidt log-norms in the current local block in case $\log \|\mathbf{b}_k^*\|$ has been updated. In the Chen–Nguyen simulator, after updating the first Gram–Schmidt log-norms $\log \|\mathbf{b}_k^*\|$ in the current local block, all the remaining log-norms $\log \|\mathbf{b}_i^*\|$ for $i > k$ will be updated by using the Gaussian heuristic directly without further checking whether the estimated value gives an improvement or not (refer to Algorithm 2). In our simulator, we consider a refined update of the remaining $\log \|\mathbf{b}_i^*\|$'s of the block. Concretely, we update all the remaining indices in the current local block by increasing them by a common amount chosen so that the volume of current block is preserved (it compensates for the decrease of $\log \|\mathbf{b}_k^*\|$). There is one further subtlety in the actual update applied by the simulator. For the second Gram–Schmidt log-norm of the block, it sets $\widehat{\ell}_{k+1} \leftarrow \ell_k + \log(\sqrt{1-1/d})$ rather than increasing it by the same amount as for the $d-2$ remaining log-norms. The quantity $\sqrt{1-1/d}$ is used to simulate the change in norm for the old vector $\mathbf{b}_k^*$ after being projected with respect to a new vector (the shortest vector of the local block inserted). We assume that the coefficients of the shortest vector in terms of the normalized Gram–Schmidt basis $(\mathbf{b}_1^*/\|\mathbf{b}_1^*\|, \mathbf{b}_2^*/\|\mathbf{b}_2^*\|, \cdots, \mathbf{b}_n^*/\|\mathbf{b}_n^*\|)$ looks like a uniformly distributed vector of the same norm. This twist also occurs in experiments: the updated second Gram–Schmidt norm is almost always a bit smaller than the old first Gram–Schmidt norm of the block. Such a strategy also makes the simulator more flexible. In the new simulator, it is not necessary to update all the remaining blocks with the value estimated by the Gaussian heuristic once we have an update: the simulator makes an update only when needed, i.e., when an improving Gram–Schmidt norm is sampled.

We also use two sets of boolean values $\{t_0^{(i)}\}_{i\le n}$ and $\{t_1^{(i)}\}_{i\le n}$, to record if there is a change of $\log \|\mathbf{b}_i^*\|$ in the last and the current tours, respectively. If we know there was no change at all in current local block during the last tour, we simply skip the current block and go to the next one. Correspondingly, in BKZ, it means that the found shortest vector in current block in this tour will be the same as the one in current block in last tour. As it was not used to make an update during the last tour, so will it not be used in this tour.

Algorithm 3 The probabilistic BKZ simulator

Input: The Gram–Schmidt log-norms $\{\ell_i = \log \|\mathbf{b}_i^*\|\}_{i \le n}$ and an integer $N \ge 1$.
Output: A prediction of the Gram–Schmidt log-norms after N tours of BKZ.

1: **for** $i = 1$ `to` 45 **do** $r_i \leftarrow \mathbb{E}[\log \|\mathbf{b}_k^*\| : \mathbf{B}$ HKZ-reduced basis of $\Lambda \leftarrow \Gamma_{45}]$
2: **end for**
3: $t_0^{(i)} \leftarrow$ `true`, $\forall i \le n$
4: **for** $j = 1$ `to` N **do**
5: $\quad t_1^{(i)} \leftarrow$ `false`, $\forall i \le n$
6: $\quad$ **for** $k = 1$ `to` $n - 45$ **do**
7: $\quad\quad d \leftarrow \min(\beta, n - k + 1)$; $e \leftarrow k + d$
8: $\quad\quad \tau \leftarrow$ `false`
9: $\quad\quad$ **for** $k' = k$ `to` e **do** $\tau \leftarrow \tau \| t_0^{(k')}$
10: $\quad\quad$ **end for**
11: $\quad\quad \log \mathrm{vol}(\Lambda_{[k,e]}) \leftarrow \sum_{i=1}^{e-1} \ell_i - \sum_{i=1}^{k-1} \widehat{\ell}_i$
12: $\quad\quad$ **if** $\tau =$ `true` **then**
13: $\quad\quad\quad X \hookleftarrow$ `Expo`$[1/2]$
14: $\quad\quad\quad g \leftarrow (\log X + \log \mathrm{vol}(\Lambda_{[k,e]}) - \log v_d)/d$
15: $\quad\quad\quad$ **if** $g < \ell_k$ **then**
16: $\quad\quad\quad\quad \widehat{\ell}_k = g$
17: $\quad\quad\quad\quad \widehat{\ell}_{k+1} \leftarrow \ell_k + \log(\sqrt{1 - 1/d})$
18: $\quad\quad\quad\quad \gamma \leftarrow (\ell_k + \ell_{k+1}) - (\widehat{\ell}_k + \widehat{\ell}_{k+1})$
19: $\quad\quad\quad\quad$ **for** $k' = k + 2$ `to` e **do**
20: $\quad\quad\quad\quad\quad \widehat{\ell}_{k'} \leftarrow \ell_{k'} + \gamma/(d-2)$
21: $\quad\quad\quad\quad\quad t_1^{(k')} \leftarrow$ `true`
22: $\quad\quad\quad\quad$ **end for**
23: $\quad\quad\quad\quad \tau \leftarrow$ `false`
24: $\quad\quad\quad$ **end if**
25: $\quad\quad$ **end if**
26: $\quad\quad \{\ell_k, \cdots, \ell_{e-1}\} \leftarrow \{\widehat{\ell}_k, \cdots, \widehat{\ell}_{e-1}\}$
27: $\quad$ **end for**
28: $\quad \log \mathrm{vol}(\Lambda_{[k,e]}) \leftarrow \sum_{i=1}^{n} \ell_i - \sum_{i=1}^{n-45} \widehat{\ell}_i$
29: $\quad$ **for** $k' = n - 44$ `to` n **do**
30: $\quad\quad \widehat{\ell}_{k'} \leftarrow \frac{\log \mathrm{vol}(\Lambda_{[k,e]})}{45} + r_{k'+45-n}$
31: $\quad\quad t_1^{(k')} \leftarrow$ `true`
32: $\quad$ **end for**
33: $\quad \{\ell_1, \cdots, \ell_n\} \leftarrow \{\widehat{\ell}_1, \cdots, \widehat{\ell}_n\}$
34: $\quad \{t_0^{(1)}, \cdots, t_0^{(n)}\} \leftarrow \{t_1^{(1)}, \cdots, t_1^{(n)}\}$
35: **end for**

As the Chen–Nguyen simulator is deterministic, it terminates relatively fast, within a few hundreds of tours typically. Oppositely, our probabilistic simulator may perform far more tours and continue making further (though smaller and smaller) Gram–Schmidt progress. Another difference between the behaviors of the simulators comes from the fact that the expectation of a given sample (g in `Line 14` of Algorithm 3) for updating each local block in our simulator is slightly larger than the one used in the Chen–Nguyen simulator that uses the Gaussian heuristic (but they are closer to each other as the block size increases). As a result, the sampled value for the first minimum of a local block can be slightly larger than in the Chen–Nguyen simulator. However (and more importantly), the chosen value can also be smaller than the Gaussian heuristic, and in fact smaller than the current value even if that one is already quite small. This is exactly what makes the Gram–Schmidt log-norms progress further in the simulations and makes the simulations closer to the practical behavior of BKZ.

4.2 Heuristic justification

We now give a heuristic explanation as to why the probabilistic simulator (and BKZ) keeps making progress even after some significant amount of time. Every time it considers a block, it keeps trying to find a shorter vector than the current first vector of the block, thanks to fresh random sampling. Let $X \hookleftarrow \mathrm{Expo}(1/2)$ and $Y = X^{1/n}$. Assume for simplicity that there is only one block (i.e., $n = \beta$) and that the lattice Λ has been scaled so that the volume of the lattice is 1, which implies that $\lambda_1(\Lambda)$ has the same distribution as Y. The CDF of Y is

$$F(y) = 1 - \mathrm{e}^{-y^n/2}.$$

Let $Y_{\min,K}$ be the minimum among K independent Y_i's. Its CDF and PDF are

$$F_{\min,K}(y) = 1 - \mathrm{e}^{-Ky^n/2} \quad \text{and} \quad f_{\min,K}(y) = Kny^{n-1}\mathrm{e}^{-Ky^n/2}/2,$$

respectively. We can hence compute the expected value

$$\mathbb{E}(Y_{K,min}) = (2/K)^{1/n} \cdot \Gamma(1 + 1/n) = \mathbb{E}(\lambda_1(\Lambda))/K^{1/n}.$$

One sees that the expectancy keeps decreasing, although much more slowly as K increases. Notice that K here can be regarded as proportional to the number of tours in our probabilistic simulator. We conjecture that BKZ is enjoying a similar phenomenon.

This simple model does not work for explaining BKZ with a single block (because for a single block, once the SVP instance has been solved, it cannot be improved further). In the more interesting case where $\beta < n$, the fact there are many intertwined blocks helps as an improvement for one block 'refreshes' the neighbouring blocks, which then have a chance to be improved. In this case, however, this simple model does not capture the impact of one block on the neighbouring blocks, nor the fact that the SVP instances across blocks are not statistically independent (in particular, the blocks overlap).

4.3 Quality of the new simulator

In this subsection, we describe experiments aiming to assess the accuracy of our probabilistic BKZ simulator. We measure the quality of our simulator by comparing with the practical BKZ and the Chen–Nguyen simulator using two quantities: the Gram–Schmidt log-norms after certain tours and the root Hermite factors. We then describe some limitations of our simulator.

(a) Graph of Gram–Schmidt log-norms. In practice, the full sequence of Gram–Schmidt log-norms is important for evaluating the quality of a basis. For example, if we extrapolate the log-norms by a straight line, the slope of the line can be used to indicate whether the basis is of good quality or not. For this reason, we are interested in how accurately the simulator predicts the evolution of the full sequence of Gram–Schmidt log-norms during the BKZ execution. We consider the following two experiments: (1) The input lattices are SVP-100

instances and we use BKZ_{45} without pruned enumeration up to $2,000$ tours. For this experiment, the setup is the same as the one used in Subsection 3.4. We plot the averaged Gram–Schmidt log-norms at tours 50 and $2,000$. (2) The input lattices are SVP-150 instances and we use BKZ_{60} with pruned enumeration up to $20,000$ tours. Note some experiments (and simulation) completes before $20,000$ tours. In such cases, we take the Gram–Schmidt log-norms of the basis obtained at completion. We plot the averaged Gram–Schmidt log-norms at tours 50 and $20,000$ (or the last Gram–Schmidt log-norms if it completes before $20,000$ tours). We record the full log-norm sequences at the end of selected tours. As shown in Figures 15–22, after a few tours, both the new BKZ simulator and the Chen–Nguyen simulator approach the experimental behavior of BKZ. As the number of BKZ tours increases, both the experimental BKZ and the probabilistic BKZ simulator evolve, and the corresponding log-norms eventually become concave in the head region. However, the Chen–Nguyen simulator stops making progress after a few tours. By comparison, the new simulator fits the experimental results quite accurately in both situations.

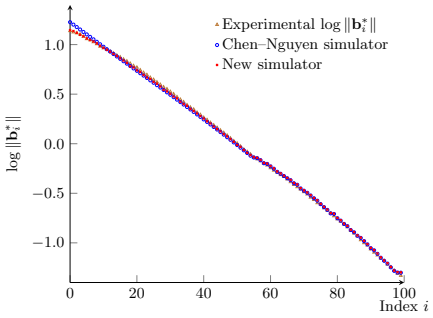

Fig. 15: Gram–Schmidt log-norms for BKZ_{45} at tour 50.

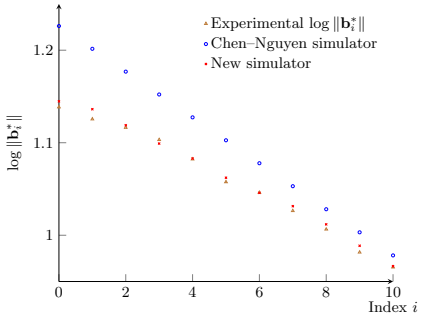

Fig. 16: Same as left hand side, but zoomed in.

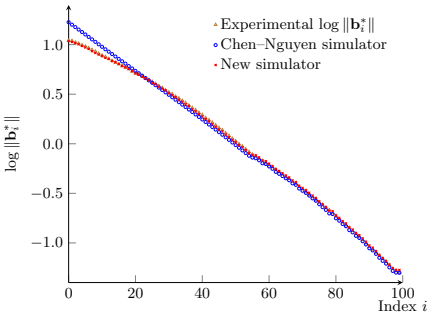

Fig. 17: Gram–Schmidt log-norms for BKZ_{45} at tour $2,000$.

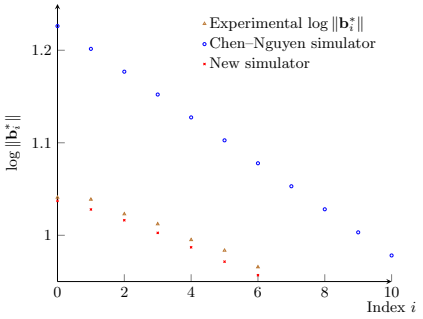

Fig. 18: Same as left hand side, but zoomed in.

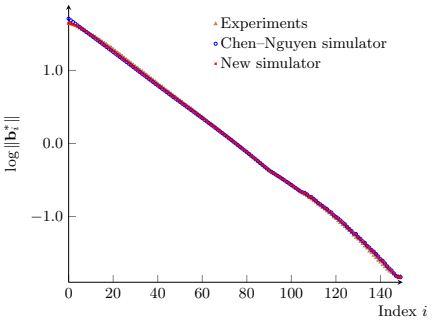

Fig. 19: Gram–Schmidt log-norms for BKZ_{60} at tour 50.

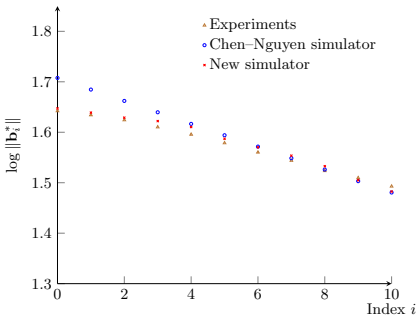

Fig. 20: Same as left hand side, but zoomed in.

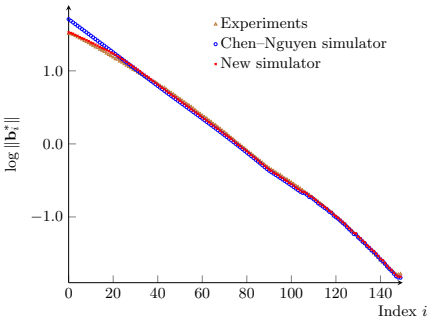

Fig. 21: Gram–Schmidt log-norms for BKZ_{60} at tour $20,000$.

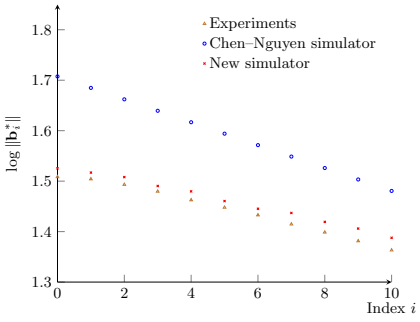

Fig. 22: Same as left hand side, but zoomed in.

(b) Root Hermite factor. In Subsection 3.4, we have seen that the asymptotes (for a large number of tours) of the root Hermite factors obtained with the genuine BKZ algorithm and the Chen–Nguyen simulator diverge. Here in Figures 23 and 24, we investigate the behavior of the root Hermite factor obtained with the new probabilistic simulator, when the number of tours increases. One can observe that, after a few tours, the probabilistic simulator predicts the experimental data more closely. One can also observe that, in the very first several tours, neither the Chen–Nguyen simulator nor the probabilistic simulator is very accurate. In the case of pruned enumeration, the experimental root Hermite factors seem to drop faster than in the simulators; while with non-pruned enumeration, the root Hermite factors in experiments evolve more slowly. This may be due to the algorithmic and implementation complications brought by the pre-processing, the SVP solver, pruning and post-processing.

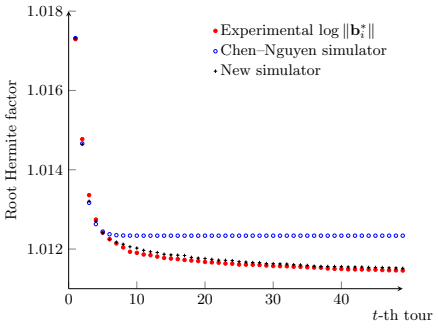

Fig. 23: Evolution of the root Hermite factor during the execution of BKZ_{45} (no pruned enumeration) on SVP-100.

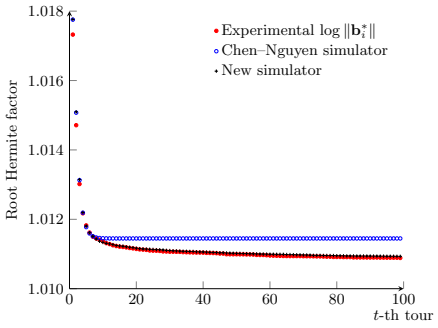

Fig. 24: Evolution of the root Hermite factor during the execution of BKZ_{60} (with pruned enumeration) on SVP-150.

(c) Limitations of the new simulator. The probabilistic simulator does not fully match with the experimental behavior of BKZ in the first few tours. In particular, the progress of the real Gram–Schmidt log-norms with BKZ with non-pruned enumeration is slower than the simulator's (refer to Figure 23). One potential reason is the local SVP solver only attempts to find a vector slightly smaller than $\|\mathbf{b}_i^*\|$ (instead of the Gaussian heuristic value). On the other hand, it may be observed that the progress of the real Gram–Schmidt log-norms is a bit faster than the simulated log-norms in the very first BKZ tours, for BKZ with pruned enumeration (refer to Figure 24) in these experiments. One potential reason could be that the pruned enumeration uses extensive pre-processing in a local block (which is not captured by the simulator), and this helps to lower the root Hermite factor in the beginning of the execution. However, as soon as the number of tours increases, the probabilistic phenomenon seems to weigh more and the new simulator becomes accurate.

Next, we verify if the preprocessing indeed helps make the experimental root Hermite factor decrease faster than the corresponding quality in the simulator. To verify the impact of preprocessing in pruned enumeration, we run BKZ_{60} (with pruned enumeration) without pre-processing on SVP_{150} instances. We plot the root Hermite factor of the basis after each tours (up to 100 tours). Each data is taken averaged over 100 results. As shown in Figure 25, the BKZ variant without pre-processing makes progress no faster than the simulator. This suggests that pre-processing indeed accelerates the progress made by BKZ. On the other hand, we can also observe that without preprocessing, the root Hermite factor decrease slower than the corresponding quality in the simulator. One possible reason for this could be that the vector found by extreme pruning in each local block may be longer than the minimum of the local lattice.

In conclusion, it seems quite difficult to use the simulator to estimate the precise evolution of Gram–Schmidt log-norms for the first few tours due to the following two reasons: (1) we are not clear about how much improvement is

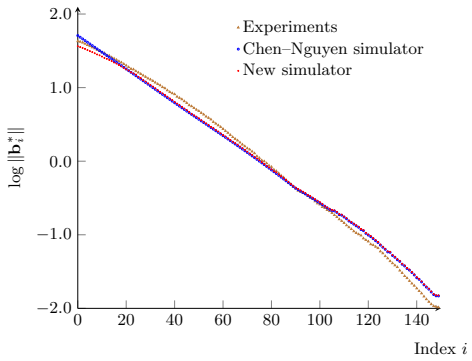

Fig. 25: Comparison of Gram–Schmidt log-norms obtained by the simulators and BKZ_{60} (no pre-processing) on SVP-150, after 4,000 tours.

provided by the pre-processing; (2) we do not have a precise understanding on the distribution of norms of vectors output by the enumeration (ideally with pruning). On other other hand, after the first few tours, the simulator seems to be more accurate when estimating the Hermite factors, which is important for cryptographic applications.

4.4 Predicting the root Hermite factor for large block sizes

As our proposed simulator predicts real BKZ quite well for the range of block sizes for which such experiments can be run, we expect that our simulator keeps this accuracy for larger block sizes. This is in particular relevant in cryptanalysis and for security analyses of concrete lattice-based cryptosystems. Indeed, many of the existing security analyses rely on the root Hermite factor predicted by the Chen–Nguyen simulator (see [ACD+18] and the references therein), which, as we have seen, is an over-estimate. We thus run the simulators for large block sizes and large dimensions, to assess how the discrepancy scales.

In this experiment, we consider two cases:

(1) the dimension n is much larger than the block-size β.
(2) the dimension n is a small constant times larger than the block-size β.

The Case (1) is a scenario often considered to assess the quality of BKZ-type algorithms (see, e.g., [CN11]). On the other hand, in practice, we are also interested in Case (2) where the dimension/block-size ratio is small. This is a typical situation for the lattice-based NIST candidates (see [ACD+18]). Concretely, in the first case, we run our simulator of BKZ with block-size $\beta \in [50, 250]$ on 1000-dimensional lattices and our simulator on BKZ with block-size $\beta \in [260, 300]$ in 2000-dimensional lattices, both with 20,000 tours. In the second case, we run the same experiment as above except with a fixed ratio of 3 between dimension and block-size. Each data point is averaged over 10 samples. We plot the root

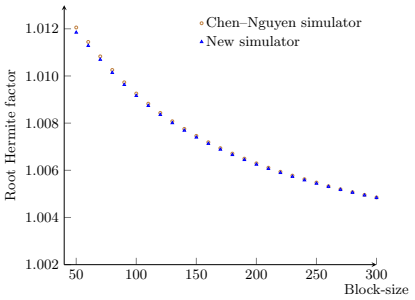

Fig. 26: Root Hermite factor for selected $\beta \in \{50, 60, \cdots, 300\}$. Here the dimension is 1000 for $\beta \in [50, 250]$ and 2000 for $\beta \in [260, 300]$.

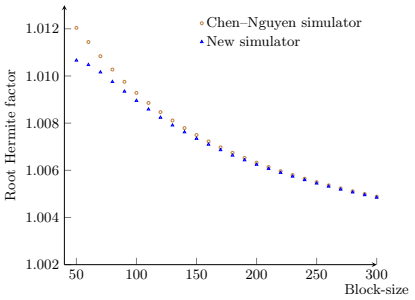

Fig. 27: Root Hermite factor for selected $\beta \in \{50, 60, \cdots, 300\}$. Here the dimension is $3 \cdot \beta$.

Hermite factor corresponding to the Gram–Schmidt log-norms output by our simulator.

As can be seen in Figures 26–27, for large block sizes, the discrepancy vanishes: both simulators converge to the same root Hermite factors. This may be explained by considering the distribution of the minimum of a uniform unit-volume lattice, used in the probabilistic simulator. The expectation is $2^{1/\beta} \cdot \Gamma(1 + 1/\beta)$, which converges to 1, the Gaussian heuristic value (when β grows to infinity). Further, as we have seen in Subsection 2.1, the variance of the selected value is $2^{2/\beta} \cdot \left(\Gamma\left(1 + 2/\beta\right) - \left(\Gamma\left(1 + 1/\beta\right)\right)^2\right) \cdot v_\beta^{-2/\beta}$, which decreases to 0 as $O(1/\beta^2)$, making the distribution "more concentrated" and lowering the chance of being "lucky" in finding unexpectedly short vectors in local lattices.

5 Pressing the concavity

In this section, we propose a new BKZ variant. For practical purposes, we further twist this new algorithm with several different strategies. We also quantify the quality of the obtained lattice bases.

5.1 Pressed-BKZ

Below, we first describe the new BKZ variant, pressed-BKZ, and then explain why it provides an improvement. Pressed-BKZ is described as Algorithm 4.

The pressed-BKZ algorithm runs standard BKZ on block $\Lambda_{[s,n]}$ with an incrementally increased starting index $s \in [1, n - \beta + 1]$. In particular, in the case of $s = 1$, pressed-BKZ executes standard BKZ. Note that in `Line 12`, "no change occurs" means that no local block was updated during the last tour (from $k = s$ to $k = n - 1$). The difference between pressed-BKZ and standard BKZ starts with $s > 1$. At that stage, it does not run BKZ on the full lattice basis anymore. Instead, it freezes the first $s - 1$ lattice vectors $\{\mathbf{b}_i\}_{i \in [1, s-1]}$ and re-randomizes

Algorithm 4 The pressed-BKZ algorithm

Input: A basis $\mathbf{B} = \{\mathbf{b}_1, \cdots, \mathbf{b}_n\}$, a block-size $\beta \geq 2$ and a constant $\delta < 1$.
Output: A basis of $\Lambda(\mathbf{B})$.

for $s = 1$ `to` $n - \beta + 1$ **do** // progressive starting point
  Re-randomize the projected lattice $\Lambda_{[s,n]}$.
  repeat
    for $k = s$ `to` $n - 1$ **do**
      `Find b such that` $\|\pi_k(\mathbf{b})\| = \lambda_1(\Lambda_{[k,\min(k+\beta-1,n)]})$
      if $\delta \cdot \|\mathbf{b}_k^*\| > \|\mathbf{b}\|$ **then**
        LLL-reduce$(\mathbf{b}_1, \cdots, \mathbf{b}_{k-1}, \mathbf{b}, \mathbf{b}_k, \cdots, \mathbf{b}_{\min(k+\beta,n)})$.
      else
        LLL-reduce$(\mathbf{b}_1, \cdots, \mathbf{b}_{\min(k+\beta,n)})$.
      end if
    end for
  until no change occurs (or other condition).
end for

the projected lattice $\Lambda_{[s,n]}$, then runs stardard BKZ on the projected lattice. Note that the re-randomization is necessary, otherwise after the BKZ reduction on $\Lambda_{[1,n]}$, no improvement will happen in BKZ reduction on $\Lambda_{[2,n]}$ in the second iteration and either in the following iterations. In particular, in the second iteration, the re-randomization helps randomize the basis vectors of the projected lattice $\Lambda_{[2,n]}$, thus gives a chance of generating a denser leading block of $\Lambda_{[2,n]}$ via BKZ reduction. The re-randomization on the projected lattice is done via tranforming the basis of the projected lattice with a unimodular matrix. Here, we use the unimodular matrix generated in the `fplll` library.

The design rationale is as follows. Suppose BKZ creates a head concavity for the Gram–Schmidt log-norms. Then the first iteration with $s = 1$ will help to lower $\log \|\mathbf{b}_1^*\|$. The iteration with $s = 2$ will preserve $\log \|\mathbf{b}_1^*\|$ and help to lower $\log \|\mathbf{b}_2^*\|$, etc. This explains the name of the algorithm.

5.2 On the behavior of pressed-BKZ

The goal of pressed-BKZ is to further improve the quality of bases obtained by the original BKZ algorithm without a block-size increase. In order to illustrate the idea, we run standard BKZ_{60} on 120-dimensional random lattices (generated in the same way as mentioned at the start of Section 4) with 500 tours (i.e., with early-abort) first, and then run pressed-BKZ with the same number of tours in each iteration with start index $s = 2$. Each data point is averaged over 100 samples. As shown in Figure 28, pressed-BKZ successfully presses the "head concavity" that was produced by the standard BKZ algorithm.

From the experiment above, we can already see that pressed-BKZ produces a basis with better quality, as its corresponding Gram–Schmidt log-norms achieve a smaller slope. Next, we try to assess by how much pressed-BKZ improves standard BKZ in this respect. We first adapt the simulator for BKZ from Section 4 in the direct way to simulate pressed-BKZ. Before we go further, we check the accuracy of our simulator when simulating the behavior of pressed-BKZ by running the same experiment above, but with the simulator. As shown in Figure 29,

our simulator produces a result that is close to the one experimentally obtain with pressed-BKZ.

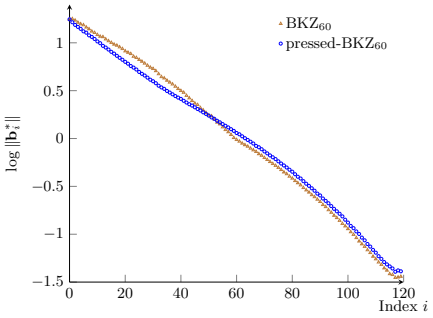

Fig. 28: Full sequences of Gram–Schmidt log-norms of bases returned by BKZ_{60} and pressed-BKZ_{60}.

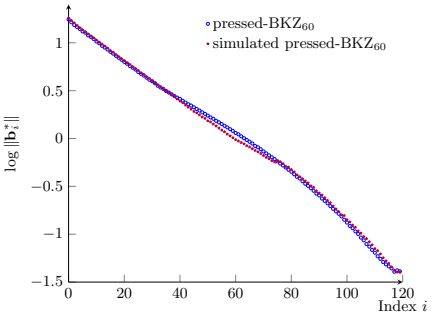

Fig. 29: Full sequences of Gram–Schmidt log-norms of bases returned by BKZ_{60}, pressed-BKZ_{60} and simulated pressed-BKZ_{60}.

Now we have an accurate simulator for pressed-BKZ, we can proceed to check the behavior of pressed-BKZ further. For this, we run our simulator of pressed-BKZ for block-sizes between 50 and 300, with many tours. We again consider two cases: (1) the dimension n is much larger than the block-size β; (2) the dimension n is a small constant times larger than the block-size β. In the first case, we simulate pressed-BKZ with block-size $\beta \in [50, 250]$ on 1,000-dimensional lattices with 5,000 tours for each iteration, and pressed-BKZ with block-size $\beta \in [260, 300]$ on 2,000-dimensional lattices with 10,000 tours. Each data is averaged over 10 samples. In the second case, we run the same experiment as above except with the dimension/block-size ratio set to 3. Further, we recall the Chen–Nguyen simulator for a comparison. Note that we can also adapt the Chen–Nguyen simulator for pressed-BKZ, which however, gives a same result as the simulation for standard BKZ. Here, we use the extrapolated slope to evaluate the quality of a reduced basis. To compute the slope, we fit the Gram–Schmidt log-norms with a line using the least square method of `fplll` and `fpylll`. Note that the default implementation in `fplll` and `fpylll` computes the slope using the Gram–Schmidt log-norms multiplied by 2. Here we compute the slope using the Gram–Schmidt log-norms only. As we can see in Figure 30, there is a difference between our simulator for pressed-BKZ and the Chen–Nguyen simulator (for standard BKZ), which means our simulator for pressed-BKZ may be used to make a severer cryptanalysis on lattice-based cryptography compared to the Chen–Nguyen simulator.

As can be seen in Figure 31, when the dimension is relatively close to the block-size, our simulator for pressed-BKZ outputs the Gram–Schmidt norms with slope more significantlly better than the one output by the Chen–Nguyen simulator. In particular, for small block-size $\beta = 50$, our simulator for pressed-BKZ can produce Gram–Schmidt norms with slope almost equal to the one

produced by the Chen–Nguyen simulator for standard BKZ with block-size 85. Thus we earn almost 35 dimensions while only relying on an SVP solver in dimension 50. The difference becomes very small when the block-size considered is larger than 200.

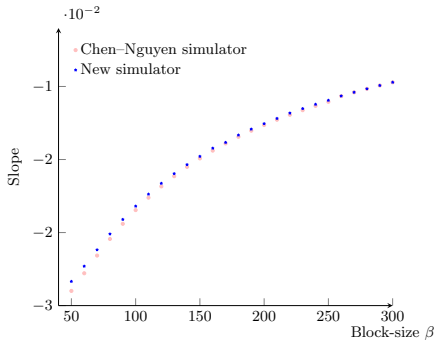

Fig. 30: Comparison of the slopes of Gram–Schmidt log-norms between our simulator for pressed-BKZ and the Chen–Nguyen simulator for standard BKZ for selected $\beta \in \{50, 60, \cdots, 300\}$. Here the dimension is 1000 for $\beta \in [50, 250]$ and 2000 for $\beta \in [260, 300]$.

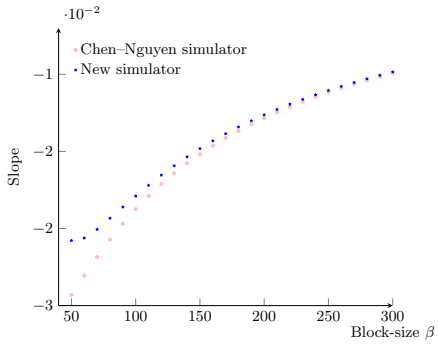

Fig. 31: Comparison of the slopes of Gram–Schmidt log-norms between our simulator for pressed-BKZ and the Chen–Nguyen simulator for standard BKZ for selected $\beta \in \{50, 60, \cdots, 300\}$. Here the dimension is $3 \cdot \beta$.

5.3 Variable block-size strategy

Pressed-BKZ helps improve the quality of the basis such that its Gram–Schmidt log-norms approximate a line. However, the concavity phenomenon may still exist within each iteration during pressed-BKZ. Concretely, in the BKZ_β reduction for each projected sub-lattice $\Lambda_{[s,n]}$ for $s \in [1, \cdots, n-\beta+1]$, one can still observe the head concavity phenomenon after a few tours of the BKZ algorithm. As a result, the costs of solving SVP instances in the leading blocks become less than those for the middle blocks (we refer to Subsection 3.2 for the correspondence between quality of basis and enumeration cost for SVP).

We adapt the variable block-size strategy from [AWHT16]. The principle of the variable block-size strategy is to adaptively use larger block-sizes for the leading blocks, so that their enumeration costs match the enumeration costs for the middle blocks. We use the following simple variant: for the case of BKZ_β on the projected $(n-s+1)$-dimensional sub-lattice, we always take the specific block $\Lambda_{[k,e]}$ in the middle with $k = \lfloor n/2 \rfloor - \lfloor \beta/2 \rfloor + \lfloor s/2 \rfloor + 1$ and $e = \lfloor n/2 \rfloor + \lfloor \beta/2 \rfloor + \lfloor s/2 \rfloor$ as the standard SVP cost for comparison. When the estimated SVP cost for any leading block is smaller than the cost of this middle block, we

progressively increase the block-size of the current leading block until its SVP cost matches the standard SVP cost. Correspondingly, we only use the variable block-size strategy for those leading blocks, with starting index not exceeding k (the starting index of the selected middle block).

Note that if such variable block-size strategy improves standard BKZ, then it is likely to improve pressed-BKZ: if the variable block-size strategy can decrease the first Gram–Schmidt norm of the projected lattice $\Lambda_{[k,n]}$ (for k from 1 to $n - \beta + 1$) a little more (compared to standard BKZ without such a strategy), then the improvement from each iteration will eventually contribute to the final pressed-BKZ reduced basis. Thus we will only consider such variable block-size strategy with standard BKZ. As our simulator seems to be precise on the quality of BKZ, we first compare BKZ with and without such a variable block-size strategy using our simulator.

We run the simulation (100 instances) for BKZ_{60} (with and without the variable block-size strategy) and plot the average root Hermite factor after each tour. As shown in Figures 32 and 33, BKZ with variable block-size makes the root Hermite factor decrease slightly faster. It seems that the difference becomes smaller when the number of tours increases. However, we also notice that after sufficiently many tours, such a difference reoccurs as shown in Figure 33. One can observe that the largest gap in Figure 33 is less than 0.0001. Thus we may conclude that the variable block-size strategy helps improve the root Hermite factor faster. However, it does not seem to give a significant improvement on the root Hermite factor itself.

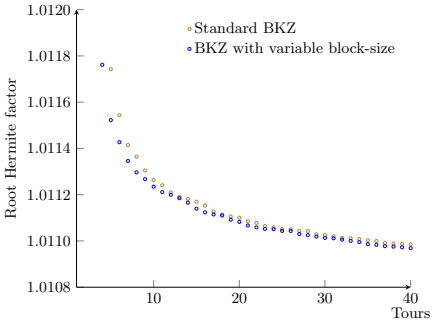

Fig. 32: Comparison of root Hermite factors of simulated BKZ with and without variable block-size. The simulation is performed with our new simulator up to 40 tours.

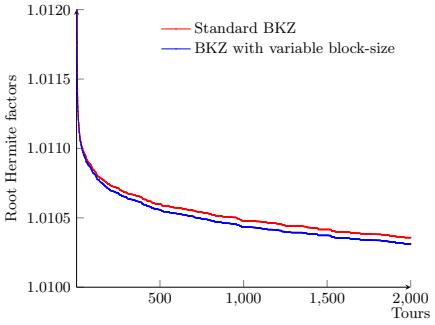

Fig. 33: Comparison of root Hermite factors of simulated BKZ with and without variable block-size. The simulation is performed with our new simulator up to 2,000 tours.

Next, we run experiment to verify if the variable block-size strategy indeed helps decreasing the root Hermite factor faster (and check if this matches the simulated results). We run standard BKZ_{60} with and without variable block-size on an SVP-120 instance from the Darmstadt lattice challenge. We plot the average root Hermite factor (over 100 samples) after each tour. As can be seen in

Figure 34 and 35, the convergence of root Hermite factor of basis output by the standard BKZ_{60} with variable block-size is slightly better than the one output by BKZ_{60} without such strategy.

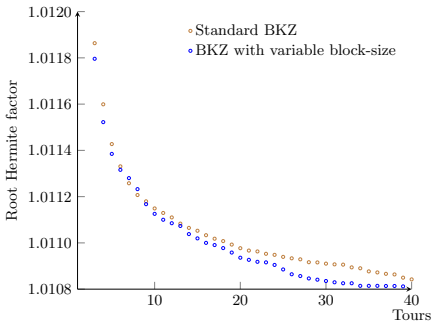

Fig. 34: Comparison of root Hermite factors of standard BKZ_{60} with and without variable block-size within 40 tours.

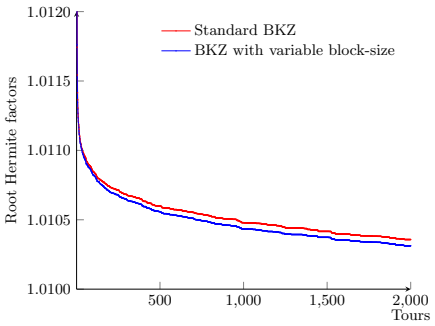

Fig. 35: Comparison of root Hermite factors of standard BKZ_{60} with and without variable block-size within 2,000 tours.

5.4 Solving SVP-120 with pressed-BKZ

In this subsection, we use pressed-BKZ for the preprocessing phase to solve an SVP-120 challenge, to demonstrate its practical relevance. We are interested in the quality of pressed-BKZ-reduced bases as reflected by the total enumeration cost, i.e., the sum the preprocessing and enumeration costs, divided by the success probability. In the experiment, we consider an SVP challenge of dimension 120 (generated using the Darmstadt lattice challenge generator) and preprocess it using pressed-BKZ_{60} with the adaptive block-size strategy described in the previous subsection. The preprocessing took a total 5×10^5 seconds on an Intel Xeon processor of 2.67GHz (the enumeration speed is 2×10^7 nodes per second and hence the runtime corresponds to an enumeration tree of 1×10^{13} nodes).

Comparison with standard BKZ. We first investigate the quality of the pressed-BKZ_{60}-reduced basis in terms of BKZ-reducedness. The aim is to find the block-sizes β (and the number of tours) for which the output of standard BKZ_β would be of similar quality. This suggests that the bases produced by pressed-BKZ_{60} and standard BKZ_β have similar full enumeration cost.

We have to determine some criterion for the quality of bases. Essentially, one wants to compare the *pruned* enumeration cost of the pressed-BKZ_{60}-reduced basis with the pruned enumeration cost of the standard BKZ_β-reduced basis. As a first approximation, we compute the *full* enumeration cost of BKZ_β-reduced basis (right after each BKZ tour) and stop as soon as it is close to the *full* enumeration cost for the pressed-BKZ-reduced basis. This also gives us roughly the number of tours needed for BKZ with the corresponding block-size to achieve

a similar quality as our basis reduced by pressed-BKZ$_{60}$. A better approach, which we did not implement, would be to invoke the pruning optimizer right after each local SVP to estimate the enumeration cost and stop as soon as it is close to the pruned enumeration cost for the pressed-BKZ reduced basis.

Instead of doing the actual BKZ$_\beta$ experiment for all candidate blocksizes, we first use simulation to find the most competitive blocksizes. As investigated in Section 4, the probabilistic simulator seems quite precise after the first few tours: if the number of tours involved in the simulation is tiny, we conduct true BKZ experiments for confirmation. After we have determined the most appropriate blocksizes β, we conduct true BKZ experiments for these blocksizes to double-check their quality and run-time (as opposed to simulation).

To start with, we have to determine some suitable searching range for the block-size β. As we already saw, in the case where the dimension is not much larger than the block-size, the quality of a basis reduced by pressed-BKZ$_{60}$ can be quite superior to that obtained by using BKZ$_\beta$ for $\beta > 60$. Thus we try several larger blocksizes starting for $\beta = 70, 75, 80, 85, 90$. For each blocksize β, the Gram–Schmidt norms of standard BKZ$_\beta$ are simulated by the probabilistic simulator. We start with the LLL-reduced basis of the SVP-120 input and average over the 100 simulations for each β. We set a maximum of $50,000$ tours in the simulator but break as soon as (if possible) the full enumeration cost is smaller than the full enumeration cost of our reduced pressed-BKZ$_{60}$.

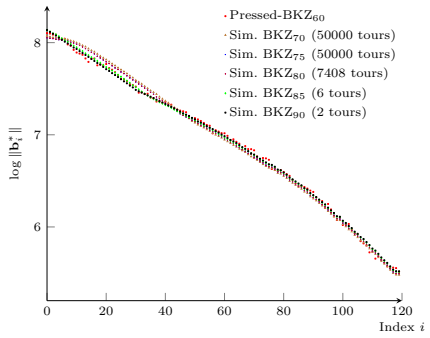

Fig. 36: Gram–Schmidt log-norms of simulated pressed-BKZ$_{60}$ and simulated BKZ$_\beta$.

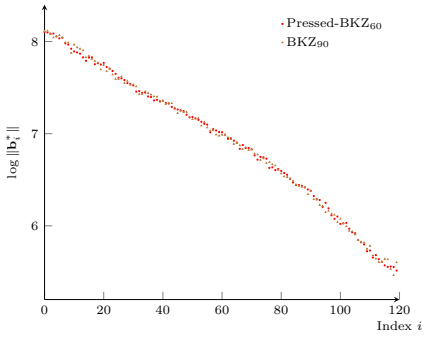

Fig. 37: Gram–Schmidt log-norms of experimental pressed-BKZ$_{60}$ and BKZ$_{90}$ (28 tours).

For blocksizes 70 and 75, the full enumeration cost cannot beat the full enumeration cost of pressed-BKZ$_{60}$ reduced basis within the limit of $50,000$ tours and therefore terminates. For other blocksizes, after the simulator terminates, we compute the (average) minimum number of tours needed. They are listed in the legend of Figure 36. In Figure 36, we also plot the Gram–Schmidt log-norms of the pressed-BKZ$_{60}$-reduced basis along with the average simulated Gram–Schmidt log-norms of the output bases of standard BKZ$_\beta$ (for the relevant numbers of tours). One can observe that, at the point of termination, the simulated

Gram–Schmidt log-norms have comparable shape as the pressed-BKZ_{60} reduced basis. We confirm this by examining their full enumeration cost in Table 1. The full enumeration cost is tabulated for each (averaged) simulated basis. As a conclusion, it can be seen that the most competitive blocksizes are 85 and 90: the number of tours involved for blocksizes $70, 75, 80$ are too large.

So far, we have only judged the quality of the preprocessed basis using simulation. It should be noted that the probabilistic simulator may not to be accurate when the number of tours involved are tiny (see Subsection 4.3). This is the case for blocksizes 85 and 90. Therefore, instead of simulation, we conduct actual BKZ_{85} and BKZ_{90} experiments with the LLL-reduced basis as input: we used a parallel implementation of the BKZ algorithm implemented in `fpylll` and ran BKZ_{90} on the LLL-reduced basis with 280 cores. The local SVP solver attempts to find a vector smaller than 1.05 times the Gaussian heuristic of the local block. Also, if too many trials have been attempted without a success, then it moves to the next block. Therefore, the number of tours in experiments could be larger than in simulations (but each local SVP takes less time).

Table 1: Estimated enumeration cost to solve the SVP-120 instance. The row "Full of Sim." records the full enumeration cost (number of nodes) based on the simulated preprocessed basis. The row "Full of Exp." records the full enumeration cost on BKZ_{85} and BKZ_{90}-reduced bases (from experiments) after 100 tours and 28 tours respectively. The row "Prune of Exp." records the cost for pruned enumeration for Pressed-BKZ_{60} and BKZ_{90} reduced basis (from experiments): it includes the costs of all trial enumerations and the cost of preprocessing before each trial enumeration (but excludes the initial preprocessing cost).

	Pressed-BKZ_{60}	BKZ_{70}	BKZ_{75}	BKZ_{80}	BKZ_{85}	BKZ_{90}
Full of Sim.	n/a	6.83×10^{27}	3.21×10^{27}	1.17×10^{27}	1.15×10^{27}	0.83×10^{27}
Full of Exp.	1.21×10^{27}	n/a	n/a	n/a	2.64×10^{27}	1.35×10^{27}
Prune of Exp.	5.9×10^{13}	n/a	n/a	n/a	n/a	6.3×10^{13}

In the experiments, we aborted the BKZ_{90} execution right after the full enumeration cost of the current basis is similar to (if possible) that of the pressed-BKZ_{60}-reduced basis. Then we used the previous BKZ tour (where the full enumeration cost was slightly larger than that of the pressed-BKZ_{60}-reduced basis). BKZ_{90} took 28 tours to reach a similar full enumeration cost and the overall run-time was 5×10^6 seconds (the number of cores is taken into account). In Figure 37, we plot the Gram–Schmidt log-norms of this BKZ_{90}-reduced basis and compare it with pressed-BKZ_{60}. This confirms that their qualities are analogous. For BKZ_{85}, we aborted the computation after 100 tours as the the overall run-time was already 8×10^6 seconds. Note that both are already much larger than the cost we spent on the pressed-BKZ_{60} preprocessing, of 5×10^5 seconds. The full enumeration costs of BKZ_{90} and BKZ_{85}-reduced bases is computed in

Table 1. Note that the experiments for BKZ_{90} (and BKZ_{85}) took much more tours to achieve the same quality (if possible) compared to simulation. This might be due to the facts that our implementation is greedy as mentioned above and does not always solve the local SVP problem.

So far, we have only used the full enumeration cost to measure the quality. We confirm this using a pruner to estimate the enumeration cost (for Pressed-BKZ_{60} and BKZ_{90}-reduced bases). A pruner optimizes the pruning coefficients to minimize the overall run-time of preprocessing plus enumeration divided by the success probability. The general strategy in extreme pruning [CN11] is to preprocess the basis using BKZ and then run the enumeration with a certain success probability p. If the enumeration fails, it rerandomizes the basis and then conducts the preprocessing and enumeration again. The expected number of repetitions to succeed in the enumeration is $\approx 1/p$. It remains to determine the preprocessing time before each enumeration. It should be noted if the first enumeration fails, one usually runs a mild re-randomization before the next preprocessing, thus the next preprocessing will be faster than the first preprocessing, since it still benefits from the BKZ reduction in last preprocessing.

We determine the preprocessing time with the following experiment. For the BKZ_{90}-reduced basis, after re-randomization, the full enumeration cost increases from 1.35×10^{27} to 1.56×10^{27}. We re-preprocess the randomized basis using BKZ_{80} until the full enumeration cost decreases to around 1.35×10^{27}. Here we just used BKZ_{80} for simplicity (there could be other strategies). The re-preprocessing took 1.7×10^5 seconds (i.e., 3.4×10^{12} nodes). We use this preprocessing cost (the preprocessing before each trial enumeration except the initial preprocessing) as input to the pruner (for both Pressed-BKZ_{60} and BKZ_{90}-reduced bases). The total pruned enumeration cost estimate in `fpylll`, tabulated in Table 1, confirms that Pressed-BKZ_{60} and BKZ_{90}-reduced bases indeed have similar quality as they all admit similar total pruned enumeration costs. In general, the pruner seems to be quite precise in practice (hence so are the estimates in Table 1). Thus it suffices to compare the initial preprocessing cost between Pressed-BKZ_{60} and BKZ_{90}: pressed-BKZ_{60} (5×10^5 seconds) took less time compared to BKZ_{90} (28 tours in 5×10^6 seconds).

In this subsection, we have only considered a straightforward strategy, BKZ plus enumeration, for solving the SVP-120 instance. In the following we will further compare with progressive-BKZ[AWHT16].

Comparison with progressive-BKZ. The main idea of progressive-BKZ is to preprocess the basis using BKZ with progressively increased blocksizes and use local enumeration with high success probability to avoid the overheads brought in by the preprocessing. Furthermore, a progressive blocksize strategy (optimized based on their adaptation of the Chen–Nguyen simulator for their progressive-BKZ algorithm) was used for preprocessing before a final enumeration. In particular, [AWHT16, Table 4] gives the cost of solving SVP challenges using their blocksize strategy: this table is to be understood as the cost of an idealized algorithm and is hence optimistic compared to current algorithms. We re-investigate

the estimates in that table by combining their progressive-BKZ method with our pressed-BKZ algorithm.

Given the pressed-BKZ$_{60}$ reduced basis, we use the progressive-BKZ method in the `bkz2_sweet_spot`[4] branch in `fplll`. Note that it implements a variant of progressive-BKZ: the progressive strategy differs from that of [AWHT16] but it suffices for our comparison. It should be noted that our pressed-BKZ$_{60}$ reduced basis is already quite reduced so we start progressive-BKZ with blocksize ≈ 75 to avoid a superfluous re-computation. We used 80 cores for the computation on the pressed-BKZ$_{60}$ reduced basis. We spent 5.78 core days (5×10^5 seconds) on the initial pressed-BKZ$_{60}$ and 1.21 core days for the progressive-BKZ to complete the SVP instance. In total, we completed the computation in a total of 6.99 core days (with enumeration speed of $\approx 2 \times 10^7$ nodes per second), faster than the 14.94 lower bound (with enumeration speed of 6×10^7 nodes per second) in [AWHT16, Table 4].

For further comparison, we also ran the same experiment using an LLL-reduced basis instead of pressed-BKZ$_{60}$ reduced basis in the beginning. The overall run-time was 8.75 core days. This implies that `bkz2_sweet_spot` is faster than the estimates in [AWHT16]. Compared to this LLL-based experiment, the pressed-BKZ$_{60}$-reduced basis helps to reduce the overall run-time by about 20%.

It should be noted that we only provide one such strategy that lowers the estimates in [AWHT16, Table 4] and demonstrate the usefulness of the pressed-BKZ algorithm. It is quite possible that this is far from an optimal strategy, which could combine variants of progressive preprocessing, extreme pruning and adaptive choices based on simulation. For instance, it may be better to also use pressed-BKZ inside progressive-BKZ (recursively for any preprocessing) to better maintain the shape of the pressed-BKZ preprocessed basis. Also, we only conducted two SVP-120 experiments, which is not statistically significant. We leave the question of how to optimize the strategy based on the existing approaches open for future work.

Acknowledgments. We thank the reviewers for detailed comments and suggestions. We acknowledge the Research Computing at Florida Atlantic University and the PSMN (Pôle Scientifique de Modélisation Numérique) of ENS Lyon for providing computing facilities. This work has been supported in part by ERC Starting Grant ERC-2013-StG-335086-LATTAC, by the European Union PROMETHEUS project (Horizon 2020 Research and Innovation Program, grant 780701) and by BPI-France in the context of the national project RISQ (P141580).

References

ACD+18. M. R. Albrecht, B. R. Curtis, A. Deo, A. Davidson, R. Player, E. W. Postlethwaite, F. Virdia, and T. Wunderer. Estimate all the {LWE, NTRU} schemes! In *SCN*, pages 351–357, 2018.

APS15. M. R. Albrecht, R. Player, and S. Scott. On the concrete hardness of learning with errors. *J. Mathematical Cryptology*, 9(3):169–203, 2015.

[4] `https://github.com/fplll/fpylll/tree/bkz2_sweet_spot`

AWHT16. Y. Aono, Y. Wang, T. Hayashi, and T. Takagi. Improved progressive BKZ algorithms and their precise cost estimation by sharp simulator. In *EUROCRYPT*, pages 789–819, 2016.

BDGL16. A. Becker, L. Ducas, N. Gama, and T. Laarhoven. New directions in nearest neighbor searching with applications to lattice sieving. In *SODA*, pages 10–24, 2016.

Che09. Y. Chen. *Réduction de réseau et sécurité concrète du chiffrement complètement homomorphe*. PhD thesis, Université Paris Diderot, 2009.

CN11. Y. Chen and P. Q. Nguyen. BKZ 2.0: Better lattice security estimates. In *ASIACRYPT*, pages 1–20, 2011.

dt16. The FPLLL development team. fplll, a lattice reduction library. `https://github.com/fplll/fplll`, 2016.

dt17. The FPYLLL development team. fpylll, a Python interface to fplll. `https://github.com/fplll/fpylll`, 2017.

Duc18. L. Ducas. Shortest vector from lattice sieving: A few dimensions for free. In *EUROCRYPT*, pages 125–145, 2018.

FP83. U. Fincke and M. Pohst. A procedure for determining algebraic integers of given norm. In *EUROCAL*, pages 194–202, 1983.

GNR10. N. Gama, P. Q. Nguyen, and O. Regev. Lattice enumeration using extreme pruning. In *EUROCRYPT*, pages 257–278, 2010.

HPS11. G. Hanrot, X. Pujol, and D. Stehlé. Analyzing blockwise lattice algorithms using dynamical systems. In *CRYPTO*, pages 447–464, 2011.

HS07. G. Hanrot and D. Stehlé. Improved analysis of Kannan's shortest lattice vector algorithm. In *CRYPTO*, pages 170–186. Springer, 2007.

HS08. G. Hanrot and D. Stehlé. Worst-case Hermite-Korkine-Zolotarev reduced lattice bases. *CoRR*, abs/0801.3331, 2008.

Kan83. R. Kannan. Improved algorithms for integer programming and related lattice problems. In *STOC*, pages 99–108. ACM, 1983.

MV10. D. Micciancio and P. Voulgaris. Faster exponential time algorithms for the shortest vector problem. In *SODA*. ACM, 2010.

Neu17. A. Neumaier. Bounding basis reduction properties. *Des. Codes Cryptography*, 84(1-2):237–259, 2017.

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

NV09. P. Q. Nguyen and B. Vallée. *The LLL Algorithm: Survey and Applications*. Information Security and Cryptography. Springer, 2009.

Rog56. C. A. Rogers. The number of lattice points in a set. *Proc. London Math. Soc.*, 3(6):305–320, 1956.

Sch59. W. Schmidt. Masstheorie in der Geometrie der Zahlen. *Acta Math.*, 102(3-4):159–224, 1959.

Sch87. C.-P. Schnorr. A hierarchy of polynomial lattice basis reduction algorithms. *Theor. Comput. Science*, 53:201–224, 1987.

Sch03. C.-P. Schnorr. Lattice reduction by random sampling and birthday methods. In *STACS*, pages 145–156. Springer, 2003.

SE94. C.-P. Schnorr and M. Euchner. Lattice basis reduction: improved practical algorithms and solving subset sum problems. *Mathematics of Programming*, 66:181–199, 1994.

Söd11. A. Södergren. On the Poisson distribution of lengths of lattice vectors in a random lattice. *Mathematische Zeitschrift*, 269(3):945–954, 2011.

YD17. Y. Yu and L. Ducas. Second order statistical behavior of LLL and BKZ. In *SAC*, pages 3–22, 2017.