mathematics

MDPI

Article

Lattice Enumeration with Discrete Pruning: Improvements, Cost Estimation and Optimal Parameters

Luan Luan [1,2], Chunxiang Gu [1,2], Yonghui Zheng [1,2,*] and Yanan Shi [1,2]

[1] Henan Key Laboratory of Network Cryptography Technology, Zhengzhou 450001, China
[2] PLA Information Engineering University, Zhengzhou 450001, China
* Correspondence: yonghui.zh@163.com

Abstract: Lattice enumeration is a linear-space algorithm for solving the shortest lattice vector problem (SVP). Extreme pruning is a practical technique for accelerating lattice enumeration, which has a mature theoretical analysis and practical implementation. However, these works have yet to be applied to discrete pruning. In this paper, we improve the discrete pruned enumeration (DP enumeration) and provide a solution to the problem proposed by Léo Ducas and Damien Stehlé regarding the cost estimation of discrete pruning. We first rectify the randomness assumption to more precisely describe the lattice point distribution of DP enumeration. Then, we propose a series of improvements, including a new polynomial-time binary search algorithm for cell enumeration radius, a refined cell-decoding algorithm and a rerandomization and reprocessing strategy, all aiming to lift the efficiency and build a more precise cost-estimation model for DP enumeration. Based on these theoretical and practical improvements, we build a precise cost-estimation model for DP enumeration by simulation, which has good accuracy in experiments. This DP simulator enables us to propose an optimization method of calculating the optimal parameters of DP enumeration to minimize the running time. The experimental results and asymptotic analysis both show that the discrete pruning method could outperform extreme pruning, which means that our optimized DP enumeration might become the most efficient polynomial-space SVP solver to date. An open-source implementation of DP enumeration with its simulator is also provided.

Keywords: lattice-based cryptanalysis; SVP; enumeration; discrete pruning

MSC: 94A60

Citation: Luan, L.; Gu, C.; Zheng, Y.; Shi, Y. Lattice Enumeration with Discrete Pruning: Improvements, Cost Estimation and Optimal Parameters. *Mathematics* **2023**, *11*, 766. https://doi.org/10.3390/math11030766

Academic Editors: Ding Wang, Qi Jiang and Chunhua Su

Received: 3 January 2023
Revised: 24 January 2023
Accepted: 28 January 2023
Published: 3 February 2023

1. Introduction

The shortest vector problem (SVP) and closest vector problem (CVP) are hard computing lattice problems, which have become central building blocks in lattice-based cryptanalysis. The security analysis of many lattice-based cryptographic primitives is usually reduced to solving the underlying mathematical problems, which are closely related to SVP and CVP. Some hard computing problems used in classical public-key cryptosystems can also be converted to a variant version of SVP or CVP, such as the knapsack problem [1–3], the hidden number problem [4] and the integer factoring problem [5].

Lattice enumeration is a general SVP solver with linear space complexity, which can be traced back to the early 1980s [6,7]. It outputs a lattice vector (or proves there is none) that is shorter than the given target length within superexponential time. Enumeration can also be used as the subroutine of the blockwise lattice basis reduction (BKZ) algorithm, and therefore plays an important role in the security analysis and parameter assessment of lattice-based cryptosystems [8–10].

Classical lattice enumeration can be traced back to the early 1980s. Kannan [6] proposed an algorithm to enumerate lattice points in a high-dimensional parallelepiped, which output the shortest vector, along with a well-reduced HKZ basis, at the cost of a huge time

consumption. Fincke and Pohst [7,11] proposed using a lighter preprocessing method, such as LLL or BKZ algorithms, and spending more time on enumerating the lattice vector in a hyperellipsoid. Fincke–Pohst enumeration has a higher level of time complexity in theory, but it performs better than Kannan's enumeration in practice.

Pruning is the most important technique to accelerate lattice enumeration. In the classical enumeration algorithm, all the coordinate vectors of lattice points are organized as an enumeration tree and are searched in a depth-first way. The pruning method cuts off the branch and stops searching in depth when the objective function value at the current node exceeds the bounding function. This might cut off the correct solution during searching; hence, enumeration becomes a probability algorithm. Gama, Nguyen and Regev [12] proposed the extreme pruning method, treating the bounding function as the solution to an optimization problem. The optimal bounding function can be regarded as an extreme point, which minimizes the expected total running time (with a given success probability). Therefore, the extreme pruning method is believed to be the most efficient pruning method for classical enumeration, which is also called GNR enumeration. The fplll library [13] provides an open-source implementation of GNR enumeration.

The classical pruned enumeration searches lattice vectors in a hypercylinder intersection, which is regarded as a continuous region in time analysis. Consequently, the computation of the expected running time of GNR enumeration is easy to handle, which implies that the upper bound on the cost of lattice enumeration is clear. Aono et al. also proved a lower bound on GNR enumeration [14].

The discrete pruning method is quite different. The discrete pruned enumeration (DP enumeration) originated from a heuristic "Sampling Reduction" algorithm [15], which iteratively samples lattice vectors under the restriction on their Gram–Schmidt coefficients and then rerandomizes the basis using lattice reduction. Ajtai, Buchmann and Ludwig [16,17] provided some analyses of the time complexity and success probability. Fukase and Kashiwabara [18] put forward a series of significant improvements, including the natural number representation (NNR) of lattice points, to make the sampling reduction method more practical and provided a heuristic analysis. Teruya et al. [19] designed a parallelized version of the Fukase–Kashiwabara sampling reduction algorithm and solved a 152-dimensional SVP challenge, which was the best record of that year. Other relevant studies include [20–22]. The Sampling Reduction algorithm shows good practicality but lacks sufficient theoretical support, especially regarding the parameter settings and estimation of running time. The conception of "discrete pruning" was formally put forward in EUROCRYPT' 17 by Aono and Nguyen [23]. They proposed a novel conception named "lattice partition" to generalize the previous sampling methods, and they solved the problem of what kind of lattice points should be "sampled" using the classical enumeration technique. The success probability of discrete pruning can be described as the volume of "ball-box intersection", and can be calculated efficiently using fast inverse Laplace transform(FILT). Aono et al. [24] made some modifications to DP enumeration and proposed a quantum variant. The theoretical foundation of DP enumeration was gradually developed, but some problems still remain.

1. A precise cost estimation: There is a gap between theoretical time complexity and the actual cost. It has been proved that each subalgorithm of DP enumeration has a polynomial-time complexity, but the actual running time is not in proportion with the theoretical upper bound, since subalgorithms with different structures are analyzed using different arithmetic operations.
2. The optimal parameter setting: The parameters of DP enumeration used to be set empirically, by hand. An important problem that [23,24] did not clearly explain is how many points should be enumerated in the iteration. The authors of [18,23] provided some examples of parameter selection, without further explanation. For a certain SVP instance, optimal parameter settings should exist to minimize the total running time. This is based on the solution to the first problem.

To solve the above problems, the whole work is carried out using two steps: First, we built a precise cost model for DP enumeration, called the "DP simulator". During this

procedure, some implementation details regarding DP enumeration are improved, and the cost model is based on these improvements. Second, we used the optimization method to find the optimal parameter of DP enumeration.

In the first step, to estimate the precise cost of DP enumeration, we studied and improved DP enumeration in terms of both mathematical theory and algorithm implementation. The main work was as follows:

1. *Rectification of the theoretical assumption:* It is generally assumed that the lattice point in participating "cells" follows the uniform distribution, but Ludwig [25] (Section 2.4) pointed out that this randomness assumption does not strictly hold for Schnorr's sampling method, i.e., Schnorr's partition. This defect also exists in the randomness assumption of natural number partition and leads to a paradox, where two symmetric lattice vectors with the same length have a different moment value and success probability. This will lead to inaccuracies in cell enumeration and success probability analyses; hence, we provided a rectified assumption to describe lattice point distribution in cells more cautiously and accurately, and consequently eliminate the defect.
2. *Improvements in algorithm implementation:* We propose a new polynomial-time binary search algorithm to find the cell enumeration radius, which guarantees a more precise output than [24] and is more conducive to building a precise cost model. We proposed using a lightweight rerandomization method and a truncated version of BKZ, "k-tours-BKZ", as the reprocessing method when DP enumeration fails in one round and has to be repeated. This method takes both the basis of quality and controllable running time into consideration. We examined the stabilization of basis quality during repeated reprocessing and proposed a model to describe the relationship between orthogonal basis information and the parameters of DP enumeration.
3. *A cost simulator of DP enumeration:* Based on the above improvements, we provided an open-source implementation of DP enumeration. To describe the actual running time of DP enumeration, it is necessary to define a unified "basic operation" for all subalgorithms of DP enumeration and fit the coefficients of polynomials. We calculated the fitted time cost formula in CPU-cycles for each subalgorithm of our implementation. We also modified the calculation procedures of success probability according to the rectified randomness assumption. Then, we built a cost model, the DP simulator, to estimate the exact cost of DP enumeration under any given parameters. In addition, for random lattices with GSA assumption holding, this works in a simple and efficient way, without computing any specific lattice basis.

In the second step, to predict the optimal parameter for DP enumeration, we proposed an optimization model. In this model, the DP enumeration parameter is regarded as the input of the DP simulator, and the output of the DP simulator is the estimated running time. We found that the Nelder–Mead method is suitable for solving the optimization problem, since the DP simulator has no explicit expression and has unknown derivatives. As the cost model is accurate, the parameter that minimizes the output of the DP simulator can also be considered the optimal parameter of the DP enumeration algorithm.

Contributions of this work: We propose a systematic solution to the open problem of DP enumeration by combining the improved implementation of DP enumeration, DP simulator and the optimization method used to find optimal parameters. The experiment confirms that DP enumeration outperforms the state-of-art enumeration with extreme pruning and provides concrete crossover points for algorithm performance. Furthermore, the experimental result is extrapolated to higher dimensions, and we provided the asymptotic expression of the cost of DP enumeration. These results provide valuable references for lattice-based cryptography. We released our implementation as an open source (https://github.com/LunaLuan9555/DP-ENUM, accessed on 30 December 2022) to promote the development of lattice-based cryptography.

Roadmap: Section 2 introduces the fundamental knowledge of lattices and provides an overview of pruning technologies for lattice enumeration. Section 3 first rectifies the

basic randomness assumption of lattice partition, and then describes the details of three improvements in discrete pruning enumeration. Section 4 shows the details of our DP enumeration cost simulator, including the runtime estimation of every subalgorithm and the rectified success probability model. Section 5 describes how to find the optimal parameter setting for DP enumeration using our cost simulator. Section 6 provides experimental results to verify the accuracy of our cost simulator, and compares the efficiency of our implementation with the extreme pruned enumeration in fplll library. Finally, Section 7 provides the conclusion and discusses some further works.

2. Preliminaries

2.1. Lattice

Lattice: Let $\mathbb{R}^m$ denote the m-dimensional Euclidean space. Given n linear independent vectors $\mathbf{b}_1, \ldots, \mathbf{b}_n \in \mathbb{R}^m$ ($m \geq n$), a *lattice* $\mathcal{L}$ is defined by a set of points in $\mathbb{R}^m$: $\mathcal{L} = \{\sum_{i=1}^{n} x_i \mathbf{b}_i : x_i \in \mathbb{Z}\}$. The vector set $\{\mathbf{b}_1, \ldots, \mathbf{b}_n\}$ is called a *basis* of lattice $\mathcal{L}$ and can be written in the form of column matrix $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_n]$. The rank of matrix $\mathbf{B}$ is n, which is also known as the *dimension of lattice*. From a computational perspective, we can only consider the case that $\mathbf{b}_i \in \mathbb{Z}^m$ for $i = 1, \ldots, n$ for convenience, since the real number is represented by a rational number in the computer, and a lattice with a rational basis can always be scaled to one with an integral basis. The lattice is *full-rank* when $m = n$, which is a common case in lattice-based cryptanalysis. In the following, we only consider the case $\mathbf{B} \in \mathbb{Z}^{n \times n}$.

A lattice has many different bases. Given two bases $\mathbf{B}_1, \mathbf{B}_2 \in \mathbb{Z}^{m \times n}$ of a lattice $\mathcal{L}$, there exists a unimodular matrix $\mathbf{U}$ such that $\mathbf{B}_1 = \mathbf{B}_2\mathbf{U}$. A basis of the lattice corresponds to a *fundamental parallelepiped* $\mathcal{P}(\mathbf{B}) = \{\sum_{i=1}^{n} \mathbf{b}_i x_i : 0 \leq x_i < 1, i = 1, \ldots, n\}$. The shape of fundamental parallelepiped varies depending on the basis, but the volume of those fundamental parallelepipeds is an invariant of the lattice, which is denoted by $\mathrm{vol}(\mathcal{L})$. This is also called the *determinant* $\det(\mathcal{L})$ of a lattice, and we have $\det(\mathcal{L}) = \mathrm{vol}(\mathcal{L}) = \sqrt{\det(\mathbf{B}^T\mathbf{B})}$.

Random lattice: The formal definition and generation algorithm of a random lattice can refer to Goldstein and Mayer's work in [26]. The SVP challenge also adopts the Goldstein–Mayer lattice. The lattice of an n-dimensional SVP challenge instance has a volume of about 2^{10n}.

Gaussian heuristic: For a lattice $\mathcal{L}$ and a measurable set S in $\mathbb{R}^n$, we intuitively expect that the set contains $\mathrm{vol}(S)/\mathrm{vol}(\mathcal{L})$ fundamental parallelepipeds; therefore, there should be the same number of points in $S \cap \mathcal{L}$.

Assumption 1. *Gaussian heuristic. Let $\mathcal{L}$ be a n-dimensional lattice in $\mathbb{R}^n$ and S be a measurable set of $\mathbb{R}^n$. Then,*

$$\#\{S \cap \mathcal{L}\} \approx vol(S)/vol(\mathcal{L})$$

We note that the Gaussian heuristic should be used carefully, because in some "bad" cases, this assumption does not hold (see Section 2.1.2 in [27]). However, in random lattice, this assumption generally holds, especially for some "nice" set S; therefore, we can use the Gaussian heuristic to predict $\lambda_1(\mathcal{L})$:

$$\mathrm{GH}(\mathcal{L}) = \frac{\mathrm{vol}(\mathcal{L})^{1/n}}{B_n(1)^{1/n}} = \frac{1}{\sqrt{\pi}}\Gamma\left(\frac{n}{2}+1\right)^{\frac{1}{n}}\mathrm{vol}(\mathcal{L})^{\frac{1}{n}} \approx \sqrt{\frac{n}{2\pi e}}\mathrm{vol}(\mathcal{L})^{\frac{1}{n}} \quad (1)$$

In fact, $\mathrm{GH}(\mathcal{L})$ is exactly the radius of an n-dimensional ball with volume $\mathrm{vol}(\mathcal{L})$. It is widely believed that $\mathrm{GH}(\mathcal{L})$ is a good estimation of $\lambda_1(\mathcal{L})$ when $n \gtrapprox 45$.

Shortest vector problem (SVP): For a lattice $\mathcal{L} = \mathcal{L}(\mathbf{B})$ with basis $\mathbf{B} \in \mathbb{Z}^{m \times n}$, one can find a lattice vector $\mathbf{Bx}$ with $\mathbf{x} \in \mathbb{Z}^n \setminus \{\mathbf{0}\}$ such that $\|\mathbf{Bx}\| \leq \|\mathbf{By}\|$ for any $\mathbf{y} \in \mathbb{Z}^n \setminus \{\mathbf{0}\}$. The length of the shortest vector is denoted by $\lambda_1(\mathcal{L})$.

It is of great interest to find the shortest nonzero vector of a lattice in the fields of complexity theory, computational algebra and cryptanalysis. However, a more common

case in cryptanalysis is to find a lattice vector that is shorter than a given bound. In other words, researchers are more interested in finding an approximate solution to SVP. For example, the target of the SVP challenge [28] is to find a lattice vector $\mathbf{v} \in \mathcal{L}$ such that $\|\mathbf{v}\|^2 \leq 1.05 \cdot \mathrm{GH}(\mathcal{L}) \approx 1.05\lambda_1(\mathcal{L})$.

Orthogonal projection: The Gram–Schmidt orthogonalization can be considered a direct decomposition of lattice and is frequently used in lattice problems.

Definition 1. *Gram–Schmidt orthogonalization. Let* $\mathbf{B} = [\mathbf{b}_1, \dots \mathbf{b}_n] \in \mathbb{Z}^{m \times n}$ *be a lattice basis, The Gram–Schmidt orthogonal basis* $\mathbf{B}^* = [\mathbf{b}_1^*, \dots \mathbf{b}_n^*] \in \mathbb{Q}^{m \times n}$ *is defined with* $\mathbf{b}_i^* = \mathbf{b}_i - \sum_{j=1}^{i-1} \mu_{i,j}\mathbf{b}_j^*$, *where the orthogonal coefficient* $\mu_{i,j} = \frac{\langle \mathbf{b}_i, \mathbf{b}_j^* \rangle}{\langle \mathbf{b}_j^*, \mathbf{b}_j^* \rangle}$.

Definition 2. *Orthogonal projection. Let* $\pi_i : \mathbb{R}^m \to span(\mathbf{b}_1, \dots, \mathbf{b}_{i-1})^\perp$ *be the i-th orthogonal projection. For* $\mathbf{v} \in \mathbb{R}^m$, *we define* $\pi_i(\mathbf{v}) = \mathbf{v} - \sum_{j=1}^{i-1} \frac{\langle \mathbf{v}, \mathbf{b}_j^* \rangle}{\|\mathbf{b}_j^*\|^2} \mathbf{b}_j^*$. *Since any lattice vector* $\mathbf{v}$ *can be represented by the orthogonal basis* $\mathbf{B}^*$ *as* $\mathbf{v} = \sum_{i=1}^{n} u_i \mathbf{b}_i^*$, *we also have* $\pi_i(\mathbf{v}) = \sum_{j=i}^{n} u_j \mathbf{b}_j^*$.

For lattice $\mathcal{L}(\mathbf{B})$ and $i = 1, \dots, n$, we can define the $n - i + 1$-dimensional projected lattice $\pi_i(\mathcal{L}(\mathbf{B})) = \mathcal{L}([\pi_i(\mathbf{b}_i), \pi_i(\mathbf{b}_{i+1}), \dots, \pi_i(\mathbf{b}_n)])$. Note that the orthogonal basis of $\pi_i(\mathcal{L}(\mathbf{B}))$ is exactly $[\mathbf{b}_i^*, \mathbf{b}_{i+1}^*, \dots, \mathbf{b}_n^*]$.

2.2. Discrete Pruning

In classical enumeration, we search for lattice points directly, according to their coordinates $(x_1, \dots, x_n)$ with respect to basis $\mathbf{B}$, such that $\|\mathbf{v}\| = \|\sum_{i=1}^{n} x_i \mathbf{b}_i\| \leq R$. However, enumeration with discrete pruning behaves in a very different way.

Considering the representation $\mathbf{v} = \sum_{j=1}^{n} u_j \mathbf{b}_j^*$, it is intuitive to search for a lattice vector with a small $|u_j|$, especially for index j, corresponding to a very large $\|\mathbf{b}_j^*\|$. This idea is first applied in a heuristic vector sampling method proposed by Schnorr [15] and dramatically improved by Fukase and Kashiwabara [18]. These sampling strategies are summarized by Aono and Nguyen, and they defined *lattice partition* to generalize these sampling methods.

Definition 3 (Lattice partition [23])**.** *Let* $\mathcal{L}$ *to be a full-rank lattice in* $\mathbb{Z}^n$. *An* $\mathcal{L}$*-partition* $(\mathcal{C}(), T)$ *is a partition of* $\mathbb{R}^n$ *such that:*

- $\mathbb{R}^n = \cup_{\mathbf{t} \in T} \mathcal{C}(\mathbf{t})$ *and* $\mathcal{C}(\mathbf{t}) \cap \mathcal{C}(\mathbf{t}') = \varnothing$. *The index set* T *is a countable set.*
- *There is exactly one lattice point in each cell* $\mathcal{C}(\mathbf{t})$, *and there is a polynomial time algorithm to convert a tag* $\mathbf{t}$ *to the corresponding lattice vector* $\mathbf{v} \in \mathcal{C}(\mathbf{t}) \cap \mathcal{L}$.

A nontrivial partition is generally related to the orthogonal basis $\mathbf{B}^*$. Some examples are given in [23]. Here, we only introduce natural partition, which was first proposed by Fukase and Kashiwabara [18], since it has smaller moments than other lattice partitions such as Babai's and Schnorr's, implying that enumeration with natural partition tends to have more efficiency. In this paper, we only discuss discrete pruning based on natural partition, following the work of [18,23,24].

Definition 4. *Given a lattice* $\mathcal{L}$ *with the basis* $\mathbf{B} \in \mathbb{Z}^{n \times n}$, *and a lattice vector* $\mathbf{v} = \sum_{j=1}^{n} u_j \mathbf{b}_j^*$, *the natural number representation (NNR) of* $\mathbf{v}$ *is a vector* $\mathbf{t} = (t_1, \dots, t_n) \in \mathbb{N}^n$ *such that* $u_j \in \left(-\frac{t_j+1}{2}, -\frac{t_j}{2}\right] \cup \left(\frac{t_j}{2}, \frac{t_j+1}{2}\right]$ *for all* $j = 1, \dots, n$. *Then, natural number representation* $\mathbf{t} \in \mathbb{N}^n$ *leads to the natural partition* $(\mathcal{C}(), \mathbb{N}^n)$ *by defining*

$$\mathcal{C}(\mathbf{t}) = \left\{ \sum_{i=1}^{n} x_i \mathbf{b}_i^*, -\frac{t_i+1}{2} < x_i \leq -\frac{t_i}{2} \text{ or } \frac{t_i}{2} < x_i \leq \frac{t_i+1}{2} \right\}$$

The shape of $\mathcal{C}(\mathbf{t}) = \left\{\sum_{i=1}^{n} x_i \mathbf{b}_i^*, -\frac{t_i+1}{2} < x_i \leq -\frac{t_i}{2} \text{ or } \frac{t_i}{2} < x_i \leq \frac{t_i+1}{2}\right\} \subsetneq \mathbb{R}^n$ is a union of 2^j hypercuboids (boxes), which are centrosymmetric and disjoint, where j is the number of nonzero coefficients in $\mathbf{t}$.

Given a certain cell, the lattice vector lying in it can be determined by the tag and lattice basis; however, if we randomly pick some cells, the position of the lattice vector in $\mathcal{C}(\mathbf{t})$ always shows a kind of randomness. We can naturally assume that the lattice point belonging to $\mathcal{C}(\mathbf{t})$ follows a random uniform distribution. The prototype of this assumption was first proposed by Schnorr [15] and generalized by Fukase and Kashiwabara [18]. Aono, Nguyen and Shen [23,24] also use this assumption by default.

Assumption 2 (Randomness Assumption [18]). *Given a lattice $\mathcal{L}(\mathbf{B}) \in \mathbb{Z}^{n\times n}$ with orthogonal basis $\mathbf{B}^*$ and a natural number vector $\mathbf{t} \in \mathbb{N}^n$, for the lattice vector $\mathbf{v} = \sum_{j=1}^{n} u_j \mathbf{b}_j^*$ contained in $\mathcal{C}(\mathbf{t})$, the Gram–Schmidt coefficients u_j $(j = 1, \ldots, n)$ are uniformly distributed over $\left(-\frac{t_j+1}{2}, -\frac{t_j}{2}\right] \cup \left(\frac{t_j}{2}, \frac{t_j+1}{2}\right]$ and statistically independent with respect to j.*

In such an ideal situation, by considering the GS coefficients u_j of $\mathbf{v}$ as random variables, one can compute the expectation and variance value of $\|\mathbf{v}\|^2$, since $\|\mathbf{v}\|^2$ can also be considered a random variable. They are also defined as the first moment of cell $\mathcal{C}(\mathbf{t})$ [23]:

$$E[\mathcal{C}(\mathbf{t})] = \frac{1}{12}\sum_{j=1}^{n}(3t_j^2 + 3t_j + 1)\|\mathbf{b}_j^*\|^2 \tag{2}$$

This means that, for a given tag $\mathbf{t}$, we can predict the length of the lattice vector $\mathbf{v} \in \mathcal{C}(\mathbf{t})$ immediately without converting $\mathbf{t}$ to $\mathbf{v}$, which is precise but takes a rather long time. This leads to the core idea of the discrete pruning method: we first search for a batch of cells $\cup_{\mathbf{t}\in S}\mathcal{C}(\mathbf{t})$ that are "most likely" to contain very short lattice vectors; then, we decode them to obtain the corresponding lattice vectors and check if there is a $\mathbf{v}$ such that $\|\mathbf{v}\| \leq R$. The pruning set is $P = \cup_{\mathbf{t}\in S}\mathcal{C}(\mathbf{t})$. If the randomness assumption and Gaussian heuristic hold, the probability that P contains a lattice vector shorter than R can be easily described by the volume of the intersection $\mathrm{vol}(Ball_n(R) \cap P) = \sum_{\mathbf{t}\in S}\mathrm{vol}(Ball_n(R) \cap \mathcal{C}(\mathbf{t}))$.

The outline of discrete pruned enumeration is given in Algorithm 1.

Algorithm 1 Discrete Pruned Enumeration

Require: well-reduced lattice basis $\mathbf{B}$, number of tags M, target vector length R
Ensure: $\mathbf{v}\mathcal{L}(\mathbf{B})$ such that $\|\mathbf{v}\| < R = 1.05 \cdot \mathrm{GH}(L)$

```
Reduce lattice basis B
while true do
    S ← ∅
    Use binary search to find bound r such that there are M tags t satisfying f(t) < r;
    Enumerate all these M tags and save them in set S
    for t ∈ S do
        Decode t to recover the corresponding v such that v ∈ C(t) ;
        if ||v||^2 < R^2 then return v;                          // Find a solution
        end if
    end for
    Rerandomize B by multiplying a unimodular matrix Q ∈ Z^{n×n}, i.e., B ← BQ
    Reprocess B using lattice reduction algorithms such as BKZ or LLL
end while
```

3. Improvements in Discrete Pruning Method

3.1. Rectification of Randomness Assumption

Most of the studies on discrete pruning take randomness assumption as a foundation of their analyses; therefore, they can apply Equation (2) to predict vector length. However, we can easily point out a paradox if the following assumption holds: for two cells with tag $\mathbf{t} = [t_1, \ldots, t_k \neq 0, 0, \ldots, 0]$ and $\mathbf{t}' = [t_1, \ldots, t_k + 1 \neq 0, 0, \ldots, 0]$, if t_k is odd; then, it is easy

to verify that the corresponding lattice vectors of $\mathbf{t}$ and $\mathbf{t}'$ are in opposite directions with the same length. However, the Equation (2) indicates $E[\mathcal{C}(\mathbf{t})] < E[\mathcal{C}(\mathbf{t}')]$. In fact, we also have $\mathrm{vol}(Ball_n(R) \cap \mathcal{C}(\mathbf{t})) \neq \mathrm{vol}(Ball_n(R) \cap \mathcal{C}(\mathbf{t}'))$, which means these two cells have different success probabilities, while the lattice vectors contained in them are essentially the same.

This paradox implies that the distribution of lattice points in cells is not completely uniform. As a matter of fact, for a tag $\mathbf{t} = [t_1, \ldots, t_k \neq 0, 0, \ldots, 0]$, GS coefficient $u_k, u_{k+1}, \ldots, u_n$ of the corresponding lattice vector, $\mathbf{v} \in \mathcal{C}(\mathbf{t})$ are fixed integers rather than uniformly distributed real numbers. The exact values of $u_k, u_{k+1}, \ldots, u_n$ are given in Proposition 1.

Proposition 1. *Given a lattice $\mathcal{L}(\mathbf{B})$ with orthogonal basis $\mathbf{B}^*$ and a tag $\mathbf{t} = [t_1, \ldots, t_k \neq 0, 0, \ldots, 0]$, the corresponded lattice vector is denoted by $\mathbf{v} = \sum_{j=1}^{n} u_j \mathbf{b}_j^* \in \mathcal{C}(\mathbf{t})$, then*

$$\begin{aligned} u_k &= \begin{cases} \dfrac{-t_k}{2}, & \text{if } t_k \text{ is even} \\ \dfrac{t_k + 1}{2}, & \text{if } t_k \text{ is odd} \end{cases} \\ u_{k+1} &= \ldots = u_n = 0 \end{aligned} \tag{3}$$

Proof. We can verify the proposition through the procedures of decoding algorithm, and a brief theoretical proof is also provided. For lattice vector $\mathbf{v} = \sum_{i=1}^{n} x_i \mathbf{b}_i = \sum_{i=1}^{n} u_i \mathbf{b}_i^* \in \mathcal{C}(\mathbf{t})$, where $u_k = x_k + \sum_{i=k+1}^{n} \mu_{i,k} x_i$, the last nonzero coefficient of $\mathbf{x}$ is x_k if, and only if, u_k is the last nonzero coefficient of $\mathbf{u}$, and $u_k = x_k$. Then, according to Definition 4, we have $t_j = 0$ for all $j > k$; u_k is non-negative if, and only if, $t_k = 2x_k - 1$ is odd; $u_k < 0$ if and only if $t_k = -2x_k$ is even. For brevity, the tags with odd and even numbers in the last nonzero coefficient are called the "odd-ended tag" and "even-ended tag", respectively. □

Based on Proposition 1, the rectified randomness assumption is given below, and the moments of natural partition are also modified.

Assumption 3 (The Rectified Randomness Assumption)**.** *Let $\mathcal{L}(\mathbf{B})$ be an n-dimensional lattice with orthogonal basis $\mathbf{B}^*$. Given a tag $\mathbf{t}$ with corresponding lattice vector $\mathbf{v} = \sum_{j=1}^{n} u_j \mathbf{b}_j^* \in \mathcal{C}(\mathbf{t})$, suppose the last nonzero coefficient of $\mathbf{t}$ is t_k; then, for $j < k$, we assume that u_j is uniformly distributed over $\left(-\frac{t_j+1}{2}, -\frac{t_j}{2}\right] \cup \left(\frac{t_j}{2}, \frac{t_j+1}{2}\right]$ and independent with respect to j, for $j \geq k$, u_j can be directly given by proposition (3).*

Then, the moments of lattice partition should also be modified, since the last several coefficients are not random variables after the rectification. For a tag $\mathbf{t} = [t_1, \ldots, t_k \neq 0, 0, \ldots, 0]$, the expectation of the corresponding $\|\mathbf{v}\|^2$ is

$$E'[\mathcal{C}(\mathbf{t})] = \frac{1}{12} \sum_{j=1}^{k-1} (3t_j^2 + 3t_j + 1)\|\mathbf{b}_j^*\|^2 + u_k^2 \|\mathbf{b}_k^*\|^2 \tag{4}$$

where u_k is defined by Equation (3).

After the rectification, for two tags $\mathbf{t}, \mathbf{t}'$, which only differ by 1 in the last nonzero coefficient, we have $E'[\mathcal{C}(\mathbf{t})] = E'[\mathcal{C}(\mathbf{t}')]$, and the paradox mentioned in the beginning of this subsection is eliminated.

3.2. Binary Search and Cell Enumeration

A crucial step of Algorithm 1 is called *cell enumeration* (line 5), aiming to find the "best" M cells. We use the objective function $f(\mathbf{t})$ to measure how good the cells are. $E[\mathcal{C}(\mathbf{t})]$ (Equation (2)) is a tacit indicator for searching for the proper $\mathbf{t}$, since it is exactly the expected squared length of lattice vector $\mathbf{v} \in \mathcal{C}(\mathbf{t})$. Aono and Gama [23] directly use $E[\mathcal{C}(\mathbf{t})]$ as the objective function $f(\mathbf{t})$ in cell enumeration, and Aono et al. [24] use a modified version of $E[\mathcal{C}(\mathbf{t})]$ to guarantee its polynomial runtime. They require the function $f(\mathbf{t}) = \sum_{i=1}^{n} f(i, t_i)$

satisfying $f(i,0) = 0$ and $f(i,j) \geq f(i,j')$ for all i and $j > j'$, which means we have to drop the constants in $E[\mathcal{C}(\mathbf{t})]$, i.e., let $f(i,j) = \frac{1}{4}(j^2+j)\|\mathbf{b}_i^*\|^2$. Based on their work and the rectification above, we propose a modified objective function. Given a tag vector $\mathbf{t} = [t_1, \ldots, t_k \neq 0, 0, \ldots, 0]$ as input, we first define

$$f(i, t_i) \stackrel{\text{def}}{=} \begin{cases} 0, & \text{for } i > k \\ u_i^2 \|\mathbf{b}_i^*\|^2, & \text{for } i = k \\ \frac{1}{4}(t_i^2 + t_i)\|\mathbf{b}_i^*\|^2, & \text{else} \end{cases} \tag{5}$$

where u_i is defined by Equation (3). Then, the objective function of cell enumeration is

$$f(\mathbf{t}) \stackrel{\text{def}}{=} \sum_{i=1}^{n} f(i, t_i) = \frac{1}{4}\sum_{i=1}^{n}(t_i^2 + t_i)\|\mathbf{b}_i^*\|^2 = \frac{1}{4}\sum_{i=1}^{k}(t_i^2 + t_i)\|\mathbf{b}_i^*\|^2 + u_k^2\|\mathbf{b}_k^*\|^2 \tag{6}$$

The complete cell enumeration procedure is given below:

Remark 1. *Remark. Considering the symmetry of the lattice vector, we only search for even-ended tags (line 17 and line 31).*

The time complexity of Algorithm 2 is similar to that of [24]: the number of times that Algorithm 2 enters the loop is, at most, $(2n-1) \cdot M/2$, where M is the number of tags such that $f(\mathbf{t}) \leq r$. For each loop iteration, the number of arithmetic operations performed is $O(1)$, and the number of calls to $f()$ is exactly one. The proof is essentially the same as that of theorem 11 in [24]. (Note that, although we change the definition of $f(i, t_i)$, and therefore change the value calculated in line 3, this does not affect the total number of while loops in the asymptotic sense. Furthermore, the modification of $f(\mathbf{t})$ does not change the key step in the proof: each partial assignment $\sum_{i=i_0}^{n} f(i, t_i) \leq R$ of a middle node can be expanded to a larger sum $\sum_{i=1}^{n} f(i, t_i) \leq R$.)

In cell enumeration, a bound r should be determined as in the previous section, such that there are exactly M tags satisfying $f(\mathbf{t}) \leq r$. Aono and Nugyen [23] first proposed the idea to use the binary search method to find a proper r. Aono et al. [24] gave a detailed binary search algorithm (Algorithm 5 in [24]), which was proved to have a polynomial running time $O(n(n+1)M) + n^{O(1)} + O(\log_2 M)$. Their algorithm uses the input (radius r) precision to control the termination of the binary search, but a slight vibration of the radius r will cause a large disturbance to the number of valid tags, since the number of tags is $\mathbf{t}$, such that $f(\mathbf{t}) < r$ grows exponentially with r. Therefore, their binary search method could only guarantee an output of N tags with $N \in [M, (n+1)M]$, which is a relatively large interval. Then, it would be intractable to estimate the precise cost of the subsequent tagwise calculation, such as in the decoding algorithm.

Since the current termination condition in the binary search of DP enumeration will bring uncertainty into the total cost-estimation model, we provide a more practical and stable polynomial-time binary search strategy to determine the parameter r for cell enumeration. The essential difference with ANS18 [24] is that we use the number of output tags as the bisection indicator of binary search. This method guarantees that cell enumeration algorithm (Algorithm 1, line 5) outputs about $(1-\epsilon)M$ to $(1+\epsilon)M$ tags $\mathbf{t}$, satisfying $f(\mathbf{t}) < r$. When ϵ is small, the number of output tags can be approximately counted as M.

Algorithm 2 CellENUM

Require: Orthongonal basis $\mathbf{B}^* = [\mathbf{b}_1^*, \ldots, \mathbf{b}_n^*]$, r
Ensure: All $\mathbf{t} \in \mathbb{N}^n$ such that $f(\mathbf{t}) \leq r$ where $f(\mathbf{t})$, as defined in Equation (6), and $\mathbf{t}$ is even-ended

1: $S \leftarrow \varnothing$
2: $t_1 = t_2 = \ldots = t_n = 0$;
3: $c_1 = c_2 = \ldots = c_{n+1} = 0$;
4: $k \leftarrow 1$
5: **while true do**
6: **if** $t_k = t_{k+1} = t_n = 0$ **then**
7: $c_k \leftarrow 0$
8: **else if** t_k is the last nonzero component of $\mathbf{t}$ **then**
9: $c_k \leftarrow u_k^2 \|\mathbf{b}_k^*\|^2$ // u_k is calculated using Equation (3)
10: **else**
11: $c_k \leftarrow c_{k+1} + \frac{1}{4}(t_i^2 + t_i)\|\mathbf{b}_i^*\|^2$;
12: **end if** // calculating $f(k, t_k)$, defined by Equation (5)
13: **if** $c_k < r$ **then**
14: **if** $k = 1$ **then**
15: $S \leftarrow S \cup \{\mathbf{t} = (t_1, t_2, \ldots, t_n)\}$
16: **if** $t_{k+1} = \ldots = t_n = 0$ **then**
17: $t_k \leftarrow t_k + 2$ // Only output "even-ended" tags
18: **else**
19: $t_k \leftarrow t_k + 1$
20: **end if**
21: **else**
22: $k \leftarrow k - 1$
23: $t_k \leftarrow 0$
24: **end if**
25: **else**
26: $k \leftarrow k + 1$
27: **if** $k = n + 1$ **then**
28: **exit**
29: **else**
30: **if** $t_{k+1} = \ldots = t_n = 0$ **or** $k = n$ **then**
31: $t_k \leftarrow t_k + 2$;
32: **else**
33: $t_k \leftarrow t_k + 1$;
34: **end if**
35: **end if**
36: **end if**
37: **end whilereturn** S

Remark 2. *In Algorithm 3, CellENUM in line 3 is actually a variant of the original Algorithm 2. It only needs to count the number of qualified tags and return early when* $\#S > (1 + \epsilon)M$*, and it has no need to store the valid tags.*

Algorithm 3 ComputeRadius

Require: $M, \epsilon, \mathbf{B}^*$
Ensure: $r \in \mathbb{R}$ such that $\#\{\mathbf{t} : f(\mathbf{t}) < r\} \approx M$
1: $R_l \leftarrow \sum_{i=1}^{n} f(i, 0) = 0$
2: $R_r \leftarrow \sum_{i=1}^{n} f(i, \lceil M^{\frac{1}{n}} \rceil)$
3: **while** $R_l < R_r$ **do**
4: $R_m \leftarrow \frac{R_l + R_r}{2}$
5: $S \leftarrow \mathsf{CellENUM}(\mathbf{B}, R_m)$
6: **if** $\#S < (1-\epsilon)M$ **then**
7: $R_l \leftarrow R_m$ // R_m is too small
8: **else if** $\#S > (1+\epsilon)M$ **then**
9: $R_r \leftarrow R_m$ // R_m is too large
10: **elsereturn** $r \leftarrow R_m$ // R_m is acceptable
11: **end if**
12: **end while**

Theorem 1 gives the asymptotic time complexity of Algorithm 3:

Theorem 1. *Given lattice $\mathcal{L}(\mathbf{B})$, M, a relaxation factor ϵ, Algorithm 3 ends within $O\left(\log n + \log \frac{1}{\epsilon} + n \log\left(ndet(L)^{\frac{2}{n}}\right)\right)$ loops and output the enumeration parameter r, such that $(1-\epsilon)M \leq \#\{\mathbf{t} \in \mathbb{N}^n : f(\mathbf{t}) < R\} \leq (1+\epsilon)M$. In each loop, subalgorithm cellENUM is called exactly once.*

The approximate proof of Theorem 1 is given in Appendix A. In the following experiments, we set $\epsilon = 0.005$.

3.3. Lattice Decoding

The decoding algorithm converts a tag $\mathbf{t} \in \mathbb{N}^n$ to a lattice vector $\mathbf{v} \in \mathcal{L}(\mathbf{B}) \subset \mathbb{R}^n$. The complete algorithm is described both in [18,23]. However, in discrete pruned enumeration, almost all the tags we process do not correspond to the solution for SVP, and there is no need to recover the coordinates of those lattice vectors. Instead, inspired by classical enumeration, we use an intermediate result, the partial sum of the squared length of lattice vector (line 14 in Algorithm 4), as an early-abort indicator: when the projected squared length of lattice vector $\rho = \sum_{k=i}^{n} |x_k + \sum_{i=k+1}^{n} \mu_{ik} x_i|^2 \|\mathbf{b}_k^*\|^2$ is larger than the target length of SVP, we stop the decoding, since it is not a short lattice vector. Therefore, we avoid a vector–matrix multiplication with time $O(n^2)$.

Algorithm 4 has $O(n)$ loops and, in each loop, there are about $O(n)$ arithmetic operations, which are mainly organized by line 7. Therefore, the time complexity of Algorithm 4 is $O(n^2)$. During experiments, we noticed that, for the SVP challenge, decoding terminates at index $i \approx 0.21n$ on average, which means that the early-abort technique works and saves decoding time.

Space complexity of DP enumeration. We note that Decode can be embedded into line 15 in CellENUM. In other words, the decoding procedure can be instantly determined when a candidate tag is found, and then we can decide whether to output the final solution to SVP or throw out the tag. This indicates that DP enumeration essentially has polynomial space complexity, since it has no need to store any tags.

Algorithm 4 Decode

Require: tag $\mathbf{t} \in \mathbb{N}^n$, SVP target length $R = 1.05 \cdot \mathrm{GH}(L)$, orthogonalization information $\mathbf{U} = (\mu_{i,j})_{n \times n}$, $\mathbf{B}^* \in \mathbb{R}^{n \times n}$;
Ensure: lattice vector $\mathbf{v}$ such that $\|\mathbf{v}\|^2 < R^2$ or output $\varnothing$
1: $\rho \leftarrow 0$
2: $\Delta \leftarrow 1$ // to indicate whether we should stop decoding;
3: **for** $i = i$ **to** n **do**
4: $u_i = 0$
5: **end for**
6: **for** $i = n$ **to** 1 **do**
7: $y = -\sum_{j=i+1}^{n} u_j \mu_{j,i}$
8: $u_i = \lfloor y + 0.5 \rfloor$
9: **if** $u_i \leq y$ **then**
10: $u_i = u_i - (-1)^{t_i} \lceil t_i/2 \rceil$
11: **else**
12: $u_i = u_i + (-1)^{t_i} \lceil t_i/2 \rceil$
13: **end if**
14: $\rho \leftarrow \rho + (u_i - y)^2 \|\mathbf{b}_i^*\|^2$ // $\rho = \sum_{k=i}^{n} |x_k + \sum_{i=k+1}^{n} \mu_{ik} x_i|^2 \|\mathbf{b}_k^*\|^2$
15: **if** $\rho > R$ **then**
16: $\Delta \leftarrow 0$
17: **exit**
18: **else**
19: $\Delta \leftarrow 1$ // find a solution to SVP
20: **end if**
21: **end for**
22: **if** $\Delta = 1$ **then return** $\mathbf{v} = \mathbf{Bu}$
23: **else return** $\varnothing$
24: **end if**

3.4. Lattice Reduction for Reprocessing

3.4.1. Rerandomization and Reduction

To solve an SVP instance, the DP enumeration should be repeatedly run on many different bases, which means that the lattice basis should be rerandomized when DP enumeration restarts; hence, it should be reprocessed to maintain good quality. A plain reduction method is to use the polynomial-time LLL reduction as the reprocessing method, which only guarantees some primary properties and is not as good as BKZ reduction. However, a complete BKZ reduction will take a long time, and the estimation of its runtime requires a sophisticated method. Additionally, the BKZ algorithm produces diminishing returns: after the first dozens of iterations, the quality of the basis, such as the root Hermite factor, changes slowly during iterations, as illustrated in [29,30].

A complete reduction is unnecessary, since our DP enumeration algorithm does not require that the lattice basis is strictly BKZ-reduced. The key point of reprocessing for DP enumeration is to achieve a compromise between time consumption and basis quality. An early-abort strategy called terminating BKZ [31] is a good attempt to decrease the number of iterations of BKZ reduction while maintaining some good properties. However, the runtime of BKZ is still hard to estimate, since the number of iterations is not fixed, and those properties mainly describe the shortness of $\mathbf{b}_1$, providing little help for our cost estimation.

Another idea is to use a BKZ algorithm with limited tours of reduction, which is convenient for runtime analysis and also efficiently produces a well-reduced basis. This is what we call the "k-tours-BKZ" algorithm (Algorithm 5), which restricts the total number of "tours" (lines 4–18 in Algorithm 5) of BKZ within k. Given BKZ blocksize β and k, the time consumption of Algorithm 5 can be approximately estimated by multiplying $k(n - \beta)$ and the cost of solving a single β-dimensional SVP oracle. This is explained in Section 4.

Algorithm 5 k-tours-BKZ

Require: Lattice basis$\mathbf{B}$; BKZ blocksize β, k
Ensure: a reprocessed lattice basis $\mathbf{B}'$
$Z \leftarrow 0$; $i \leftarrow 0$; // Z is used to judge the termination condition for original BKZ;
$K \leftarrow 0$; // K records the tours;
LLL($\mathbf{b}_1, \ldots, \mathbf{b}_n$);
while $Z < n-1$ **or** $K < k$ **do**
  $K++$
  for $i = 1$ **to** $n-1$ **do**
    $j \leftarrow \min(i+\beta-1, n)$;
    $h \leftarrow \min(j+1, n)$;
    $\mathbf{v} \leftarrow \mathrm{ENUM}(\pi_i(\mathbf{b}_i), \ldots, \pi_i(\mathbf{b}_j))$; // call the SVP oracle
    if $\mathbf{v} \neq \mathbf{0}$ **then**
      $Z \leftarrow 0$;
      LLL($\mathbf{b}_1, \ldots, \mathbf{b}_{i-1}, \sum_{s=i}^{j} v_s \mathbf{b}_s, \mathbf{b}_i, \ldots, \mathbf{b}_h$);
    else
      $z++$;
      LLL($\mathbf{b}_1, \ldots, \mathbf{b}_h$);
    end if
  end for
end while

The value of k is tightly related to the rerandomization; hence, the rerandomization method should also be cautiously discussed. The essence of this idea is to generate a unimodular matrix Q as the elementary column transformation performed on $\mathbf{B}$; therefore, $\mathbf{BQ}$ will become the new basis waiting to be reprocessed. A very "heavy" matrix Q means the basis is transformed with high intensity, which implies that the basis will lose many good properties after rerandomization, e.g., some very short basis vectors, which were achieved by the previous reduction, and the reprocessing procedure needs a long time to reach the well-reduced status. However, a very sparse Q may lead to insufficient randomness during transformation, which may not guarantee a very different basis. To balance the randomness and reprocessing cost, we heuristically require a Hamming distance between $\mathbf{Q}$ and an identity matrix $\mathbf{I}$ satisfying

$$d_1(\mathbf{Q}, \mathbf{I}) \overset{def}{=} \sum_{1 \le i \le n} \sum_{1 \le j \le n} |Q_{ij} - I_{ij}| \sim O(n)$$

A practical way to generate such a $\mathbf{Q}$ is to randomly select n position (i, j) with $i < j$ and let $Q_{ij} \leftarrow \{\pm 1, \pm 2, \ldots\}$ as a small integer, as well as let $Q_{ii} = 1$ for all $1 \le i \le n$. This forms an upper-triangular unimodular matrix and immediately satisfies the above formula.

In practice, we find that this method can guarantee a new basis after reprocessing and will not destroy or reduce the basis quality too much; therefore, this can help the k-tours-BKZ to achieve a stable basis quality again in only few tours (see Figures 1–3). The experiments in the next subsection suggest that we can set a very small k to save time in the reprocessing procedure. There is a broad consensus that "the most significant improvements of BKZ reduction only occurs in the first few rounds" [27], and this is proved in [31]. Some cryptanalyses also use an optimistically small constant c as the number of BKZ tours used to estimate the attack overhead (like $c = 8$ in [32]). Considering the previous literature and the following experiments, we set $k = 8$ for our k-tours-BKZ method.

3.4.2. Average Quality of Lattice Basis During Reprocessing

Even when using the same parameters, DP enumeration will have different runtimes and probabilities of success when used on different bases of a lattice. We expect the basis to have a stable quality that will remain largely unchanged by the reprocessing operation and can be easily simulated or predicted without conducting a real lattice reduction. The very first work aims to define the quality of the lattice basis and study how it changes

during enumeration loops. We chose three indicators to describe the quality of the basis and observe their behavior during reprocessing.

Gram–Schmidt Sum.

For DP-enumeration, its success probability is tightly related to the lengths of the Gram–Schmidt basis vectors $\{\|\mathbf{b}_1^*\|, \ldots, \|\mathbf{b}_n^*\|\}$, and Fukase et al. [18] proposed using *Gram–Schmidt Sum* as a measurement of lattice basis quality and also gave an intuitive and approximate analysis of the strong negative correlation between GSS($\mathbf{B}$) and the efficiency of finding a short lattice vector: the larger the Gram–Schmidt Sum is, the smaller the success probability becomes. The *Gram–Schmidt Sum* is defined as

$$\mathrm{GSS}(\mathbf{B}) \stackrel{def}{=} \sum_{i=1}^{n} \|\mathbf{b}_i^*\|^2 \tag{7}$$

We generated an $n = 120$ dimensional SVP challenge basis at random and showed how the k-tours-BKZ change the GSS($\mathbf{B}$) of the basis during reprocessing. They started to form a BKZ$_\beta$-reduced basis; then, the lattice basis was rerandomized and reprocessed by k-tours-BKZ for every $k = 8$ tours with blocksize β. As shown in Figure 1, the peak that comes up before every $k = 8$ tours of BKZ reduction corresponds to the GSS($\mathbf{B}$) of the rerandomized basis without reprocessing. The peak indicates that the lattice basis is evidently disturbed by our rerandomization method and GSS($\mathbf{B}$) becomes worse, while, in the following k tours, the value of GSS($\mathbf{B}$) quickly returns to a similar state to the well-reduced initial state and hardly changes again. In general, when the lattice basis is iteratively reprocessed, the value of GSS($\mathbf{B}$) only shows mild fluctuations, which implies that the success probability when finding very short lattice vectors is quite stable.

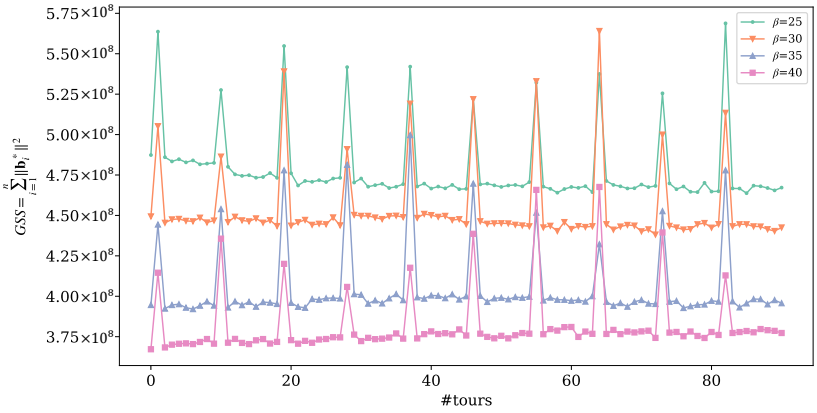

Figure 1. The Evolution of GSS($\mathbf{B}$) During reprocessing, $n = 120$, $k = 8$, $\beta = 25, 30, 35, 40$.

Geometry Series Assumption and GS Slope.

For a well-reduced basis of a random lattice, such as the lattice of the SVP challenge, the Gram–Schmidt orthogonal basis generally has a regular pattern, which is called the *Geometry Series Assumption* (GSA). For an n-dimensional lattice with a BKZ$_\beta$-reduced basis $\mathbf{B}$, GSA means there is a $q \in (0, 1)$, such that

$$\log \|\mathbf{b}_i^*\| = (i-1) \cdot \log q + \log \|\mathbf{b}_1\|, \; i = 1, \ldots, n \tag{8}$$

If GSA holds, the points $\{(i, \log \|\mathbf{b}_i^*\|)\}_{i=1}^n$ form a straight line with a slope of $\log q$. In other words, q defines the "shape" of Gram–Schmidt sequence $\{\|\mathbf{b}_i^*\|\}_{i=1}^n$. In the following, we call $q \in (0, 1)$ the *GS slope*. In the real case of lattice reduction, the points $\{(i, \log \|\mathbf{b}_i^*\|)\}_{i=1}^n$ do not strictly lie on a straight line, and the approximate value of q can be obtained by least square fitting. Generally, a q closer to 1 means that the basis is more reduced and vice versa. Figure 2 shows the evolution of fitted q in reprocessing. The value

of q sharply decreases after each rerandomization and returns in the next few reprocessing tours, showing a stable trend during the iterative rerandomization and BKZ tours.

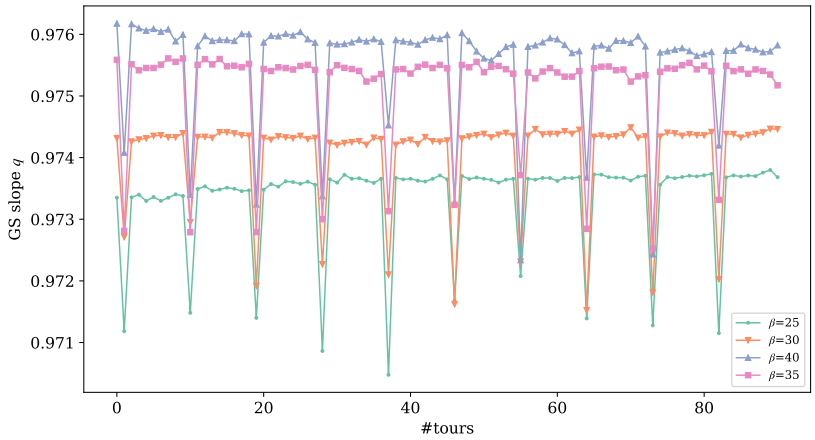

Figure 2. The evolution of GS slope q during reprocessing, $n = 120$, $k = 8$, $\beta = 25, 30, 35, 40$.

Root Hermite Factor.

In the studies of BKZ algorithm, the root Hermite factor δ is used to describe the "ability" of BKZ to find a short lattice vector, which is given as the first basis vector $\mathbf{b}_1$ in the output basis:

$$\delta \overset{\text{def}}{=} \left(\frac{\|\mathbf{b}_1^*\|}{(\text{vol}(\text{L})^{1/n})} \right)^{1/n} \tag{9}$$

Gama and Nguyen [8] pointed out the phenomenon that, for BKZ algorithm, when blocksize parameter $\beta \leqq n$, the root Hermite factor is only affected by the blocksize parameter β but has no relationship with the lattice dimension. Additional observations of root Hermite factor are given by Table 2 of [29].

As in $GSS(\mathbf{B})$ and GS slope q, Figure 3 shows the evolution of the root Hermite factor δ of the reprocessed basis. The peaks arising before reprocessing tours also indicate that the first basis vector $\mathbf{b}_1$ is frequently changed by our rerandomization method, and in the following k tours, the value of δ quickly returns to a similar state to the well-reduced initial state and hardly changes again.

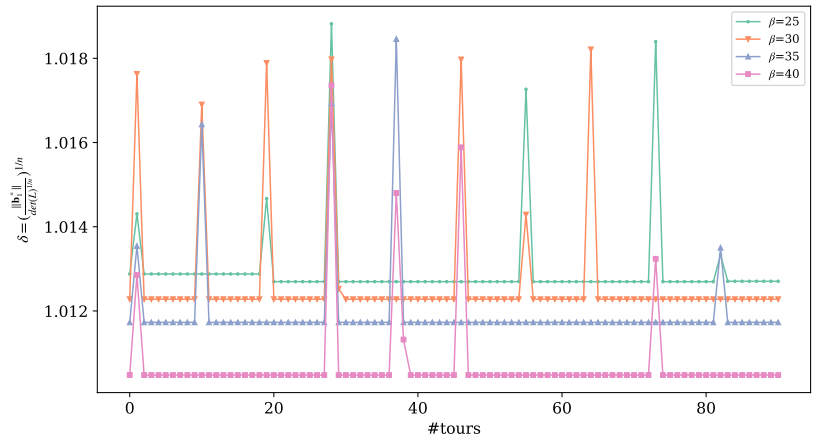

Figure 3. The evolution of δ during reprocessing, $n = 120$, $k = 8$, $\beta = 25, 30, 35, 40$.

Based on the above observations, we believe that, during the reprocessing stage, only a few BKZ reduction tours can stabilize the properties of the lattice bases.

4. Precise Cost Estimation of DP-ENUM

The precise cost estimation of DP enumeration is a great concern and remains an open problem for cryptanalysis. However, there are several obstacles to building a good runtime model that is consistent with the experimental results of a viable dimension and can be extrapolated to a very high dimension.

First, DP enumeration contains many subalgorithms with different structures, such as binary search, cell enumeration, decoding and reprocessing. Although the asymptotic time complexity expression of each part is clearly discussed in Section 3, the real runtime of the DP enumeration still needs to be handled carefully. These subalgorithms involve a variety of arithmetic operations and logic operations, which make it hard to define a universal "basic operation" for all the DP enumeration procedures. To build a runtime model for DP enumeration, our key idea is to use CPU cycles as the basic operation unit, since this can avoid the differences caused by different types of operations and is also easy to count.

Second, the searching space of DP enumeration is a union of many discrete boxes irregularly distributed in $\mathbb{R}^n$. It is quite hard to compute the volume of the pruning set, which directly determines the probability for pruning success. Aono et al. proposed using FILT to compute the volume of its pruning set [23], but this calculation model should be modified to achieve better accuracy, according to the rectification of the original randomness assumption we made in Section 3.

According to Algorithm 1, the cost of each loop in DP enumeration can be divided into four parts:

- T_{bin}: Use binary search to determine cell enumeration parameter r (Algorithm 3)
- T_{cell}: Enumerate all the tags of candidate cells (Algorithm 2)
- T_{decode}: Decode a tag and check the length of the corresponding lattice vector (Algorithm 4)
- T_{repro}: If there is no valid solution to SVP, rerandomize the lattice basis and reprocess it by k-tours-BKZ algorithm (Algorithm 5)

Denote the success probability when finding a lattice vector shorter than R in a single loop of DP enumeration by p_{succ} and assume that p_{succ} is stable during rerandomizing; then, the expected number of loops is about $\frac{1}{p_{succ}}$ according to geometric distribution, and the total runtime T_{total} of DP enumeration can be estimated by

$$T_{total} = T_{pre} + \frac{T_{repro} + T_{bin} + T_{cell} + M \cdot T_{decode}}{p_{succ}} \tag{10}$$

We assume that the preprocessing time for T_{pre}, which denotes the time for a full-tour BKZ$_\beta$ reduction on the initial lattice basis, is far less than the time spent in the main iteration and can be ignored when $\beta \ll n$. In this section, our aim is to determine the explicit expression of T_{repro}, T_{bin}, T_{cell} and T_{decode}, as well as provide an accurate estimation of p_{succ}.

For all the experiments in this paper, the computing platform is a server with Intel Xeon E5-2620 CPUs, with eight physical cores running at 2.10 GHz, and 64GB RAM. To obtain accurate CPU cycles for the DP enumeration algorithm, we fixed the CPU basic frequency and set the CPU affinity, and all the time data for fitting were obtained from our single-thread implementation.

4.1. Simulation of Lattice Basis

Some parts in the total time cost model Equation (10) are hugely dependent on the quality of the basis: more precisely, the vector lengths of Gram–Schmidt orthogonal basis $\{\|\mathbf{b}_i^*\|\}_{i=1}^n$. Based on the reprocessing method and the stability analysis in Section 3.4, we can reasonably assume the Gram–Schmidt sequence $\{\|\mathbf{b}_i^*\|\}_{i=1}^n$ will not severely change in the loops of DP enumeration. Then, the issue is to simulate an "average" GS sequence for a BKZ$_\beta$-reduced basis.

BKZ simulator [29,30] is a universal method, especially when β is quite large, since the Gaussian heuristic generally holds in β-dimensional blocks where $\beta \gtrsim 45$. In this case, the probabilistic BKZ simulator proposed by Bai et al. [30] is an appropriate model due to its reliable depiction of the "head concavity" phenomenon. Based on Algorithm 3 in [30], we provide this BKZ simulator in C++ as a component of our implementation.

However, the BKZ simulator does not work well for small- and medium-sized β ($\beta < 45$), because the keystone of the BKZ simulator is to estimate the shortest vector length in a β-dimensional sublattice (block) $\mathcal{L}_{[i,j]}$ by computing $GH(\mathcal{L}_{[i,j]})$, which will decrease in accuracy when $\beta < 45$. This is rarely the case with asymptotic cryptanalysis. However, this is a prevailing case in preprocessing and also needs investigation; hence, we have to propose a method that considers this case and fills the vacancy in GS sequence simulation.

If we combine

$$\mathrm{vol}(\mathcal{L}) = \prod_{i=1}^{n} \|\mathbf{b}_i^*\|$$

and Equation (8), then we have

$$n \cdot \log \|\mathbf{b}_1\| + \frac{n(n-1)}{2} \log q = \log(\mathrm{vol}(\mathcal{L})) \tag{11}$$

Using Equation (11), we can approximately calculate the whole Gram–Schmidt sequence $\{\|\mathbf{b}_i^*\|\}_{i=1}^{n}$ if GSA holds and one of $\|\mathbf{b}_1\|$ or q is known. Here, we prefer to use the GS slope q rather than the value of $\|\mathbf{b}_1\|$, since it contains more information on the Gram–Schmidt orthogonal basis. Additionally, using $\|\mathbf{b}_1\| = \delta^n \mathrm{vol}(\mathcal{L})^{1/n}$ to recover the Gram–Schmidt sequence might lead to an overly optimistic estimation, since the "head concavity" phenomenon [30] indicates that the $\|\mathbf{b}_1\|$ might significantly deviate from the prediction given by GSA. Then, the feasible approach is to give an average estimation of q for a given lattice and BKZ parameter β.

We find that the GS slope has similar properties to the root Hermite factor: when $\beta \ll n$, the GS slope of a reduced basis is mostly related to its blocksize β but not its lattice dimension. For each parameter set $(n, \beta) \in \{120, \ldots, 180\} \times \{11, 13, \ldots, 45\}$, we generate 50 random SVP challenge instances and apply BKZ_β on the n-dimensional lattice basis to verify this phenomenon. Then, we use the least square method to fit the $\log q$ of reduced bases. Figure 4 shows the relationship between q and lattice dimension n, indicating that q is hardly dependent on lattice dimension n and mostly varies with β.

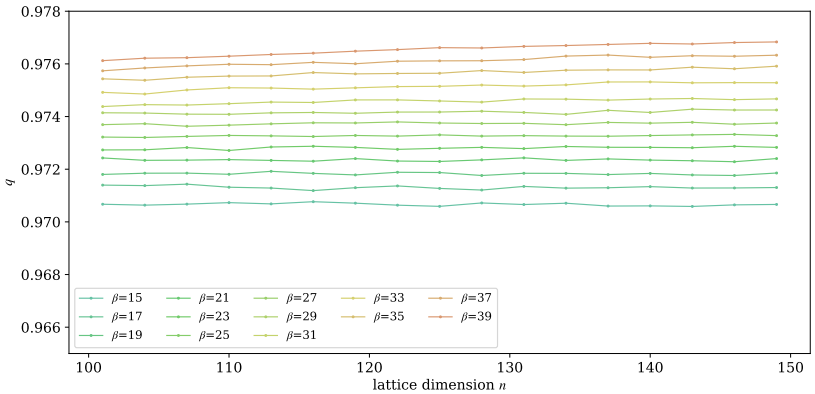

Figure 4. The relation between q and lattice dimension n.

Figure 5 illustrates the positive correlation of q and β, which is consistent with the idea that a larger blocksize β makes the lattice basis better, and the GS slope is milder, which means that $q < 1$ is closer to 1.

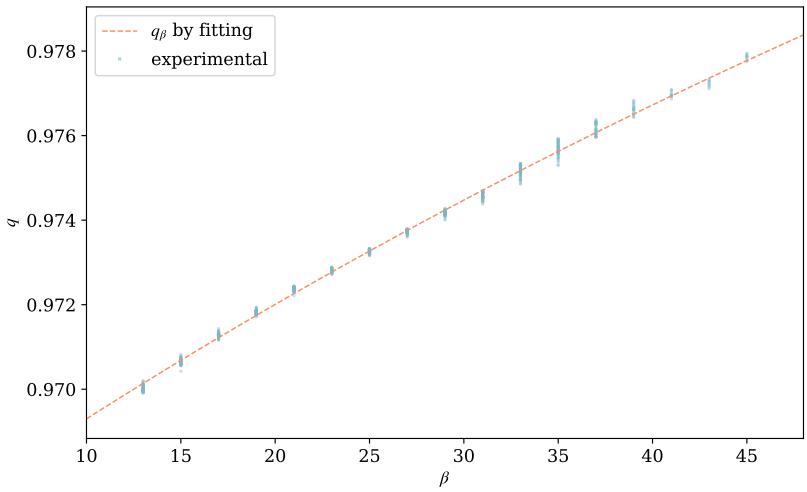

Figure 5. The relation between q and BKZ blocksize β.

Table 1 gives an estimated value of q_β.

Table 1. The estimated GS slope q of BKZ$_\beta$-reduced lattice basis.

β	11	13	15	17	19	21	23	25	27
q	0.9698	0.9703	0.9708	0.9713	0.9718	0.9723	0.9727	0.9733	0.9737
β	29	31	33	35	37	39	41	43	45
q	0.9742	0.9746	0.9751	0.9755	0.9759	0.9763	0.9767	0.9772	0.9776

By using the empirical data of q_β, we can generate a virtual GS sequence $\{B_1, B_2, \ldots, B_n\}$ to simulate the real behavior of the Gram–Schmidt orthogonal basis of a BKZ$_\beta$-reduced lattice basis by solving the following equations:

$$\begin{cases} \mathrm{vol}(\mathcal{L}) = \prod_{i=1}^{n} B_i \\ n \cdot \log B_1 + \frac{n(n-1)}{2} \log q_\beta = \log(\mathrm{vol}(\mathcal{L})) \\ \log B_i = (i-1) \cdot \log q_\beta + \log B_1,\ i = 1, \ldots, n \end{cases} \tag{12}$$

Remark 3. *All the explicit values of q_β for $\beta \leq 45$ are given in our open-source implementation. Note that the method proposed above only takes effect when $\beta \leq 45$, while we still give an extrapolation as an alternative reference. Since $q < 1$ holds for all $\beta > 0$ and q_β is an increasing function of β, which implies a trend $q \xrightarrow{\beta \to \infty} 1$, we heuristically assume this function has a form of $q = 1 - \exp(-a\beta + b)$; then, the fitting result of Table 1 is*

$$q_\beta = 1 - \exp(-0.0092200\beta - 3.3919) \tag{13}$$

The fitting curve is also illustrated in Figure 5.

4.2. Cost of Subalgorithms

Cost of Binary Search and Cell Enumeration.

For the cell enumeration algorithm (Algorithm 2), the asymptotic time complexity is $O((2n-1) \cdot M)$. We take $n = 60, \ldots, 160$, $M = 1.0 \times 10^4, 1.5 \times 10^4, \ldots, 1.0 \times 10^5$ and, for each parameter set (n, M), we generate 100 SVP lattices at random. The fitting result is

$$T_{cell} \approx 2.4339nM + 108.74M - 17455n + 1334139 \tag{14}$$

For the binary search algorithm (Algorithm 3), Theorem 1 indicates that the asymptotic time complexity has an asymptotic upperbound $\left(\log n + \log \frac{1}{\epsilon} + n \log(n\det(L)^{\frac{2}{n}})\right) \times (2n-1)M$. To simplify the fitting function and retain accuracy, we only take the dominant term of the complete expansion, which is $n \log\left(n\det(L)^{\frac{2}{n}}\right) \cdot (2n-1)M$. For the SVP challenge lattice, we have $\mathrm{vol}(\mathcal{L}) \sim 2^{10n}$. Then, the fitting function of T_{bin} is

$$T_{bin} \approx 0.11341Mn^2 + 13.155Mn\log n + 265.65M - 84679n + 15455380 \tag{15}$$

Both fitting functions obtained by the least square method have a coefficient of determination (R-squared) larger than 0.95.

Cost of Decoding.

To decode one tag by running Algorithm 4, the number of times to enter the for loop is $O(n)$, and in each loop, this performs $O(n)$ arithmetic operations. Therefore, T_{decode} can be regarded as a quadratic function of n. We take $n = 60, \ldots, 160$ and fix $M = 1.0 \times 10^5$, and we generate 100 SVP lattices at random for each n. The expected runtime T_{decode} of decoding algorithm is fitted by

$$T_{decode} = 0.39045n^2 + 167.06n - 4350.4 \tag{16}$$

Figure 6 shows that the fitting curve of T_{decode} is almost strictly consistent with the experimental data. The fitting function also has a coefficient of determination (R-squared) larger than 0.95.

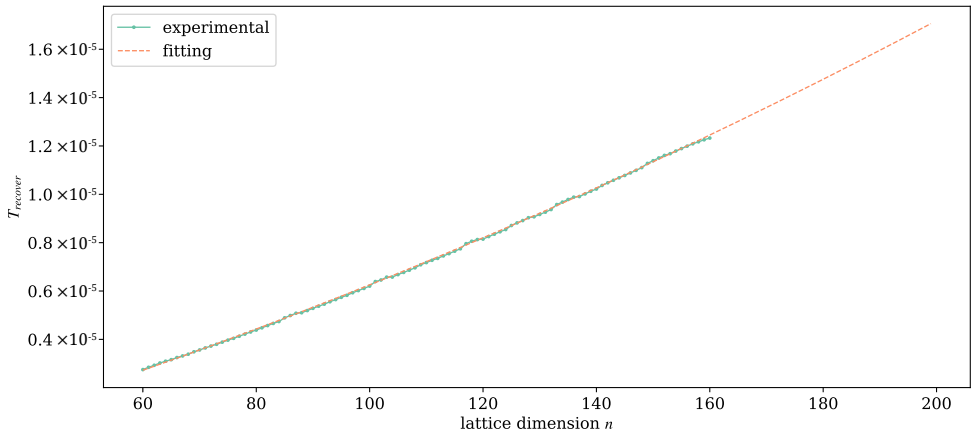

Figure 6. The relation between T_{decode} and dimension n.

Cost of Reprocessing.

The cost of the k-tours-BKZ algorithm is a little complicated, since it iteratively calls an $O(\beta)$-dimensional SVP oracle. Our implementation of k-tours-BKZ is based on the BKZ 2.0 algorithm in fplll library [13]. In one tour of BKZ_β, the total runtime is composed of the processing time on $n-1$ blocks. For each block $\mathcal{L}[i,j] = \mathcal{L}(\mathbf{b}_i, \ldots, \mathbf{b}_j)$ with $i = 1, \ldots, n-1$ and $j = \min(n, i+\beta-1)$, the main steps are classical enumeration and LLL reduction for updating:

$$\mathrm{BlockCost}(i,j) = \mathrm{BlockProcess}(j,n,\log A) + C_{node} \cdot \mathrm{EnumCost}(i,j) \tag{17}$$

Then, the cost of k-tours-BKZ can be estimated by

$$\mathrm{BKZCost}(L) = k \cdot \sum_{i=1}^{n-1} \mathrm{BlockCost}(i,j),\ j = \min(n, i+\beta-1) \tag{18}$$

In Equation (17), BlockProcess$(j, n, \log A)$ is the cost of updating basis (Algorithm 5, line 12). The asymptotic time complexity of this part is $O(j^3 m \log A)$, which is mainly donated by LLL reduction [33], where $A \lesssim 2^{10n}$ for the SVP challenge lattice. When β is small, the cost of updating cannot be ignored. For the cost of classical pruned enumeration, C_{node} is the CPU cycles for processing a single node in the enumeration tree, which is said to be $C_{node} \approx 200$ [34]; EnumCost(i, j) is the total amount of nodes that needs to be traversed to find a short vector on $\mathcal{L}[i, j]$.

Let $n = 60, \ldots, 150$, $\beta = 11, 13, \ldots, 43$ and $k = 8$, we record the cost of each stage (including runtime and the number of nodes) of k-tours-BKZ$_\beta$ on 50 random lattices. The least squares fitting shows $C_{node} \approx 205.45$, and

$$\text{BlockProcess}(j, n, \log A) \approx 0.000904381 \times j^3 n^2 + 28752188 \tag{19}$$

The remaining part is EnumCost(i, j), which is the number of nodes of enumeration on block $\mathcal{L}[i, j]$. For a full enumeration of $\mathcal{L}[i, j]$, the total number of nodes can easily be derived from the Gaussian heuristic, which can be considered a baseline enumeration cost:

$$\text{FullEnumCost}(i, j) = \frac{1}{2} \sum_{k=1}^{j-i+1} \frac{V_k(\|\mathbf{b}_i^*\|)}{\prod_{\ell=j-k+1}^{j} \|\mathbf{b}_\ell^*\|} \tag{20}$$

where $V_k(R)$ denotes the volume of a k-dimensional ball with radius R.

However, in our implementation of k-tours-BKZ, the SVP oracle uses extreme pruning and heuristic enumeration radius $c = \min(1.1\text{GH}(\mathcal{L}[i, j]), \|\mathbf{b}_i^*\|)$ for acceleration. We assume that, for classical enumeration on a $\beta' = j - i + 1$ dimensional block $\mathcal{L}[i, j]$, these methods offer a speedup ratio of $r_{\beta'}$ in total, and $r_{\beta'}$ is independent with the block index i and lattice dimension n. The key point is obtaining an explicit expression of $r_{\beta'}$. (An alternative (lowerbound) estimation of enumeration cost is provided by Chen and Nguyen [29]. The coefficients of their model are given in LWE estimator [35]. However, their model is more suitable for when β is very high and is not very precise when the blocksize is small).

$$r_{\beta'} = \frac{\text{FullEnumCost}_{\beta'}}{\text{ExtremeEnumCost}_{\beta'}} \tag{21}$$

The value of FullEnumCost$_{\beta'}$ can be calculated by Equation (20) with GS sequence $\{\|\mathbf{b}_i^*\|\}_{i=1}^n$, and the actual number of enumeration nodes ExtremeEnumCost$_{\beta'}$ is obtained from experiments. We recorded the number of enumeration nodes ExtremeEnumCost$_{\beta'}$ to calculate the speedup ratio data. To fit $r_{\beta'}$, we run k-tours-BKZ$_\beta$ on BKZ$_\beta$-reduced bases with $n = 60, \ldots, 150$ and $\beta = 11, 13, \ldots, 43$; then, all the data of the same blocksize were gathered and averaged. It is well known that extreme pruning can offer an exponential speedup [12], and tightening the radius also leads to a superexponential speedup. We assume $r_{\beta'} \sim \exp O(\beta' \log \beta')$, and by fitting, we can obtain

$$\log r_{\beta'} = 0.35461\beta' \log \beta' - 1.5331\beta' + 4.8982 \log \beta' - 2.9084 \tag{22}$$

Figure 7 shows the fitting results and the value of $r_{\beta'}$ in experiments, reflecting that the assumptions we made are reasonable.

To predict EnumCost(i, j) without any information on a specific lattice basis, the GS sequence contained Equation (21) should be replaced by the simulated values $\{B_1, B_2, \ldots, B_n\}$ derived by equations set using the (12) or BKZ simulator.

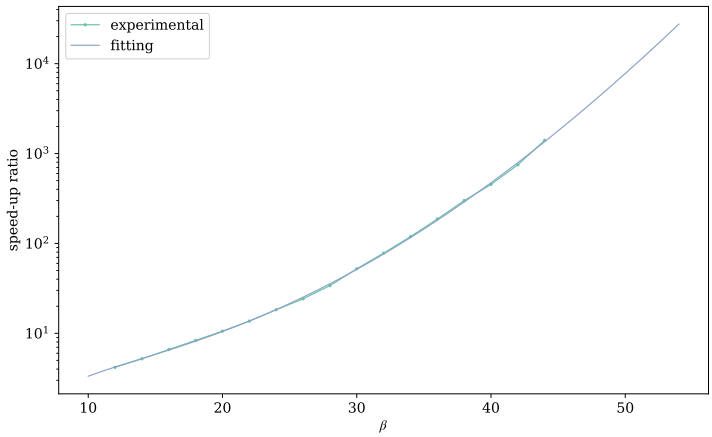

Figure 7. Speedup ratio of extreme pruning.

Algorithm 6 gives the procedures of calculating T_{repro}, i.e., the cost of reprocessing.

Algorithm 6 Calculating T_{repro}

Require: β, lattice dimension n, k, $\mathrm{vol}(\mathcal{L})$
Ensure: The running time T_{repro} of reprocessing with k-tours-BKZ$_\beta$

if $\beta < 45$ **then**
  Find q corresponding to β // See Table 1 or Equation (13)
  $\log B_1 \leftarrow \frac{1}{n}\left(\log(\mathrm{vol}(\mathcal{L})) - \frac{1}{2}n(n-1)\right)$ // Equation (11)
  $B_1 \leftarrow \exp(\log B_1)$
  for $i = 2$ **to** n **do**
    $\log B_i \leftarrow (i-1)\cdot \log q + \log B_1$ // Equation (8)
    $B_i \leftarrow \exp(\log B_i)$
  end for
else
  $\{B_i\}_{i=1}^{n} \leftarrow$BKZSim() // Use BKZ simulator in [30], Algorithm 3
end if
$i \leftarrow 0$
$Cost \leftarrow 0$
while $i = 1$ **to** $n-1$ **do**
  $\beta' = \min(\beta, n-i+1)$
  $r_{\beta'} \leftarrow \exp\left(0.35461\beta' \log \beta' - 1.5331\beta' + 4.8982 \log \beta' - 2.9084\right)$ // Equation (22)
  $Cost = Cost + \frac{1}{r_{\beta'}}\cdot$FullENUMCost$(B_i, \ldots, B_{i+\beta'-1})$ + BlockProcess$(i+\beta'-1, n)$ // by replacing $\|\mathbf{b}_i^*\|$ with B_i in Equation (20)
end while
return $k \cdot C_{node} \cdot Cost$

4.3. Success Probability

Under a Gaussian heuristic, the success probability of pruned enumeration can be directly reduced to computing the volume of the pruning set. For discrete pruning, the shape of the pruning set has always been considered a union of "ball-box intersections", which is not easy to compute. Aono et al. [23] proposed an efficient numerical method based on fast inverse Laplace transform (FILT) to compute the volume of a single "ball-box intersection" $\mathcal{C}(\mathbf{t}) \cup \mathrm{Ball}_n(R)$ and used stratified sampling to deduce the total volume of the union.

However, the imperfections in original randomness assumption (Assumption 2) lead to reduced accuracy of the success probability model for discrete pruning. For two cells with tag $\mathbf{t} = [t_1, \ldots, t_k \neq 0, 0, \ldots, 0]$ and $\mathbf{t}' = [t_1, \ldots, t_k + 1 \neq 0, 0, \ldots, 0]$, if t_k is odd, i.e., $\mathbf{t}$ is odd-ended and $\mathbf{t}'$ is the corresponding even-ended tags, they will have different success

probabilities according to the model given by [23]. However, the lattice vectors contained in $\mathcal{C}(\mathbf{t})$ and $\mathcal{C}(\mathbf{t}')$ have exactly the same length.

Figure 8 illustrates the gap at a larger scale. For the parameter settings $n = 60, \ldots, 84$, $M = 50{,}000$ and $\beta = 20, 30$, we used 30 BKZ$_\beta$-reduced n-dimensional lattice bases to compute the average value of theoretical success probability $p_{succ,odd}$ of M odd-ended cells enumerated by Algorithm 2, as well as compute $p_{succ,even}$ of their corresponding even-ended cells, both using the method provided by [23]. Then, we ran a complete DP enumeration on each lattice basis using the same parameters and recorded the number of iteration rounds. Figure 8 shows that the actual number of rounds of DP enumeration is in the apparent gap between expectation value $1/p_{succ,odd}$ and $1/p_{succ,odd}$, which were estimated using odd-ended and even-ended cells, respectively.

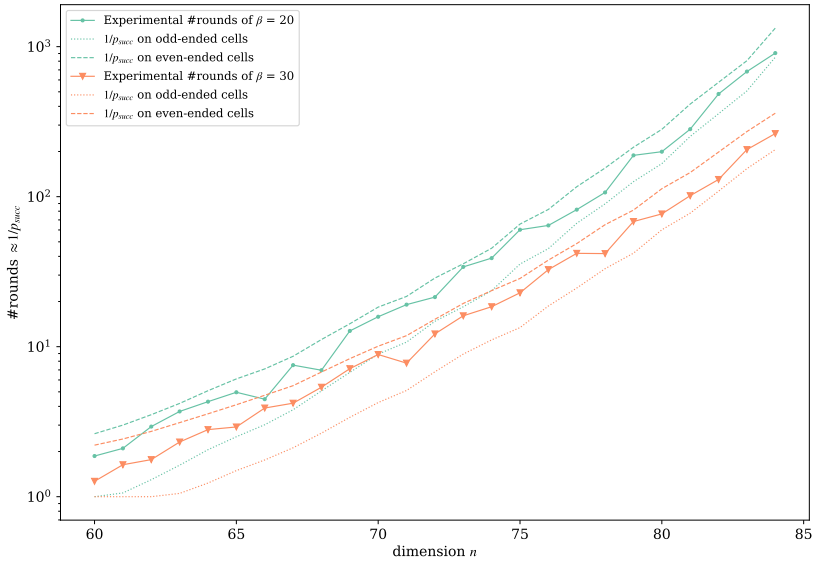

Figure 8. The difference of p_{succ} estimated with odd-ended cells and even-ended cells.

This phenomenon calls for a proper rectification of the success probability model. As a matter of fact, in Section 3.1, Proposition 1 and the rectified Assumption 3 indicate that lattice point is actually randomly distributed in an hyperplane contained in $\mathcal{C}(\mathbf{t}) \cup \text{Ball}_n(R)$, which can be described by the assumption below:

Assumption 4. *Given lattice basis* $\mathbf{B}$ *and its orthogonal basis* $\mathbf{B}^*$*, for a tag vector* $\mathbf{t} = [t_1, \ldots, t_k \neq 0, 0, \ldots, 0]$*, the lattice vector of* $\mathcal{C}(\mathbf{t})$ *can be considered to be uniformly distributed over* $\mathcal{C}'(\mathbf{t}) \subset \mathcal{C}(\mathbf{t})$*, where*

$$\mathcal{C}'(\mathbf{t}) \overset{def}{=} \left\{ \sum_{i=1}^{n} x_i \mathbf{b}_i^*,\ x_i \in \mathbb{R} \text{ and } \begin{cases} x_i \in (-t_i + 1/2, -t_i/2] \cup (t_i/2, t_i + 1/2] \text{ for } i < k \\ x_k = u_k \text{ as defined in Equation (3)} \\ x_{k+1} = \ldots x_n = 0 \end{cases} \right\} \tag{23}$$

This assumption gives a more precise distribution of the lattice vector in the cell. In fact, $\mathcal{C}'(\mathbf{t})$ is the union of a 2^{k-1} $k-1$-dimensional "box", which is formally denoted by

$$\mathcal{C}_{k-1}(\mathbf{t}) \overset{def}{=} \left\{ \sum_{i=1}^{k-1} x_i \mathbf{b}_i^*, x_i \in \left(-\frac{t_i + 1}{2}, -\frac{t_i}{2} \right] \cup \left(\frac{t_i}{2}, \frac{t_i + 1}{2} \right] \right\}$$

Based on Proposition 1 and the new assumption of lattice vector distribution, we redefine the success probability of DP enumeration on a single cell. For a $\mathcal{C}(\mathbf{t})$ with $\mathbf{t} = [t_1, \ldots, t_k \neq 0, 0, \ldots, 0]$, denoting the lattice vector $\mathbf{v} \in \mathcal{C}(\mathbf{t})$ by $\mathbf{v} = \sum_{j=1}^{n} u_j \mathbf{b}_j^*$, the probability that $\|\mathbf{v}\| \leq R$ is defined by

$$\begin{aligned} p_{succ}(\mathbf{t}) &\overset{def}{=} \underset{\mathbf{v}\leftarrow\mathcal{C}'(\mathbf{t})}{\text{Prob}}\left(\|\mathbf{v}\|^2 \le R^2\right) \\ &= \underset{\mathbf{v}\leftarrow\mathcal{C}'(\mathbf{t})}{\text{Prob}}\left(\sum_{i=1}^{k-1} u_i^2\|\mathbf{b}_i^*\|^2 < (R^2 - u_k^2\|\mathbf{b}_k^*\|^2)\right) \\ &= \frac{\text{vol}\left(Ball_{k-1}(\sqrt{R^2 - u_k^2\|\mathbf{b}_k^*\|^2}) \cap \mathcal{C}_{k-1}(\mathbf{t})\right)}{\text{vol}(\mathcal{C}_{k-1}(\mathbf{t}))} \\ &= \frac{\text{vol}\left(Ball_{k-1}(\sqrt{R^2 - u_k^2\|\mathbf{b}_k^*\|^2}) \cap \mathcal{C}_{k-1}(\mathbf{t})\right)}{\prod_{i=1}^{k-1}\|\mathbf{b}_i^*\|^2} \end{aligned} \tag{24}$$

Let $R' = \sqrt{R^2 - u_k^2\|\mathbf{b}_k^*\|^2}$, $\alpha_i = \frac{t_i}{2R'}\|\mathbf{b}_i^*\|$ and $\beta_i = \frac{t_i+1}{2R'}\|\mathbf{b}_i^*\|$; then, the numerator part in Equation (24) can be written as

$$\begin{aligned} &\text{vol}\left(Ball_{k-1}(R') \cap \mathcal{C}_{k-1}(\mathbf{t})\right) \\ =&2^{k-1}\cdot R'^{k-1}\cdot\prod_{i=1}^{k-1}(\beta_i - \alpha_i)\cdot \underset{(x_1,\ldots,x_{k-1})\leftarrow\prod_{i=1}^{k-1}[\alpha_i,\beta_i]}{Pr}\left\{\sum_{i=1}^{k-1} x_i^2 \le 1\right\} \end{aligned} \tag{25}$$

Then, the calculation of $p_{succ}(\mathbf{t})$ is reduced to computing the sum distribution of $k-1$ independent and identically distributed variables $x_1^2, \ldots, x_{k-1}^2$, which can be approximated by the FILT method combined with Euler transformation. The details of these methods are given in Appendix B.

For a set of tags $\mathcal{U}$, which is the output of cellENUM (Algorithm 2), the total success probability of finding a short lattice vector among $\mathcal{U}$ is

$$p_{succ} \approx \min\left(1, \sum_{\mathbf{t}\in\mathcal{U}} p_{succ}(\mathbf{t})\right) \tag{26}$$

To extrapolate the probability model to higher-dimensional SVP instances without performing any time-consuming computation of real lattice reduction, the concrete value of the GS sequence involved in the calculation of p_{succ} should be replaced by the simulated GS sequence $\{B_1, B_2, \ldots, B_n\}$.

Figure 9 verifies the accuracy of the rectified success probability model (Equations (24) and (26)). We take the SVP instances with $n = 60, \ldots, 84$, $\beta = 20, 30$ and M = 50,000 as examples, and we run the DP enumeration algorithm to solve the SVP challenge on each SVP instance to record the total iteration rounds. The experiment was repeated 30 times on each parameter set to obtain the average value. The dashed line shows the expected iteration rounds $1/p_{succ}$ calculated using the original $\{\|\mathbf{b}_i^*\|\}_{i=1}^n$ of the real reduced basis, and the dotted line was only calculated with the simulated GS sequence $\{B_i\}_{i=1}^n$. The results illustrate that the rectified model gives a more precise estimation of success probability than the original method provided in [23].

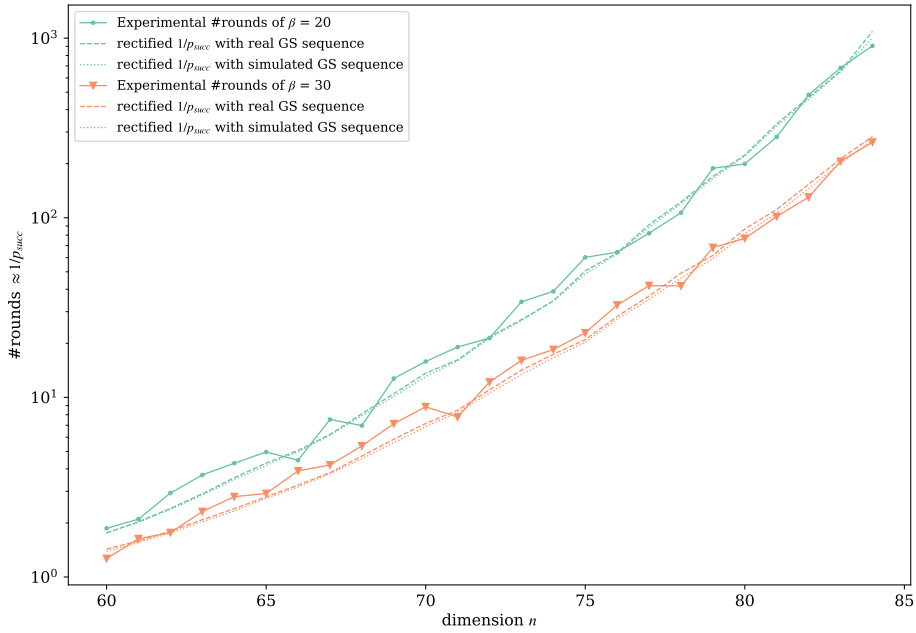

Figure 9. Verification of the rectified success probability model.

4.4. Simulator for DP Enumeration

Based on all the works in this section, the runtime of DP enumeration can be estimated by Algorithm 7, shown below. This simulator only needs minimal information for a lattice $\mathcal{L}$.

Algorithm 7 DP-simulator

Require: Lattice dimension and volume $n, \mathrm{vol}(\mathcal{L}), k, \beta, M$, target length R of SVP
Ensure: Expected runtime (CPU cycles) to find $\mathbf{v} \in \mathcal{L}$ such that $\|\mathbf{v}\| \leq R$ by DP enumeration

1: **if** $\beta < 45$ **then**
2: Generate the simulated GS sequence $B_1, \ldots, B_n$ by solving Equation (12)
3: **else**
4: Generate the simulated GS sequence $B_1, \ldots, B_n$ by BKZ simulator
5: **end if**
6: Calculate r_β by Equation (22)
7: Calculate T_{repro} by calling Algorithm 6 with parameters$(\beta, n, k, \mathrm{vol}(\mathcal{L}))$ as input
8: Calculate T_{cell} by Equation (14) with M, n
9: Calculate T_{bin} by Equation (15) with M, n
10: Calculate T_{decode} by Equation (16) with n
11: Call Algorithms 2 and 3 with $B_1, \ldots, B_n$ as the GS sequence and output the M tags with minimal value of $f(\mathbf{t})$
12: Calculate the total success probability p_{succ} on the M tags, with GS sequence $B_1, \ldots, B_n$
return $\frac{T_{repro}+T_{bin}+T_{cell}+M \cdot T_{decode}}{p_{succ}}$

Remark 4. *The simulating method of GS sequence (line 1) only works for lattice bases that meet GSA. For those lattices that lack good "randomness" and do not satisfy GSA, one has to use a real BKZ$_\beta$ reduction algorithm on several lattice bases and compute an averaged GS sequence $B_1, \ldots, B_n$ as a good simulation of $\{\|\mathbf{b}_i^*\|\}_{i=1}^n$.*

5. The Optimal Parameters for DP-ENUM

To solve a certain SVP instance, the parameters of DP enumeration that need to be manually determined are as follows: β of BKZ reduction algorithm, k of preprocessing and M of cell enumeration.

It should be noted that k could be a fixed constant. There is no need to set a very large k because of the "diminishing returns" of lattice reduction, which means the improvement in basis quality would slow down with an increase in k. We heuristically set $k = 8$ for SVP instances with $n \leq 200$, which is also roughly consistent with the observation of [32] (see Section 2.5 of [32]). Then, only β and M should be determined with restrictions $0 < \beta \leq n$ and $M > 0$. The two parameters should minimize the total cost of DP enumeration, i.e., the expression value of (10). This value is calculated by Algorithm 7 and can barely be represented by a differentiable function. The Nelder–Mead simplex method is an effective method to solve this type of optimization problem. Since there are only two independent variables, it is reasonable to believe that the Nelder–Mead method can quickly converge to the optimal solution.

Algorithm 8 gives the optimal values of β, M for a certain SVP instance based on the standard version of the Nelder–Mead method.

Table 2 gives some concrete values of optimal parameter sets for solving medium-size SVP challenges ($R = 1.05\mathrm{GH}(\mathcal{L})$) and the corresponding estimation of running time. For the medium-size SVP challenges, the optimal parameter set basically follows $M \sim 10^5$ and $\beta < n/2$. Neither of them increase very rapidly with the growth of n.

Table 2. Optimal parameters of DP-ENUM for solving SVP challenge.

n	β	M	Expected Time (CPU Cycles)
80	39	65,000	9.08×10^{10}
82	42	110,000	1.22×10^{11}
84	42	95,000	1.96×10^{11}
86	42	175,000	2.92×10^{11}
88	42	155,000	4.83×10^{11}
90	42	100,000	8.90×10^{11}
92	39	150,000	1.52×10^{12}
94	42	150,000	2.09×10^{12}
96	39	170,000	6.88×10^{12}
98	42	180,000	1.08×10^{13}
100	42	145,000	2.15×10^{13}
102	39	195,000	4.16×10^{13}
104	39	190,000	1.11×10^{14}
106	42	130,000	1.34×10^{14}
108	42	175,000	4.24×10^{14}
110	44	190,000	1.23×10^{15}

Algorithm 8 Finding optimal parameters for DP enumeration

Require: lattice dimension n, lattice volume $\mathrm{vol}(\mathcal{L})$ and the target vector length R of SVP
Ensure: (β, M) that minimizes the output of DP-simulator (n, β, M, R)

$S(\beta, M) :=$ DP-simulator $(n, \beta, M, R) + P(\beta, M)$ // $P(\beta, M)$ is a penalty function to avoid parameters exceeding the feasible region, i.e., $\beta > n$ or $M < 0$.
$N \leftarrow 2$ // 2 independent variables
Select initial points $\mathbf{x}_1 = [\beta_1, M_1], \ldots, \mathbf{x}_{N+1} = [\beta_{N+1}, M_{N+1}]$ at random
while true do
  reorder the $N+1$ points, such that $S(\mathbf{x}_1) < \ldots < S(\mathbf{x}_{N+1})$
  $y_1 \leftarrow S(\mathbf{x}_1), \ldots, y_{N+1} \leftarrow S(\mathbf{x}_{N+1})$
  if $|\beta_1 - \beta_{N+1}| < 2$ **and** $|M_1 - M_{N+1}| < 1000$ **then**
    break;
  end if
  $\mathbf{x}_m \leftarrow \frac{1}{N} \sum_{i=0}^{N} \mathbf{x}_i$ // calculate the centroid (midpoint)
  $\mathbf{x}_r \leftarrow 2\mathbf{x}_m - \mathbf{x}_{N+1}$ // reflection
  $y_r \leftarrow S(\mathbf{x}_r)$
  if $y_1 \leq y_r < y_N$ **then**
    $\mathbf{x}_{N+1} \leftarrow \mathbf{x}_r$
    continue;
  else if [**then**expansion] $y_r < y_1$
    $\mathbf{x}_e \leftarrow \mathbf{x}_m + 2(\mathbf{x}_r - \mathbf{x}_m)$
    if $S(\mathbf{x}_e) < y_r$ **then**
      $\mathbf{x}_{N+1} \leftarrow \mathbf{x}_e$
    else
      $\mathbf{x}_{N+1} \leftarrow \mathbf{x}_r$
    end if
  else if [**then**contraction] $y_N \leq y_r < y_{N+1}$
    $\mathbf{x}_c \leftarrow \mathbf{x}_m + (\mathbf{x}_r - \mathbf{x}_m)/2$
    if $S(\mathbf{x}_c) < y_r$ **then**
      $\mathbf{x}_{N+1} \leftarrow \mathbf{x}_c$
      continue;
    end if
  else
    $\mathbf{x}_c \leftarrow \mathbf{x}_m + (\mathbf{x}_{N+1} - \mathbf{x}_m)/2$
    if $S(\mathbf{x}_c) < y_r$ **then**
      $\mathbf{x}_{N+1} \leftarrow \mathbf{x}_c$
      continue;
    end if
  end if
  for [**do**shrink] $i = 2$ **to** $N+1$
    $\mathbf{x}_i \leftarrow \mathbf{x}_1 + (\mathbf{x}_i - \mathbf{x}_1)/2$
  end for
end while return The optimal parameters $\mathbf{x}_{min} \leftarrow \mathbf{x}_1 = [\beta_1, M_1]$ and the corresponding cost estimation $S(\mathbf{x}_{min})$

6. Experiments and Analysis

We compared the performance of our optimized DP enumeration with different SVP solvers, including the polynomial-space extreme pruned enumeration and exponential-space sieving. For each n, the experiments were repeated on 40 different SVP challenge instances. We ran our optimized DP enumeration with parameters given by Algorithm 8. The lattice dimension n ranged from 60 to 110, and the time cost predicted by DP simulator is also provided. The extreme pruned enumeration was implemented by fplll library [13] with the default pruning function up to $n = 90$. The data of sieving were implemented by G6K library [36] with the dimension-for-free method, i.e., G6K's `WorkOut` $(s = 0, f^+ = 1)$ from $n = 60$ to 110. Figure 10 illustrates the prediction of the DP simulator, as well as the experimental results of our optimized DP enumeration, extreme pruning and G6K sieving.

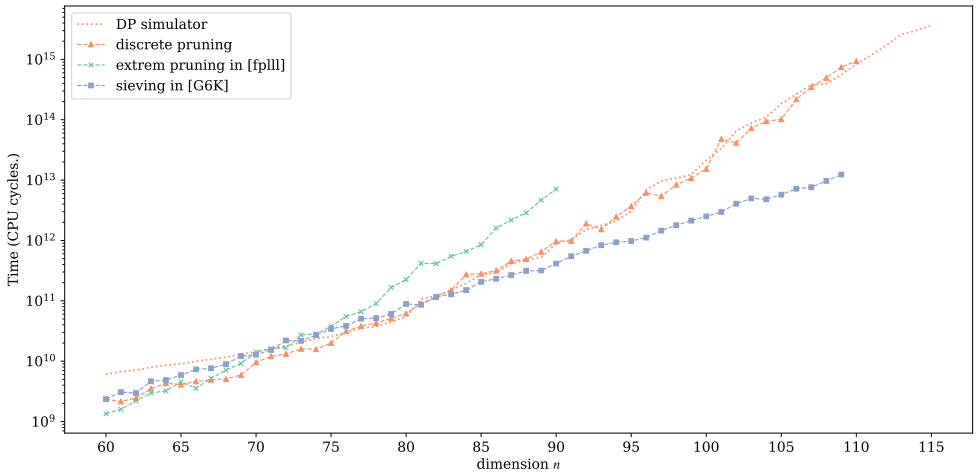

Figure 10. The performance of optimized discrete pruning and other SVP solvers.

The experiments confirm the accuracy of our cost model proposed in Section 4. The prediction (orange dotted line) is quite consistent with the actual performance of DP enumeration (orange broken line). For $n \lesssim 75$, the DP enumeration algorithm sometimes finds a solution before the first round ends; therefore, the actual running time is slightly smaller than the simulated time. However, $n > 80$ shows that our implementation of DP enumeration (with optimal parameter set) coincides with the DP simulator very well.

Compared with extreme pruning, Figure 10 shows that when $n \gtrsim 67$, the optimized DP enumeration has a shorter runtime than the state-of-the-art extreme pruning enumeration. As for sieving, Albrecht et al. [37] has observed that the state-of-the-art sieving algorithm outperforms classical enumeration at dimension $n \gtrsim 70$, which is also verified in our experiments. The experimental results reveal that the crossover point of DP enumeration and sieving is around $n \approx 82$, which is an update of the crossover dimension between enumeration and sieving.

In addition to the experimental performance, we also compared the asymptotic behavior of extreme pruned enumeration, G6K sieving and our implementation.

A commonly accepted cost model of extreme pruned enumeration originates from the work of Chen and Nguyen in ASIACRYPT'11 [29]. An explicit fitting function is given by LWE estimator `estimator.BKZ.CheNgu12` [35]:

$$T_{\text{CN11}} = C_{node} \times 2^{0.27019n\ln(n) - 1.0192n + 16.103}$$

However, a more recent work [38] (denoted by [ABF+20] in Figure 11) suggests that the fitting formula should strictly follow the form $2^{1/(2e)n\log(n)+an+b}$, and their fitting result of [29] is

$$T_{\text{extreme}} = C_{node} \times 2^{\frac{1}{2e}n\log(n) - 0.995n + 16.25}$$

The time complexity of sieving is believed to be $2^{0.292+o(n)}$ [10], and G6K gives an even better result by fitting [37]:

$$T_{\text{sieve}} = 2^{0.249n - 14.7} * \text{CPUfreq}$$

Here, CPUfreq $= 2.3 \times 10^9$ according to their implementation; thus, the metric of T_{sieve} is unified to CPU cycles.

Since the cost model of (optimized) DP enumeration can accurately estimate the actual runtime in a high dimension, we used the DP simulator with an optimal parameter set to predict the runtime of discrete pruning during SVP challenge (from $n = 80$ to $n = 160$) and

provide an asymptotic prediction. We required the fitting function to have a fixed form $2^{1/(2e)n\log(n)+an+b}$ to be consistent with [38]. The fitting result is

$$T_{discrete} = 2^{\frac{1}{2e}n\log(n)-1.0232n+24.590}$$

Figure 11 shows the asymptotic behavior of extreme pruned enumeration ($T_{extreme}$), sieved with the dimension-for-free technique (T_{sieve}), and the fitting function of DP simulator ($T_{discrete}$). Both the experimental and asymptotic comparison indicate that the discrete pruned enumeration might have more practical potential than the (classical) extreme pruning in solving high-dimensional SVP, and it might become the most efficient polynomial-space SVP solver known to date.

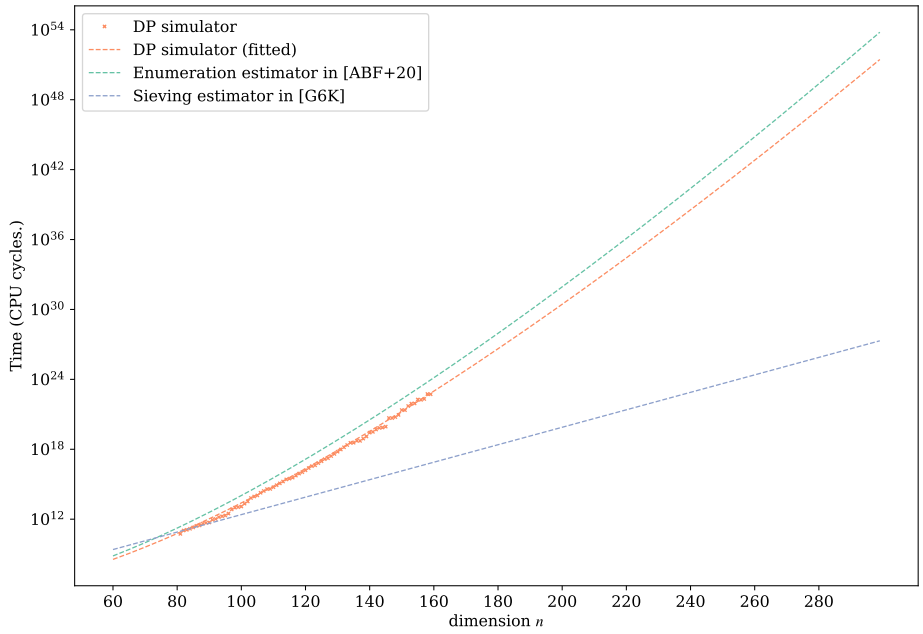

Figure 11. The asymptotic behavior of DP enumeration and other SVP solvers.

7. Conclusions

In this paper, the discrete pruned enumeration algorithm for solving SVP was thoroughly studied and improved. We refined the mathematical theory underlying DP enumeration and propose some improvements to the DP enumeration algorithm to make it more practical. The most valuable part is that our discrete pruning simulator combined theoretical analysis and many numerical techniques. The experimental results verify that the DP simulator can precisely predict the performance of DP enumeration. For a certain SVP instance, we can use the DP simulator to find optimal parameters to minimize the DP enumeration runtime. The explicit time and space consumption is also given by the simulator. Using simulation experiments, we believe that the time complexity of DP enumeration is still superexponential, and the space complexity is still linear, which does not change the conclusion of the enumeration algorithm.

When comparing the performance of our implementation and extreme pruned enumeration, we show that DP enumeration, under optimal parameter settings, could outperform extreme pruning when $n \gtrsim 67$. By comparing this with the state-of-the-art exponential-space SVP algorithm, sieving with dimension for free [36,37], we report an updated crossover point of enumeration and sieving at $n \approx 82$, which is slightly higher than previously observed. Then, at a higher dimension ($80 < n < 300$), we compared the asymptotic behavior of DP enumeration, extreme pruned enumeration and sieving, which also shows the advantage of the discrete pruning method compared with other polynomial-space SVP solvers.

We provide the analytical cost formula of DP enumeration as an asymptotic estimation for cryptanalysis reference, and we hope that the open-source implementation of this work could help cryptologists to further develop the algorithm.

There are several possible directions for improvement:

- Using a stronger reduction algorithm: As the results indicate, when $n \gtrsim 67$, DP enumeration outperforms classical enumeration with extreme pruning, which means that the BKZ algorithm for preprocessing and reprocessing should use DP enumeration as an SVP oracle to achieve higher efficiency, and sieving is also an alternative SVP oracle. The structure of progressive BKZ algorithm [34] also shows high power, although it has a very complicated runtime estimator.
- Discussing the efficiency of many heuristic methods: Fukase and Kashiwabara [18] tried to improve the quality of the basis by inserting short lattice vectors into the basis, but this barely has theoretical proof. Since this method will influence the p_{succ} in every round, the success probability model of the FK algorithm should be modified.
- A parallelized implementation of DP enumeration, as well as an adaptive optimization model.

Author Contributions: Conceptualization and methodology, L.L.; writing—original draft preparation, L.L.; supervision, C.G.; writing—review and editing, Y.Z. and Y.S.; All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Publicly available data was analyzed in this study. These data can be found here: http://www.latticechallenge.org/svp-challenge/, accessed on 30 December 2022.

Acknowledgments: The authors would like to thank Léo Ducas of CWI Amsterdam for his useful comments on this work. We also thank Hao Liang of IHEP (The Institute of High Energy Physics of the Chinese Academy of Sciences) for helping us improve the numerical calculation skills.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. Proof of Theorem 1

Let $\overline{f}(\mathbf{t}) = \sum_{i=1}^{n} \overline{f}(i, t_i) = \sum_{i=1}^{n} (t_i^2 + t_i)\|\mathbf{b}_i\|^2$ be the original objective function proposed in [24]. We only prove Theorem 1 in the case that Algorithm 3 uses $\overline{f}(\mathbf{t})$ as objective function. When GSA holds, $\overline{f}(\mathbf{t}) \approx f(\mathbf{t})$, and we assume the conclusion of $f(\mathbf{t})$ is asymptotically the same with the $\overline{f}(\mathbf{t})$ case.

We note that the initial value $R_1 \leftarrow \sum_{i=1}^{n} \overline{f}(i, \lceil M^{\frac{1}{n}} \rceil)$ guarantees that there are at least M tags, such that $\overline{f}(\mathbf{t}) < R_1$, which is a necessary condition of the correctness of Algorithm 3.

Proof. Let$R_0 = 0$, $R_1 = \sum_{i=1}^{n} \overline{f}(i, \lceil M^{\frac{1}{n}} \rceil)$, and denote an n-dimensional ellipsoid by

$$E_n(\mathbf{a}, R) = \left\{ \mathbf{x} \in \mathbb{R}^n : \sum_{i=1}^{n} \frac{x_i^2}{a_i^2} \leq R \right\}$$

In Algorithm 3, for any $R \in [R_0, R_1]$, the inequality$\overline{f}(\mathbf{t}) \leq R$ is equivalent to

$$\sum_{i=1}^{n} f(i, t_i) = \sum_{i=1}^{n} (t_i + \frac{1}{2})^2 \|\mathbf{b}_i\|^2 - \frac{1}{4} \sum_{i=1}^{n} \|\mathbf{b}_i\|^2 \leq R,$$

i.e.,

$$\sum_{i=1}^{n} (t_i + \frac{1}{2})^2 \|\mathbf{b}_i\|^2 \leq R + \frac{1}{4} \sum_{i=1}^{n} \|\mathbf{b}_i\|^2$$

The number of tags $\mathbf{t} \in \mathbf{Z}^n$, such that satisfies the inequality above, is exactly the number of integer points in an n-dimensional ellipsoid centered on $(-\frac{1}{2}, \ldots, -\frac{1}{2})$. To simplify

the problem, we assume that a translation operation on the ellipsoid would not change the total number of integer points in it, and then we can focus on a centrosymmetric ellipsoid $E_n(\mathbf{a}, R')$, where $a_i = \frac{1}{\|\mathbf{b}_i\|}$ and $R' = R + \frac{1}{4}\sum_{i=1}^{n}\|\mathbf{b}_i\|^2$. Then, we define

$$M(R) \overset{def}{=} \#\{\mathbf{t} : \overline{f}(\mathbf{t}) \le R\} \sim \#\{E_n(\mathbf{a}, R') \cap \mathbb{Z}^n\}$$

It is obvious that $M(R)$ is a monotone undecreasing function of R. Assume that binary search Algorithm 3 terminates at the k-th iteration, and denote the upper bound and lower bound of radius by R_{lk} and R_{rk}, then we have $(1-\epsilon)M_0 \le M(\frac{R_{rk}+R_{lk}}{2}) \le (1+\epsilon)M_0$, with M_0 being the input of Algorithm 3. The target of our proof is to find a ΔR such that $R_{rk} - R_{lk} > \Delta R$ holds for all possible terminating values of (R_{lk}, R_{rk}). Then, it is easy to prove that the binary search ends in $\log \frac{R_1 - R_0}{\Delta R}$ rounds of iteration.

Now, assume R_{lk}, R_{rk} satisfying $M(R_{lk}) < (1-\epsilon)M_0 \le M(\frac{R_{rk}+R_{lk}}{2}) \le (1+\epsilon)M_0 < M(R_{rk})$. Then we have

$$M(R_{rk}) - M(R_{lk}) > 2\epsilon M_0 \tag{A1}$$

Let $A_n(R') = \#E_n(\mathbf{a}, R' = R + \frac{1}{4}\sum_{i=1}^{n}\|\mathbf{b}_i\|^2) \cap \mathbb{Z}^n$; we can investigate the asymptotic behavior of $M(R)$ by estimating the value of $A_n(R')$.

There are some mature conclusions on the estimation of $A_n(x)$ [39,40]. $A_n(x)$ can be written as

$$A_n(x) = \frac{V(B_n)}{\prod_{i=1}^{n}\|\mathbf{b}_i\|}x^{n/2} + P_n(x) \tag{A2}$$

where $P_n(x) \ll x^{\frac{n}{2}\cdot\frac{n-1}{n+1}}$ and can be written as $P_n(x) = O(x^n)$, and $V(B_n)$ is the volume of n-dimensional unit sphere:

$$V(B_n) = \frac{\pi^{n/2}}{\Gamma(\frac{1}{2}+1)} \approx \left(\frac{2\pi e}{n}\right)^{\frac{n}{2}}$$

Although $M(R)$ is a discrete function of R, we can use the value of $A_n(x)$ at $x = R + \frac{1}{4}\sum_{i=1}^{n}\|\mathbf{b}_i\|^2$ as an approximation. In this case $A_n(x) = O(2x^{n/2})$ in an asymptotic sense, and then according to the Lagrange mean value theorem, for $x_{lk} = R_{lk} + \frac{1}{4}\sum_{i=1}^{n}\|\mathbf{b}_i\|^2, x_{rk} = R_{rk} + \frac{1}{4}\sum_{i=1}^{n}\|\mathbf{b}_i\|^2$, there exists $x_\xi \in (x_{lk}, x_{rk})$ such that

$$\frac{A_n(x_{rk}) - A_n(x_{lk})}{x_{rk} - x_{lk}} = A_n'(x_\xi) \le n{x_\xi}^{\frac{n}{2}-1} < n{x_{rk}}^{\frac{n}{2}-1} \tag{A3}$$

Combining Equations (A1) and (A3), we have

$$R_{rk} - R_{lk} = x_{rk} - x_{lk} = \frac{A_n(x_{rk}) - A_n(x_{lk})}{A_n'(x_\xi)} \gtrapprox \frac{2\epsilon M_0}{n{x_{rk}}^{n/2-1}} \tag{A4}$$

Therefore, the total rounds of iterations of Algorithm 3 is at most

$$\begin{aligned} &\log\frac{R_1 - R_0}{R_{rk} - R_{lk}} \\ &< \log n + \frac{n}{2}\log\left(R_1 + \frac{1}{4}\sum_{i=1}^{n}\|\mathbf{b}_i\|^2\right) - \log 2\epsilon M_0 \\ &= \log n + \frac{n}{2}\log\left(\left(\lceil M_0^{\frac{1}{n}}\rceil + \frac{1}{2}\right)^2 \mathrm{GSS}(\mathbf{B})\right) - \log 2\epsilon M_0 \end{aligned} \tag{A5}$$

where $\mathrm{GSS}(\mathbf{B}) = \sum_{i=1}^{n}\|\mathbf{b}_i\|^2 \le n\det(L)^{\frac{2}{n}}$. By further approximation and simplification, we can know that the algorithm ends in at most $O\left(\log n + \log\frac{1}{\epsilon} + n\log\left(n\det(L)^{\frac{2}{n}}\right)\right)$ rounds. □

Appendix B. Calculating Success Probability by FILT and Euler Transformation

In this part, we introduce some detailed derivation of the numerical methods for computing success probability.

Let x_i be uniformly distributed on $[\alpha_i, \beta_i]$, then the probability density function of x_i^2 is

$$\rho_{x_i^2}(z) = \begin{cases} \frac{1}{2(\beta_i - \alpha_i)\sqrt{z}}, & z \in [\alpha_i^2, \beta_i^2] \\ 0, & \text{else} \end{cases}$$

Therefore, the p.d.f. of $\sum_{i=1}^n x_i^2$ is

$$\rho_{\sum_{i=1}^n x_i^2}(z) = \left(\rho_{x_1^2} * \rho_{x_2^2} * \ldots * \rho_{x_n^2}\right)(z)$$

where "$*$" denotes the convolution operation $f * g(z) \overset{def}{=\!=} \int_0^z f(\tau) g(z-\tau)\, d\tau$.

To estimate the success probability of DP enumeration, our goal is to calculate $Pr\{\sum_{i=1}^n x_i^2 \le 1\}$.

Step 1. Fast inverse Laplace transform

Theorem A1. *If the random variable X is non-negative and has p.d.f. $p(x)$, then the c.d.f of X is*

$$D(x) = \mathcal{L}^{-1}\left\{\frac{1}{s}\mathcal{L}\{p\}(s)\right\}(x)$$

Here, the symbol $\mathcal{L}$ specially refers to the Laplace transform, which satisfies

$$\mathcal{L}\left\{\rho_{\sum_{i=1}^n x_i^2}(z)\right\} = \mathcal{L}\{\rho_{x_1^2}\} \cdot \ldots \cdot \mathcal{L}\{\rho_{x_n^2}\}$$

Then, our goal is to calculate the value

$$D(1) = Pr\left\{\sum_{i=1}^n x_i^2 \le 1\right\} = \mathcal{L}^{-1}\left\{\frac{1}{s}\mathcal{L}\{\rho_{\sum_{i=1}^n x_i^2}\}(s)\right\}(t) \mid_{t=1} \tag{A6}$$

Note that $s \in \mathbb{C}$, since Laplace inverse transform is an integral in a complex field with an integral path perpendicular to the x-axis.

To calculate $D(1)$ in Equation (A6), we first perform the Laplace transform

$$\begin{aligned} F(s) &\overset{def}{=\!=} \frac{1}{s}\mathcal{L}\{\rho_{\sum_{i=1}^n x_i^2}\}(s) \\ &= \frac{1}{s}\mathcal{L}\{\rho_{x_1^2}\} \cdot \ldots \cdot \mathcal{L}\{\rho_{x_n^2}\} = \frac{\pi^{n/2}}{s^{\frac{n}{2}+1}} \prod_{i=1}^n \frac{\mathrm{erf}(\beta_i\sqrt{s}) - \mathrm{erf}(\alpha_i\sqrt{s})}{2(\beta_i - \alpha_i)} \end{aligned}$$

and then apply inverse Laplace transform

$$\begin{aligned} D(1) =& Pr\left\{\sum_{i=1}^n x_i^2 \le 1\right\} \\ =& \frac{1}{2\pi \mathrm{i}} \int_{c-\infty \mathrm{i}}^{c+\infty \mathrm{i}} F(s) \mathrm{e}^{st}\, ds \mid_{t=1} \\ =& \frac{1}{2\pi \mathrm{i}} \int_{c-\infty \mathrm{i}}^{c+\infty \mathrm{i}} F(s) \mathrm{e}^{s}\, ds \\ =& \frac{1}{2\pi \mathrm{i}} \int_{c-\infty \mathrm{i}}^{c+\infty \mathrm{i}} \left\{ \frac{\pi^{\frac{n}{2}}}{s^{\frac{n}{2}+1}} \cdot \prod_{j=1}^n \frac{\mathrm{erf}(\beta_j\sqrt{s}) - \mathrm{erf}(\alpha_j\sqrt{s})}{2(\beta_j - \alpha_j)} \right\} \cdot \mathrm{e}^{s}\, ds \end{aligned} \tag{A7}$$

Step 2. Approximate integral calculation with series

Put the approximation of e^s in complex field

$$e^s \approx E_{ec}(s,a) \overset{def}{=\!=} \frac{exp(a)}{2\cosh(a-s)} = \frac{e^a}{2}\sum_{m=-\infty}^{+\infty}\frac{i(-1)^m}{s-a-(m-\frac{1}{2})\pi i}$$

into Equation (A7), (here, the value of a should guarantee the convergence of series. For example, Hosono [41] claimed that the error is very small when $a \gg 1$, and Aono and Nugyen [23] recommended to use $a = \max(50, 30+3\sqrt{n})$); now, notice that the integral has singularity points $s_m = a + (m-\frac{1}{2})\pi i$, $m = 1, \ldots, \infty$ (in Equation (A7), the integral path should be to the right of all singularities, i.e., $c > a$. Additionally, for the $\sqrt{s}$ in $F(s)$, since s is a complex variable, the argument of $\sqrt{s}$ should satisfy $|\arg(z)| < \frac{\pi}{4}$ to be consistent with the integration path).

According to the residue theorem and Jordan theorem, Equation (A7) can be approximated by

$$D(1) \approx e^a \cdot \sum_{m=1}^{+\infty} \mathrm{Im}F\left(a + (m-\frac{1}{2})\pi i\right) \tag{A8}$$

Step 3. Using Euler transformation to accelerate the convergence of series

Since the series in Equation (A8) converges slowly, the Euler transformation is a practical method to accelerate the convergence. Therefore, we can use fewer terms to approximate the infinite series.

Let $F_m = \mathrm{Im}F\left(a + (m-\frac{1}{2})\pi i\right)$, then the value of Equation (A8) can be calculated by finite terms:

$$\sum_{m=1}^{+\infty}(-1)^m F_m \approx \sum_{m=1}^{K}(-1)^m F_m + (-1)^K \sum_{j=1}^{J}\frac{(-1)^j \Delta^{j-1}F_{K+1}}{2^j}$$

where $\Delta^{j-1}F_{K+1} = \sum_{i=0}^{j-1}(-1)^j\binom{j-1}{i}F_{j+K-1}$ is the "forward difference" that can be iteratively computed by van Wijngaarden transformation. In our implementation, we set $K = 40$, $J = 30$ by default.

Remark A1. *Remarks. The computation of* $ImF(s)$ *at* $s = a + (m-\frac{1}{2})\pi i$ *is a time-consuming procedure. It involves the computation of* $\mathrm{erf}(\cdot)$ *over complex field* $\mathbb{C}$*, which also needs to be approximated by series expansion. In the original computation model, since* α_i *and* β_i *only have a few fixed values only related with the GS sequence, we can accelerate the computation by building an "erf table" to record some values of* $\mathrm{erf}(\alpha_i\sqrt{s})$ *and* $\mathrm{erf}(\beta_i\sqrt{s})$ *that would be repeatedly used in the calculation. However, for the rectified success probability model, the values of* α_i *and* β_i *are also connected with the explicit value of cell tag* **t***, which makes the "erf table" invalid, and the running time could be very long. Fortunately, we still find that the original computation model could help us to estimate the rectified success probability. In our implementation of DP enumeration, in addition to the step-by-step computation, we also provide a heuristic method using the harmonic average of* $p_{succ,odd}$ *and* $p_{succ,even}$*, which can be calculated efficiently, to roughly estimate the actual success probability.*

References

1. Coster, M.; Joux, A.; Lamacchia, B.; Odlyzko, A.; Schnorr, C.; Stern, J. Improved Low-Density Subset Sum Algorithms. *Comput. Complex.* **1999**, *2*, 111–128. [CrossRef]
2. Schnorr, C.P.; Euchner, M. Lattice Basis Reduction: Improved Practical Algorithms and Solving Subset Sum Problems. *Math. Program.* **1994**, *66*, 181–199. [CrossRef]
3. Nguyen, P.Q.; Stern, J. Adapting Density Attacks to Low-Weight Knapsacks. In *Advances in Cryptology—ASIACRYPT 2005* ; Springer: Berlin/Heidelberg, Germany, 2005; pp. 41–58. [CrossRef]
4. Li, S.; Fan, S.; Lu, X. Attacking ECDSA Leaking Discrete Bits with a More Efficient Lattice. In *Proceedings of the Inscrypt 2021*; Springer: Cham, Switzerland, 2021.
5. Schnorr, C.P. Fast Factoring Integers by SVP Algorithms, Corrected. Cryptology ePrint Archive, Report 2021/933. 2021. Available online: https://ia.cr/2021/933 (accessed on 30 December 2022).

6. Kannan, R. Improved Algorithms for Integer Programming and Related Lattice Problems. In *Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing*; Association for Computing Machinery: New York, NY, USA, 1983; pp. 193–206. [CrossRef]
7. Fincke, U.; Pohst, M. Improved methods for calculating vectors of short length in a lattice, including a complexity analysis. *Math. Comput.* **1985**, *44*, 463–471. [CrossRef]
8. Gama, N.; Nguyen, P.Q. Predicting Lattice Reduction. In *Proceedings of the Advances in Cryptology—EUROCRYPT 2008*; Smart, N., Ed.; Springer: Berlin/Heidelberg, Germany, 2008; Volume 4965, pp. 31–51. [CrossRef]
9. Lindner, R.; Peikert, C. Better Key Sizes (and Attacks) for LWE-Based Encryption. In *Topics in Cryptology—CT-RSA 2011* ; Springer: Berlin/Heidelberg, Germany, 2011; pp. 319–339. [CrossRef]
10. Alkim, E.; Ducas, L.; Pöppelmann, T.; Schwabe, P. Post-quantum Key Exchange—A New Hope. In Proceedings of the 25th USENIX Security Symposium (USENIX Security 16), Austin, TX, USA, 10–12 August 2016 ; USENIX Association: Austin, TX, USA, 2016; pp. 327–343.
11. Pohst, M. On the Computation of Lattice Vectors of Minimal Length, Successive Minima and Reduced Bases with Applications. *SIGSAM Bull.* **1981**, *15*, 37–44. [CrossRef]
12. Gama, N.; Nguyen, P.Q.; Regev, O. Lattice Enumeration Using Extreme Pruning. In *Proceedings of the Advances in Cryptology—EUROCRYPT 2010*; Gilbert, H., Ed.; Springer: Berlin/Heidelberg, Germany, 2010; pp. 257–278.
13. Lattice Algorithms Using Floating-Point Arithmetic(fplll). Available online: https://github.com/fplll/fplll (accessed on 30 December 2022).
14. Aono, Y.; Nguyen, P.Q.; Seito, T.; Shikata, J. Lower Bounds on Lattice Enumeration with Extreme Pruning. In *Proceedings of the Advances in Cryptology—CRYPTO 2018*; Shacham, H., Boldyreva, A., Eds.; Springer International Publishing: Berlin/Heidelberg, Germany, 2018; pp. 608–637.
15. Schnorr, C.P. Lattice Reduction by Random Sampling and Birthday Methods. In *Proceedings of the Annual Symposium on Theoretical Aspects of Computer Science (STACS 2003)*; Alt, H., Habib, M., Eds.; Springer: Berlin/Heidelberg, Germany, 2003; pp. 145–156.
16. Ajtai, M. The Worst-Case Behavior of Schnorr's Algorithm Approximating the Shortest Nonzero Vector in a Lattice. In Proceedings of the Thirty-Fifth Annual ACM Symposium on Theory of Computing, San Diego, CA, USA, 9–10 June 2003; Association for Computing Machinery: New York, NY, USA, 2003; pp. 396–406. [CrossRef]
17. Buchmann, J.; Ludwig, C. Practical Lattice Basis Sampling Reduction. In *Proceedings of the Algorithmic Number Theory*; Hess, F., Pauli, S., Pohst, M., Eds.; Springer: Berlin/Heidelberg, Germany, 2006; pp. 222–237.
18. Fukase, M.; Kashiwabara, K. An accelerated algorithm for solving SVP based on statistical analysis. *J. Inf. Process.* **2015**, *23*, 67–80. [CrossRef]
19. Teruya, T.; Kashiwabara, K.; Hanaoka, G. Fast Lattice Basis Reduction Suitable for Massive Parallelization and Its Application to the Shortest Vector Problem. In *Proceedings of the Public-Key Cryptography—PKC 2018*; Abdalla, M., Dahab, R., Eds.; Springer International Publishing: Cham, Switzerland, 2018; pp. 437–460.
20. Fukase, M.; Yamaguchi, K. Analysis of the Extended Search Space for the Shortest Vector in Lattice. In Proceedings of the IMETI 2010—3rd International Multi-Conference on Engineering and Technological Innovation, Proceedings, Orlando, FL, USA, 29 June–2 July 2010; Volume 2.
21. Fukase, M.; Yamaguchi, K. Finding a Very Short Lattice Vector in the Extended Search Space. *Trans. Inf. Process. Soc. Jpn.* **2012**, *53*, 11. [CrossRef]
22. Ding, D.; Zhu, G. A Random Sampling Algorithm for SVP Challenge based on y-Sparse Representations of Short Lattice Vectors. In Proceedings of the 2014 Tenth International Conference on Computational Intelligence and Security, Kunming, China, 15–16 November 2014. [CrossRef]
23. Aono, Y.; Nguyen, P.Q. Random Sampling Revisited: Lattice Enumeration with Discrete Pruning. In *Proceedings of the Advances in Cryptology—EUROCRYPT 2017*; Coron, J.S., Nielsen, J.B., Eds.; Springer International Publishing: Cham, Switzerland, 2017; pp. 65–102.
24. Aono, Y.; Nguyen, P.Q.; Shen, Y. Quantum Lattice Enumeration and Tweaking Discrete Pruning. In *Proceedings of the Advances in Cryptology—ASIACRYPT 2018*; Peyrin, T., Galbraith, S., Eds.; Springer International Publishing: Berlin/Heidelberg, Germany, 2018; pp. 405–434.
25. Ludwig, C. Practical Lattice Basis Sampling Reduction. Ph.D. Thesis, der Technischen Universität Darmstadt, Darmstadt, Germany, 2005.
26. Goldstein, D.; Mayer, A. On the equidistribution of Hecke points. *Forum Math.* **2003**, *15*, 165–189. [CrossRef]
27. Chen, Y. Réduction de Réseau et Sécurité Concrète du Chiffrement Complètement Homomorphe. Ph.D. Thesis, Université Paris Diderot (Paris 7), Paris, France, 2013.
28. TU Darmstadt, SVP Challenge. Available online: https://www.latticechallenge.org/svp-challenge/ (accessed on 30 December 2022).
29. Chen, Y.; Nguyen, P.Q. BKZ 2.0: Better Lattice Security Estimates. In *Advances in Cryptology—ASIACRYPT 2011*; Springer: Berlin/Heidelberg, Germany, 2011; Volume 7073, pp. 1–20. [CrossRef]
30. Bai, S.; Stehlé, D.; Wen, W. Measuring, Simulating and Exploiting the Head Concavity Phenomenon in BKZ. In *Proceedings of the Advances in Cryptology—ASIACRYPT 2018*; Peyrin, T., Galbraith, S., Eds.; Springer International Publishing: Berlin/Heidelberg, Germany, 2018; pp. 369–404.

31. Hanrot, G.; Pujol, X.; Stehlé, D. Analyzing Blockwise Lattice Algorithms Using Dynamical Systems. In *Proceedings of the Advances in Cryptology—CRYPTO 2011*; Springer: Berlin/Heidelberg, Germany, 2011; pp. 447–464.
32. Albrecht, M.R. On Dual Lattice Attacks Against Small-Secret LWE and Parameter Choices in HElib and SEAL. In *Advances in Cryptology—EUROCRYPT 2017*; Springer International Publishing: Berlin/Heidelberg, Germany, 2017; pp. 103–129.
33. Stehlé, D. Floating-Point LLL: Theoretical and Practical Aspects. In *The LLL Algorithm: Survey and Applications*; Nguyen, P.Q., Vallée, B., Eds.; Springer: Berlin/Heidelberg, Germany, 2010; pp. 179–213. [CrossRef]
34. Aono, Y.; Wang, Y.; Hayashi, T.; Takagi, T. Improved Progressive BKZ Algorithms and Their Precise Cost Estimation by Sharp Simulator. In *Proceedings of the Advances in Cryptology—EUROCRYPT 2016*; Springer: Berlin/Heidelberg, Germany, 2016; pp. 789–819. [CrossRef]
35. Albrecht, M.; Göpfert, F.; Lefebvre, C.; Owen, J.; Player, R. LWE Estimator's Documentation. Online. Available online: https://lwe-estimator.readthedocs.io/en/latest/index.html (accessed on 25 December 2022).
36. The General Sieve Kernel. Available online: https://github.com/fplll/g6k (accessed on 25 December 2022).
37. Albrecht, M.R.; Ducas, L.; Herold, G.; Kirshanova, E.; Postlethwaite, E.W.; Stevens, M. The General Sieve Kernel and New Records in Lattice Reduction. In Proceedings of the Advances in Cryptology—EUROCRYPT 2019, Kobe, Japan, 8–12 December 2019; pp. 717–746.
38. Albrecht, M.R.; Bai, S.; Fouque, P.A.; Kirchner, P.; Stehlé, D.; Wen, W. Faster Enumeration-Based Lattice Reduction: Root Hermite Factor $k^{1/(2k)}$ Time $k^{k/8+o(k)}$. In Proceedings of the Advances in Cryptology—CRYPTO 2020, Santa Barbara, CA, USA, 17–21 August 2020; pp. 186–212.
39. Müller, W. Lattice Points in Large Convex Bodies. *Monatshefte Math.* **1999**, *128*, 315–330. [CrossRef]
40. Krätzel, E. A sum formula related to ellipsoids with applications to lattice point theory. *Abh. Aus Dem Math. Semin. Der Univ. Hambg.* **2001**, *71*, 143–159. [CrossRef]
41. Hosono, T. Numerical inversion of Laplace transform and some applications to wave optics. *Radio Sci.* **1981**, *16*, 1015–1019. [CrossRef]