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

Luan Luan, Chunxiang Gu, Yonghui Zheng and Yanan Shi

July 29, 2022

**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 mature theoretical analysis and practical implementation. However, these works are still remain to be done for discrete pruning. In this paper, we improve the discrete pruned enumeration (DP enumeration), and give a solution to the problem proposed by Léo Ducas et Damien Stehlé [19] about the cost estimation of discrete pruning. Our contribution is on the following three aspects:

First, we refine the algorithm both from theoretical and practical aspects. Discrete pruning using natural number representation lies on a randomness assumption of lattice point distribution, which has an obvious paradox in the original analysis. We rectify this assumption to fix the problem, and correspondingly modify some details of DP enumeration. We also improve the binary search algorithm for cell enumeration radius with polynomial time complexity, and refine the cell decoding algorithm. Besides, we propose to use a truncated lattice reduction algorithm – k-tours-BKZ as reprocessing method when a round of enumeration failed.

Second, we propose a cost estimation simulator for DP enumeration. Based on the investigation of lattice basis stability during reprocessing, we give a method to simulate the squared length of Gram-Schmidt orthogonalization basis quickly, and give the fitted cost estimation formulae of sub-algorithms in CPU-cycles through intensive experiments. The success probability model is also modified based on the rectified assumption. We verify the cost estimation simulator on middle size SVP challenge instances, and the simulation results are very close to the actual performance of DP enumeration.

Third, we give a method to calculate the optimal parameter setting to minimize the running time of DP enumeration. We compare the efficiency of our optimized DP enumeration with extreme pruning enumeration in solving SVP challenge instances. The experimental results in medium dimension and simulation results in high dimension both show that the discrete pruning method could outperform extreme pruning. An open-source implementation of DP enumeration with its simulator is also provided[1].

Key Words: lattice, SVP, Gram-Schmidt orthogonalization, enumeration, discrete pruning

## 1 Introduction

The shortest vector problem (SVP) and closest vector problem (CVP) are hard computing problems of lattice, 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 variant version of SVP or CVP, such as the knapsack problem [17, 37, 39], the hidden number problem [32] and the integer factoring problem [42, 43].

Lattice enumeration is a general SVP solver with linear space complexity, which can be traced back to the early 1980s [20,30,38]. It outputs a lattice vector (or proves there is none) shorter than the given target length within super-exponential time. Enumeration can be directly applied to solve exact/approximate version of SVP, and can also be modified to solve CVP and bounded distance decoding problem(BDD), see [10,34,42]. Enumeration can also be used as the subroutine of block-wise lattice basis reduction (BKZ)

[1]`https://github.com/LunaLuan9555/DP-ENUM`

algorithm, and therefore play an important role in the security analysis and parameter assessment of lattice-based cryptosystems [7, 12, 24, 33].

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. Pruning method cuts off the branch and stops searching in depth when the objective function value at current node exceeds the bounding function. It might cut off the correct solution during searching, hence enumeration becomes a probability algorithm. Pruned enumeration works well in practice since the target length is usually known, and it is expected to find a solution to SVP if the algorithm is repeated on a lattice for enough times. Schnorr and Hörner [40] proposed a heuristic pruning strategy and the algorithm implementation was given in NTL library [2]. Gama, Nguyen and Regev [25] proposed the extreme pruning method by 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. The fplll library [1] provides a set of extreme pruning strategies for BKZ 2.0 [16] on random lattice, and also provides a Nelder-Mead algorithm for searching the optimal bounding function for any given basis.

The classical pruned enumeration searches lattice vectors in a hyper cylinder intersection, which is regarded as a continuous region in time analysis. Consequently, the computation of expected running time of GNR enumeration is easy to be handled, 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 [9]. In cryptanalysis, most commonly used cost model of GNR enumeration is given in the LWE estimator [6], which adopted some conclusion of [16].

Discrete pruning method is quiet different from that. The discrete pruned enumeration (DP enumeration for short) originated from a heuristic "Sampling - Reduction" algorithm [41], which iteratively samples lattice vectors under the restriction on their Gram-Schmidt coefficients and then re-randomizes basis by lattice reduction. Ajtai, Buchmann and Ludwig [4,14] gave some analyses on the time complexity and success probability. Fukase and Kashiwabara [21] put forward a series of significant improvements, including the natural number representation (NNR) of lattice point, to make sampling reduction method more practical, and provided heuristic analysis. Teruya et al. [45] designed a parallelized version of Fukase-Kashiwabara sampling reduction algorithm and solved a 152 dimensional SVP challenge, which was the best record of that year. Other relevant studies can be referred to [18,22,23]. Sampling-Reduction algorithm shows good practicality but lacks sufficient theoretical support, especially on the parameter settings and estimation of running time. The conception of "discrete pruning" was formally put forward on EUROCRYPT' 17 by Aono and Nguyen [8]. They proposed a novel conception named "lattice partition" to generalize the previous sampling methods, and they solved the problem that what kind of lattice points should be "sampled" using 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. [10] made some modifications to DP enumeration and proposed a quantum variant. The theoretical foundation of DP enumeration is gradually developed, but some problems still remain to be figured out:

1. A precise cost estimation. For classical cylinder pruning, the most time-consuming operation is processing every node on the enumeration tree, and the running time is proportional to the total number of nodes, which is reduced to calculating the volume of several hyper cylinder-intersections. However, things are not that easy for discrete pruning. There is a gap between theoretical time complexity and the actual cost. It is proved that each sub-algorithm of DP enumeration has polynomial-time complexity, but the actual running time is not in proportion with the theoretical upper bound, since these sub-algorithms with different structure are analyzed using different arithmetic operation. To estimate the precise cost of DP enumeration, it is necessary to define a unified "basic operation" for all sub-algorithms of DP enumeration, and fit the coefficients of polynomials. Furthermore, we can extrapolate the estimating model to higher dimensions through some simulating techniques inspired by [16].

2. The optimal parameter setting. An important problem that [8, 10] did not explain clearly is how many points should be enumerated in the iteration. [21] and [8] gave some examples of parameter selection without further explanation. For a certain SVP instance, there should be an optimal parameter settings to minimize the total running time. As a matter of fact, this is based on the

solution to the first problem.

*Contributions of this work.*

1. *Rectification on assumption.* It is generally assumed that lattice point in participated "cells" follow the uniform distribution, but Ludwig [35](Sec 2.4) pointed out that this randomness assumption does not strictly hold for Schnorr's sampling method, i.e. Schnorr's partition. We point out that this defect also exists in the randomness assumption of natural number partition, and lead to a paradox that two symmetric lattice vectors with same length have different moment value and success probability. We give a rectified assumption to describe lattice point distribution in cells more cautiously and accurately, and consequently eliminate the paradox.

2. *Improvement on algorithm implementation.* We propose a new polynomial-time binary search algorithm for finding the cell enumeration radius, which has lower time complexity than [10]. We propose to use a truncated version of BKZ, which is called "k-tours-BKZ", as the reprocessing method when DP enumeration failed in one round and has to be repeated. This is not only a heuristic strategy in practice, but also crucial to our cost estimator. We examine the stabilization of basis quality during repeatedly 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.* We provide an open-source implementation of DP enumeration, and calculate the fitted time cost formula in CPU-cycles for each sub-algorithm of it. We also modify the calculation procedures of success probability according to the rectified randomness assumption. Then we propose a cost simulator to estimate the exact cost of DP enumeration under any given parameters. In addition, for random lattices with GSA assumption holding, it works in a simple and efficient way, without computing any specific lattice basis.

4. *Optimization of algorithm parameters.* The parameters of DP enumeration used to be set by hand empirically. This paper provide an optimization model for finding a set of parameters which minimize the expected total running time of DP enumeration for solving a given SVP instance. The estimated running time under the optimal parameter setting can be regarded as an upper bound of DP enumeration.

We compare the efficiency of extreme pruned enumeration in fplll library [1] and DP enumeration in our implementation on SVP challenge [3]. The result shows that on SVP challenge instances, DP enumeration under optimal parameter setting could outperform the extreme pruning when $n \gtrsim 80$. We also give an analytical cost formula as an asymptotic estimation.

*Roadmap.* Section 2 introduces the fundamental knowledge of lattice, and gives an overview of pruning technologies of lattice enumeration. Section 3 first rectifies the basic randomness assumption of lattice partition, and then describe the details of three improvements on discrete pruning enumeration. Section 4 shows the details of our cost simulator of DP enumeration, including the running time estimation of every sub-algorithms and the rectified success probability model. Section 5 describes how to find the optimal parameter setting for DP enumeration using our cost simulator, and gives experimental results to verify the accuracy of our cost simulator. We also compare the efficiency of our implementation with extreme pruned enumeration in fplll library. Finally, Section 6 6 gives conclusion and discusses some further works.

## 2 Preliminaries

### 2.1 Lattice

*Lattice.* Let $\mathbb{R}^m$ denotes the $m$-dimensional Euclidean space. Given $n$ linear independent vectors $\mathbf{b}_1, ..., \mathbf{b}_n \in \mathbb{R}^m$ $(m \geq n)$, 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 $\mathbf{B} = [\mathbf{b}_1, ..., \mathbf{b}_n]$. 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 real number is represented by rational number in computer, and lattice with rational basis can always be scaled to one with 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 have 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* $\mathcal{P}(\mathbf{B}) = \{\sum_{i=1}^{n} \mathbf{b}_i x_i : 0 \le x_i < 1, i = 1, \ldots, n\}$. 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 $\mathrm{vol}(\mathcal{L})$. It is also called the *determinant* $\det(\mathcal{L})$ of a lattice, and we have $\det(\mathcal{L}) = \mathrm{vol}(\mathcal{L}) = \sqrt{\det(\mathbf{B}^T\mathbf{B})}$.

*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 $\|\mathbf{Bx}\| \le \|\mathbf{By}\|$ 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 non-zero vector of a lattice in the fields of complexity theory, computational algebra and cryptanalysis. But a more common case in cryptanalysis is to find a lattice vector shorter than a given bound. In other words, researchers are more interested in finding an approximate solution to SVP. For example, the target of SVP challenge [3] is to find a lattice vector $\mathbf{v} \in \mathcal{L}$ such that $\|\mathbf{v}\|^2 \le 1.05 \cdot \mathrm{GH}(\mathcal{L}) \approx 1.05\lambda_1(\mathcal{L})$.

*Random lattice.* The formal definition and generation algorithm of random lattice can refer to Goldstein and Mayer's work in [26]. The SVP challenge also adopts Goldstein-Mayer lattice. The lattice of an n-dimensional SVP challenge instance has volume 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 and 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 Gaussian heuristic should be used carefully because in some "bad" cases this assumption does not hold (see [15], Section 2.1.2). But in random lattice, this assumption generally holds especially for some "nice" set $S$, and therefore we can use Gaussian heuristic to predict $\lambda_1(\mathcal{L})$:

$$\mathrm{GH}(\mathcal{L}) = \frac{\mathrm{vol}(\mathcal{L})^{1/n}}{B_n(1)^{1/n}} = \frac{1}{\sqrt{\pi}}\Gamma\Big(\frac{n}{2}+1\Big)^{\frac{1}{n}}\mathrm{vol}(\mathcal{L})^{\frac{1}{n}} \approx \sqrt{\frac{n}{2\pi e}}\mathrm{vol}(\mathcal{L})^{\frac{1}{n}} \tag{2.1}$$

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 $n \gtrapprox 45$.

*Orthogonal projection.* The Gram-Schmidt orthogonalization can be considered as a direct decomposition of lattice, and is frequently used in lattice problems.

**Definition 1** *Gram-Schmidt orthogonalization. Let $\mathbf{B} = [\mathbf{b}_1, \ldots \mathbf{b}_n] \in \mathbb{Z}^{m\times n}$ be a lattice basis, The Gram-Schmidt orthogonal basis $\mathbf{B}^* = [\mathbf{b}_1^*, \ldots \mathbf{b}_n^*]\,\mathbb{Q}^{m\times n}$ 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 $\pi_i : \mathbb{R}^m \to span(\mathbf{b}_1, \ldots, \mathbf{b}_{i-1})^\perp$ 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 $i = 1, \ldots, n$, we can define the $n-i+1$-dimensional projected lattice $\pi_i(\mathcal{L}(\mathbf{B})) = \mathcal{L}\left([\pi_i(\mathbf{b}_i), \pi_i(\mathbf{b}_{i+1}), \ldots, \pi_i(\mathbf{b}_n)]\right)$. Note that the orthogonal basis of $\pi_i(\mathcal{L}(\mathbf{B}))$ is exactly $[\mathbf{b}_i^*, \mathbf{b}_{i+1}^*, \ldots, \mathbf{b}_n^*]$.

## 2.2 Classical Enumeration and Pruning

For lattice vector represented by an orthogonal basis, we have

$$\begin{aligned}\mathbf{v} &= \sum_{j=1}^{n} u_j \mathbf{b}_j^* \\ &= \sum_{j=1}^{n} \left( x_j + \sum_{i=j+1}^{n} \mu_{ij} x_i \right) \mathbf{b}_j^*\end{aligned}$$

Then we can enumerate the coordinates $x_n, \ldots, x_1$ by a depth-first search method to find all lattice vector shorter than $R$. The searching path can be organized in the form of *enumeration tree*. This is the original idea of lattice enumeration [20] and was improved in [39]. In the $k$-th layer of the tree, all possible values of $(x_{n-k+1}, \ldots, x_n)$ are enumerated and those with partial sum $\|\pi_{n-k+1}(\mathbf{v})\| = \sum_{j=n-k+1}^{n} |x_j + \sum_{i=j+1}^{n} \mu_{ij} x_i| \|\mathbf{b}_j^*\| \leq R$ are kept. The expected number of nodes in $k$-th layer that need to be process, denoted by $H_k$, equals to the number of vectors in projected lattice $\pi_{n-k+1}(\mathcal{L})$ with length smaller than $R$ :

$$H_k = \frac{1}{2} \frac{V_k(R)}{\mathrm{vol}(\pi_{n-k+1}(\mathcal{L}))} = \frac{1}{2} \frac{V_k(R)}{\prod_{i=n+1-k}^{n} \|\mathbf{b}_i^*\|}$$

where $V_k(R)$ denotes the volume of a $k$-dimensional ball with radius $R$. Therefore, the algorithm processes $N = \sum_{k=1}^{n} H_k$ nodes in total. Obviously the cost of enumeration is tightly related to the length of orthogonal basis $\{\|\mathbf{b}_i^*\|\}_{i=1}^{n}$. Hanrot et al. [28] proved that under a well-reduced basis, the complexity of classical enumeration is $\mathrm{Poly}(\log B, m) \cdot n^{\frac{n}{2e} + o(n)}$, and it is widely believed that the cost of enumeration has an upper bound up to $2^{O(n^2)}$ on a slightly reduced basis such as LLL-reduced basis.

At some intermediate nodes of the tree, the partial sum $\|\pi_{n-k+1}(\mathbf{v})\| = \sum_{j=n-k+1}^{n} |x_j + \sum_{i=j+1}^{n} \mu_{ij} x_i| \|\mathbf{b}_j^*\|$ is very close to $R$, which means it is unlikely to find a solution in this branch, and one can stop to search downwards. To accelerate the searching, it is reasonable to prune the "hopeless" branches by restricting $\|\pi_{n+1-k}(\mathbf{v})\| \leq R_k$, where $R_1 \leq R_2 \leq \ldots \leq R_n = R$. Then the searching area of coordinates $(x_{n-k+1}, \ldots, x_n)$ becomes a $k$-dimensional cylinder-intersection

$$C_k(R_1, \ldots, R_k) = \left\{ (x_{n-k+1}, \ldots, x_n) \in \mathbb{R}^k,\ \forall j \leq k, \sum_{l=1}^{j} x_{n-l+1}^2 \leq R_j^2 \right\}$$

Pruning method may mistakenly cut off the branch which contains the right solution, and turns enumeration from a deterministic algorithm to a probabilistic one. The pruning function $\{R_i\}_{i=1}^{n}$ defines the searching region, and consequently influences both the size of enumeration tree and success probability. Gama and Nguyen [25] provided a detailed analysis on the success probability $p_{succ}$ and speedup of various pruning functions, and proposed extreme pruning method to minimize the enumeration cost.

## 2.3 Discrete Pruning

In classical enumeration, we search for lattice points directly according to their coordinates $(x_1, \ldots, x_n)$ 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 an intuitive idea to search for lattice vector with 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 [41] and dramatically improved by Fukase and Kashiwabara [21]. These sampling strategies are summarized by Aono and Nguyen, and they defined *lattice partition* to generalize these sampling methods.

**Definition 3 (Lattice partition [8]. )** *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 exists an 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 non-trivial partition is generally related to the orthogonal basis $\mathbf{B}^*$. Some examples of it are given in [8]. Here we only introduce natural partition which is first proposed by Fukase and Kashiwabara [21], since it has smaller moments than other lattice partition 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 basing on natural partition, following the work of [8, 10, 21].

**Definition 4** *Given a lattice $\mathcal{L}$ with 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 $\mathbf{t} = (t_1, \ldots, t_n) \in \mathbb{N}^n$ 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 $j = 1, \ldots, n$. Then natural number representation $\mathbf{t} \in \mathbb{N}^n$ leads to the natural partition $(\mathcal{C}(), \mathbb{N}^n)$ by defining*

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

The shape of $\mathcal{C}(\mathbf{t}) = \left\{\sum_{i=1}^{n} x_i \mathbf{b}_i^*, -\frac{t_i+1}{2} < x_i \le -\frac{t_i}{2} \text{ or } \frac{t_i}{2} < x_i \le \frac{t_i+1}{2}\right\} \subsetneq \mathbb{R}^n$ is a union of $2^j$ hyper-cuboids (boxes) which are centrosymmetric and disjoint, where $j$ is the number of non-zero 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 lattice vector in $\mathcal{C}(\mathbf{t})$ always shows a kind of randomness. We can naturally assumed 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 [41] and generalized by Fukase and Kashiwabara [21]. Aono, Nguyen and Shen [8, 10] also use this assumption by default.

**Assumption 2 (Randomness Assumption [21])** *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 the expectation and variance value of $\|\mathbf{v}\|^2$, since $\|\mathbf{v}\|^2$ can also be considered as a random variable. They are also defined as the first and second moment of cell $\mathcal{C}(\mathbf{t})$ [8]:

$$E[\mathcal{C}(\mathbf{t})] = \frac{1}{12} \sum_{j=1}^{n} (3t_j^2 + 3t_j + 1)\|\mathbf{b}_j^*\|^2 \tag{2.2}$$

$$Var[\mathcal{C}(\mathbf{t})] = \sum_{j=1}^{n} \left(\frac{t_j^2}{48} + \frac{t_j}{48} + \frac{1}{180}\right)\|\mathbf{b}_j^*\|^4 \tag{2.3}$$

This means 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 cost a rather long time. This leads to the core idea of 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 exists a $\mathbf{v}$ such that $\|\mathbf{v}\| \le R$. The pruning set is $P = \cup_{\mathbf{t}\in S}\mathcal{C}(\mathbf{t})$. If randomness assumption and Gaussian heuristic holds, 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.

## 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, and therefore they can apply equation (2.2) to predict vector length. However, we can easily point out a

**Algorithm 1** Discrete Pruned Enumeration

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

```
1: Reduce lattice basis $\mathbf{B}$;
2: while true do
3:   $S \leftarrow \varnothing$;
4:   Use binary search to find bound $r$ such that there are $M$ tags $\mathbf{t}$ satisfying $f(\mathbf{t}) < r$;
5:   Enumerate all these $M$ tags and save them in set $S$
6:   for $\mathbf{t} \in S$ do
7:     Decode $\mathbf{t}$ to recover the corresponding $\mathbf{v}$ such that $\mathbf{v} \in \mathcal{C}(\mathbf{t})$ ;
8:     if $\|\mathbf{v}\|^2 < R^2$ then
9:       return $\mathbf{v}$;                                   // Find a solution
10:    end if
11:  end for
12:  Rerandomize $\mathbf{B}$ by multiply a light-weight unimodular matrix on it or rearrange the vectors in
     basis;
13:  Reprocess $\mathbf{B}$ using lattice reduction algorithms such as BKZ or LLL;
14: end while
```

paradox if the 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 same length. However, the equation (2.2) 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 this two cells have different success probability, 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*

$$
\begin{aligned}
u_k &= \begin{cases} \dfrac{-t_k}{2}, & \text{if } t_k \text{ is even} \\ \dfrac{t_k+1}{2}, & \text{if } t_k \text{ is odd} \end{cases} \\
u_{k+1} &= \ldots = u_n = 0
\end{aligned}
\tag{3.1}
$$

*Proof.* We can verify the proposition through the procedures of algorithm 4, 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 non-zero coefficient of $\mathbf{x}$ is $x_k$ if and only if $u_k$ is the last non-zero 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 tag with odd and even number in the last non-zero coefficient are called "odd-ended tag" and "even ended tag" respectively. □

Based on proposition 1, the rectified randomness assumption is given below, and therefore 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 non-zero 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* (3.1).

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 corresponded $\|\mathbf{v}\|^2$ is:

$$E'[\mathcal{C}(\mathbf{t})] = \frac{1}{12}\sum_{j=1}^{k-1}(3t_j^2 + 3t_j + 1)\|\mathbf{b}_j^*\|^2 + u_k^2\|\mathbf{b}_k^*\|^2 \tag{3.2}$$

where $u_k$ is defined by equation (3.1).

After the rectification, for two tags $\mathbf{t}, \mathbf{t}'$ which only differs by 1 in the last non-zero coefficient, now 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

The crucial step of algorithm 1 is called *cell enumeration* (line 5), aiming to find the "best" $M$ cells. We use objective function $f(\mathbf{t})$ to measure how good the cells are. $E[\mathcal{C}(\mathbf{t})]$ (eq. (2.2)) is a tacit indicator for searching proper $\mathbf{t}$ since it is exactly the expected squared length of lattice vector $\mathbf{v} \in \mathcal{C}(\mathbf{t})$. Aono and Gama [8] directly use $E[\mathcal{C}(\mathbf{t})]$ as the objective function $f(\mathbf{t})$ in cell enumeration, and Aono et al. [10] use a modified version of $E[\mathcal{C}(\mathbf{t})]$ in order to guarantee its polynomial running time. They requires 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, now we proposed a modified objective function. Given a tag vector $\mathbf{t} = [t_1, \ldots, t_k \neq 0, 0, \ldots, 0]$ as input, first define

$$f(i, t_i) \overset{\text{def}}{=} \begin{cases} 0, & \text{for } i > k \\ u_i^2\|\mathbf{b}_i^*\|^2, & \text{for } i = k \\ \frac{1}{4}(t_i^2 + t_i)\|\mathbf{b}_i^*\|^2, & \text{else} \end{cases} \tag{3.3}$$

where $u_i$ is defined by equation (3.1). Then the objective function of cell enumeration is

$$f(\mathbf{t}) \overset{\text{def}}{=} \sum_{i=1}^{n} f(i, t_i) = \frac{1}{4}\sum_{i=1}^{n}(t_i^2 + t_i)\|\mathbf{b}_i^*\|^2 = \frac{1}{4}\sum_{i=1}^{k}(t_i^2 + t_i)\|\mathbf{b}_i^*\|^2 + u_i^k\|\mathbf{b}_k^*\|^2 \tag{3.4}$$

Then the complete procedures of cells enumeration is given below:

**Algorithm 2** CellENUM

**Input:** Orthongonal basis $\mathbf{B}^* = [\mathbf{b}_1^*, \ldots, \mathbf{b}_n^*]$, $r$
**Output:** All $\mathbf{t} \in \mathbb{N}^n$ such that $f(\mathbf{t}) \leq r$ where $f(\mathbf{t})$ as defined in equation (3.4), and $\mathbf{t}$ is even-ended

```
1: $t_1 = t_2 = \ldots = t_n = 0$;
2: $c_1 = c_2 = \ldots = c_{n+1} = 0$;
3: $k \leftarrow 1$
4: while true do
5:   $c_k \leftarrow c_{k+1} + f(k, t_k)$;
6:   if $c_k < r$ then
7:     if $k = 1$ then
8:       return $\mathbf{t} = (t_1, t_2, \ldots, t_n)$;
9:       if $t_{k+1} = \ldots = t_n = 0$ then
10:        $t_k \leftarrow t_k + 2$;                    // Only output "even-ended" tags
11:      else
12:        $t_k \leftarrow t_k + 1$;
13:      end if
14:    else
15:      $k \leftarrow k - 1$;
16:      $t_k \leftarrow 0$;
17:    end if
18:  else
19:    $k \leftarrow k + 1$;
20:    if $k = n + 1$ then
21:      exit;
```

```
22:     else
23:       if $t_{k+1} = \ldots = t_n = 0$ or $k = n$ then
24:         $t_k \leftarrow t_k + 2$;
25:       else
26:         $t_k \leftarrow t_k + 1$;
27:       end if
28:     end if
29:   end if
30: end while
```

*Remark 1.* Considering the symmetry of lattice vector, we only search for even-ended tags (line 10 and line 24).

*Remark 2.* For the last non-zero coefficient $t_k$, the value $f(k, t_k) = u_k^2\|\mathbf{b}_k^*\|^2$ changes very slightly comparing with the original definition $f(k, t_k) = \frac{1}{4}(t_k^2 + t_k)\|\mathbf{b}_k^*\|^2$ indicated by [10], since $\|\mathbf{b}_k^*\|^2$ is very small according to GSA. Therefore the modification of objective function will not make a perceptible effect on the output of algorithm 5. But we note that when using discrete pruned enumeration in cases where the GSA does not hold, this modification might provide a better result.

The time complexity of algorithm 2 is similar to that of [10]: The number of times that algorithm 2 enter the while 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 [10]. [2]

In cell enumeration, a bound $r$ should be determined in previous such that there are exactly $M$ tags satisfying $f(\mathbf{t}) \leq r$. Aono and Nugyen [8] first proposed the idea to use binary search method to find a proper $r$. Aono et al. [10] gave a detailed binary search algorithm (Alg. 5 in [10]) which is proved to be have polynomial running time. Their binary search method output $N$ tags in $O(n(n + 1)M) + n^{O(1)} + O(\log_2 M)$ where $M \leq N \leq (n + 1)M$. Then we have to find $M$ tags with the smallest $f(\mathbf{t})$ value, which means an extra sorting algorithm with time complexity $O(N \log N)$ should be run. When $N$ is very large, the time consumption of sorting could not be ignored.

We provide a simple and practical polynomial-time binary search strategy to determine the parameter $r$ for cell enumeration. This method guarantees that cell enumeration algorithm (Alg. 5) outputs about $(1-\epsilon)M$ to $(1+\epsilon)M$ tags $\mathbf{t}$ satisfying $f(\mathbf{t}) < r$. When $\epsilon$ is small, those tags can be directly used without an extra sorting.

**Algorithm 3** ComputeRadius

```
Input:  $M, \epsilon$
Output: $r \in \mathbb{R}$ 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:    $A \leftarrow$CellENUM_probe$(R_m, M, \epsilon)$   // This function returns an integer $A \in \{-1, 0, 1\}$
6:    if $A = 1$ then
7:      $R_l \leftarrow R_m$                          // $R_m$ is too large
8:    else if $A = -1$ then
9:      $R_r \leftarrow R_m$                          // $R_m$ is too small
10:   else
11:     return $r \leftarrow R_m$                     // $R_m$ is acceptable
12:   end if
13: end while
```

In algorithm 3, the subalgorithm CellENUM_probe() called in line 3 is a modification of cell enumeration (Alg. 2). It only counts the number of qualified tags and returns different values to indicate

[2]Noted that although we change the definition of $f(i, t_i)$ and therefore change the value calculated in line 3, it 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 $\sum_{i=1}^{n} f(i, t_i) \leq R$.

whether the bound is too large or too small. The pseudocode of CellENUM_probe() is given in appendix A.1.

Theorem 1 gives the asymptotic time complexity of algorithm 3:

**Theorem 1** *Given lattice $\mathcal{L}(\mathbf{B})$, $M$, a relaxation factor $\epsilon$, algorithm 3 ends within $O\left(\log n + \log \frac{1}{\epsilon} + n \log \left(n det(L)^{\frac{2}{n}}\right)\right)$ 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 9 is called exactly once.*

The approximate proof of theory 1 is given in appendix A.2. In the following experiments, we set $\epsilon = 0.005$. Although the binary search of us and that in [10] (Alg. 5) both have the same dominant term $n^2 M$ in asymptotic time complexity, in experiments we test them in a range of $60 \leq n \leq 200$ and $10^5 \leq M \leq 10^6$, and find that our method is generally at least $2 \sim 3$ times faster than that in [10].

## 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 [8,21]. 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 vector. 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 $\rho = \sum\limits_{k=i}^{n} |x_k + \sum\limits_{i=k+1}^{n} \mu_{ik} x_i|^2 \|\mathbf{b}_k^*\|^2$ is larger than the target length of SVP, we stop the decoding since it is no way a short lattice vector. Therefore we avoid a vector-matrix multiplication with time $O(n^2)$.

**Algorithm 4** Decode($\mathbf{t}, R, \alpha, \mathbf{B}$)

**Input:** tag $\mathbf{t} \in \mathbb{N}^n$, SVP target length $R = 1.05 \cdot \mathrm{GH}(L)$, orthogonalization information $\mathbf{U} = (\mu_{i,j})_{n \times n}$, $\mathbf{B}^* \in \mathbb{R}^{n \times n}$ ;
**Output:** 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
23:    return $\mathbf{v} = \mathbf{B}\mathbf{u}$
24: else
25:    return $\varnothing$
26: end if
```

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

## 3.4 Lattice Reduction for reprocessing

### 3.4.1 BKZ with Fixed Tours

To solve an SVP instance, the DP enumeration should be repeatedly run on many different bases, which means the lattice basis should be re-randomized when DP enumeration restarts, and hence it should be re-processed to maintain good quality. A plain method is to use the polynomial-time LLL reduction as reprocessing method, which only guarantees some primary properties and is not as good as BKZ reduction. However, a complete BKZ reduction will take a rather longer time, and estimating its running time is sophisticated. Besides, the BKZ algorithm produces diminishing returns: after the first dozens of iterations, the quality of basis, such as the root Hermite factor, changes slowly during iterations, which was illustrated in [13, 16].

As a matter of fact, 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 [27] is a good attempt to decrease the number of iterations of BKZ reduction while maintaining some good properties. However, the running time of BKZ is still hard to estimate since the number of iterations is not fixed, and those properties mainly describe the shortness of $\mathbf{b}_1$ and give little help to our cost estimation.

We proposed to use "k-tours-BKZ" (algorithm 5) as our reprocessing method, which is efficient and also beneficial to running time analysis. The k-tours-BKZ algorithm restricts the total number of "tours" (line 4 - 17 in Alg. 5) of BKZ within $k$. Given BKZ blocksize $\beta$ and $k$, the time consumption of Alg. 5 can be approximately estimated by multiplying $k(n-\beta)$ and the cost of solving a single $\beta$-dimensional SVP oracle. This will be explained in Section 4.

**Algorithm 5** k-tours-BKZ

**Input:** Lattice basis$\mathbf{B}$; BKZ blocksize $\beta$, $k$;
**Output:** a re-processed lattice basis $\mathbf{B}'$;

```
1:  $Z \leftarrow 0$; $i \leftarrow 0$;        // $Z$ is used to judge the termination condition for original BKZ;
2:  $K \leftarrow 0$;                            // $K$ records the tours;
3:  LLL($\mathbf{b}_1, \dots, \mathbf{b}_n$);
4:  while $Z < n-1$ or $K < k$ do
5:      $K++$
6:      $i \leftarrow i++ \mod (n-1)$;
7:      $j \leftarrow \min(i+\beta-1, n)$;
8:      $h \leftarrow \min(j+1, n)$;
9:      $\mathbf{v} \leftarrow$ ENUM($\pi_i(\mathbf{b}_i), \dots, \pi_i(\mathbf{b}_j)$);      // call the SVP oracle
10:     if $\mathbf{v} \neq \mathbf{0}$ then
11:         $Z \leftarrow 0$;
12:         LLL( $\mathbf{b}_1, \dots, \sum_{i=1}^{n} v_i \mathbf{b}_i, \dots, \mathbf{b}_h$ );
13:     else
14:         $z++$;
15:         LLL( $\mathbf{b}_1, \dots, \mathbf{b}_h$);
16:     end if
17: end while
```

To re-randomize the basis, instead of multiplying basis matrix by a unimodular matrix, we shuffle the basis vectors by randomly swapping basis vectors for $\min(8, \lceil 0.12n \rfloor)$ times. In practice we find that this method can guarantee a new basis matrix after reprocessing, meanwhile, it will not destroy the good quality of basis too much, and therefore can help the k-tours-BKZ achieve a stable basis quality again in only few tours, which means we can set $k$ very small, such as $k = 8$ or less, to save time for reprocessing procedure.

### 3.4.2 Average Quality of Lattice Basis During reprocessing

Even using the same parameters, DP enumeration will have different running time and success probability on different bases of a lattice. We expect the lattice basis should have a stable quality that will not be changed by reprocessing operation, and can be easily simulated or predicted without conducting a real lattice reduction. The very first work is to define what is the quality of a lattice basis, and study how it changes during enumeration loops. We choose three indicators to describe the quality of a basis and give observation on their behavior during reprocessing.

**Root Hermite Factor.**

In the studies of BKZ algorithm, the root Hermite factor $\delta$ 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:

$$\delta \overset{\text{def}}{=} \left( \frac{\|\mathbf{b}_1^*\|}{(\mathrm{vol(L)})^{1/n}} \right)^{1/n} \tag{3.5}$$

Gama and Nguyen [24] pointed out a phenomenon that for BKZ algorithm, when blocksize parameter $\beta \leqq n$, the root Hermite factor is only affected by the blocksize parameter $\beta$, but has no relationship with lattice dimension. More observation of root Hermite factor is given in [16], Table 2.

We generate a $n = 120$ dimensional SVP challenge basis at random, and show how the k-tours-BKZ change the root Hermite factor $\delta$ of the basis during reprocessing. As shown in figure 1, on a BKZ$_\beta$-reduced basis, the lattice basis is re-randomized and re-processed by k-tours-BKZ for every $k = 6$ tours with blocksize $\beta$, and $\delta$ hardly changes.

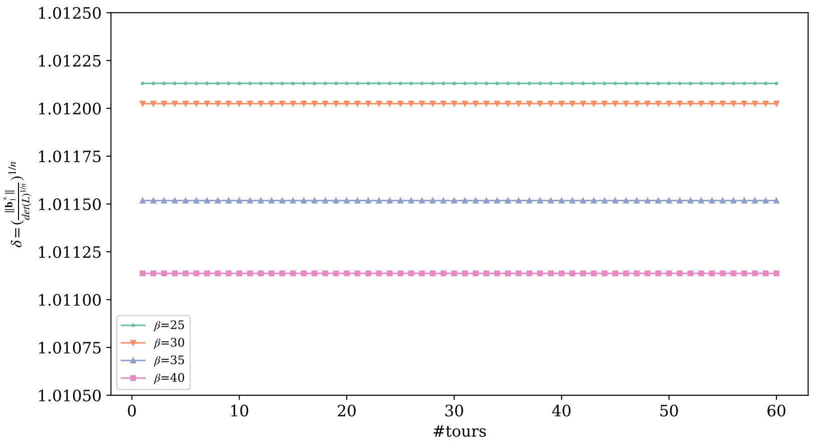

Figure 1: The Evolution of $\delta$ During reprocessing, $n = 120$, $k = 6$, $\beta = 25, 30, 35, 40$

**Gram-Schmidt Sum.**

For DP-enumeration, its success probability is tightly related with the lengths of Gram-Schmidt basis vectors $\{\|\mathbf{b}_1^*\|, \ldots, \|\mathbf{b}_n^*\|\}$, and Fukase et al. [21] proposed to use *Gram-Schmidt Sum* as a measurement of lattice basis quality, which is defined as

$$\mathrm{GSS}(\mathbf{B}) \overset{def}{=} \sum_{i=1}^{n} \|\mathbf{b}_i^*\|^2 \tag{3.6}$$

They also gave an intuitive and approximate analysis of the relation between GSS($\mathbf{B}$) and the efficiency of finding short lattice vector. Figure 2 shows the evolution of GSS($\mathbf{B}$). In general, when lattice basis is iteratively re-processed, the value of GSS($\mathbf{B}$) only has mild fluctuation and does not increase significantly. which implies that the success probability of finding very short lattice vectors is quite stable.

**Geometry Series Assumption and GS Slope.**

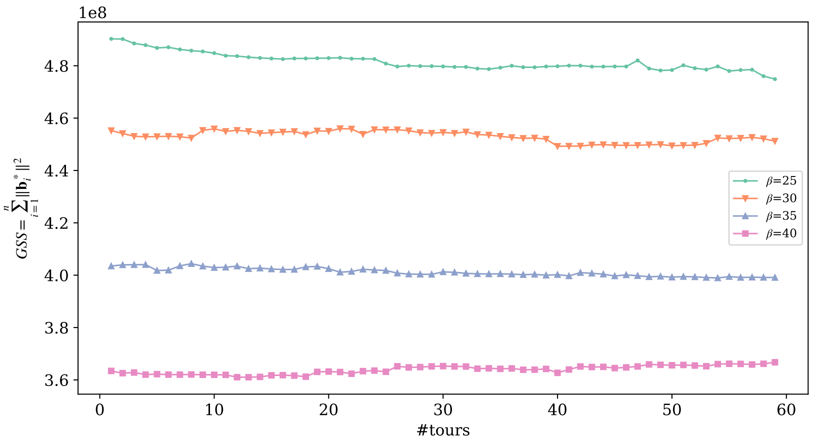

Figure 2: The Evolution of GSS($\mathbf{B}$) During reprocessing, $n = 120$, $k = 6$, $\beta = 25, 30, 35, 40$

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

$$\log \|\mathbf{b}_i^*\| = (i-1)\cdot \log q + \log \|\mathbf{b}_1\|,\ \ i = 1, \dots, n \tag{3.7}$$

If GSA holds, the points $\{(i, \log \|\mathbf{b}_i^*\|)\}_{i=1}^n$ forms 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 $q$ 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. Figure 3 shows the change of fitted $q$ in reprocessing. As in the case of $\delta$ and $GSS(\mathbf{B})$, $q$ also shows a stable trend during the iterative re-randomization and BKZ tours.

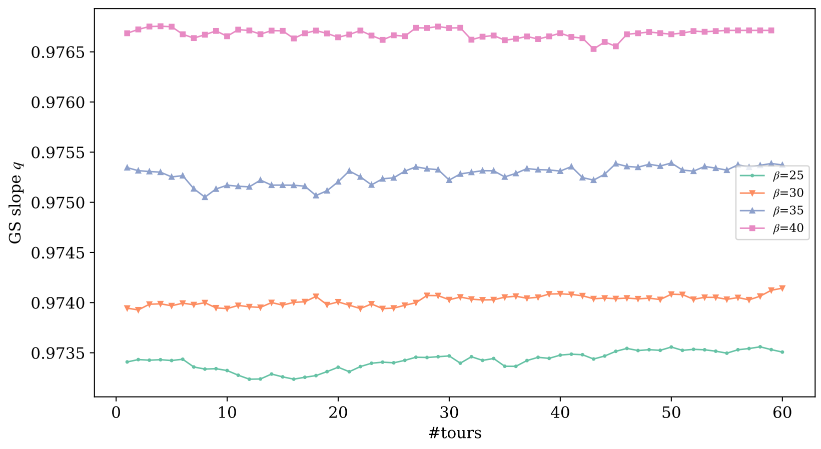

Figure 3: The Evolution of GS slope $q$ During reprocessing, $n = 120$, $k = 6$, $\beta = 25, 30, 35, 40$

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

## 4 Precise Cost Estimation of DP-ENUM

The precise cost estimation of DP enumeration calls for a great concern and is still an open problem for cryptanalysis. However, there are several obstacles to build a good running time model which can be consistent with the experimental results of viable dimension and also can be extrapolated to very high dimension.

First, DP enumeration contains many sub-algorithms 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 running time of DP enumeration still needs to be handled carefully. These sub-algorithms involves a variety of arithmetic operations and logic operations, which makes it hard to define a universal "basic operation" for all the procedures of DP enumeration. To build a running time model for DP enumeration, our key idea is to use CPU-cycles as the basic operation unit, since it can avoid the differences caused by different types of operations, and also easy to be counted.

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 pruning set, which directly determine the success probability for pruning. Aono et al. proposed to use FILT to compute the volume of its pruning set [8], 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 4 parts:

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

Denote the success probability of 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 re-randomizing, then the expected number of loops is about $\frac{1}{p_{succ}}$ according to geometric distribution, and the total running time $T_{total}$ of DP enumeration can be estimated by

$$T_{total} = T_{pre} + \frac{T_{repro} + T_{bin} + T_{cell} + M \cdot T_{decode}}{p_{succ}} \tag{4.1}$$

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

For all the experiments in this paper, the computing platform is a server which has Intel Xeon E5-2620 CPUs with 8 physical cores running at 2.10 GHz, and 64GB RAM. To obtain accurate CPU cycles for DP enumeration algorithm, we fixed the CPU basic frequency and set CPU affinity, and all the time data for fitting is obtained from our single-thread implementation.

## 4.1 Simulation of Lattice Basis

Some parts in the total time cost model eq. (4.1) are hugely dependent on the quality of basis, more precisely, the vector lengths of Gram-Schmidt orthogonal basis $\{\|\mathbf{b}_i^*\|\}_{i=1}^n$. Based on the reprocessing method we proposed and the stability analysis in Section 3.4, we can reasonably assume the Gram-Schmidt sequence $\{\|\mathbf{b}_i^*\|\}_{i=1}^n$ will not change severely in the loops of DP enumeration. Then the issue is to simulate an "average" GS sequence for a BKZ$_\beta$ reduced basis. BKZ simulator [13, 16] is a universal method to this problem especially when $\beta$ is quite large. We will give another simpler and faster method, which is more suitable for DP enumeration since the blocksize of BKZ does not have to be very large for preprocessing.

If we combine

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

and equation (3.7), then we have

$$n \cdot \log \|\mathbf{b}_1\| + \frac{n(n-1)}{2} \log q = \log(\text{vol}(\mathcal{L})) \tag{4.2}$$

By equation (4.2), 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 of the Gram-Schmidt orthogonal basis. Besides, only 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 [13] indicates that the $\|\mathbf{b}_1\|$ can be obviously smaller than the prediction given by GSA, especially when $\beta$ is large. The question that remains is to give an average estimation of $q$ for a given lattice and BKZ parameter $\beta$.

Actually we find the GS slope has a property similar to the root Hermite factor: the GS slope of a reduced basis is only related to blocksize $\beta$ but not to lattice dimension, especially when $\beta \ll n$. [3] For each parameter set $(n, \beta) \in \{102, 105, \ldots, 150\} \times \{15, 17, \ldots, 39\}$, we generate 50 random SVP challenge instances and apply BKZ$_\beta$ on the $n$-dimensional lattice basis. Then we use least square method to fit $\log q$ of reduced bases. Figure 4 show the relationship between $q$ and lattice dimension $n$, indicating that $q$ hardly dependent on lattice dimension $n$, and only varies with $\beta$.

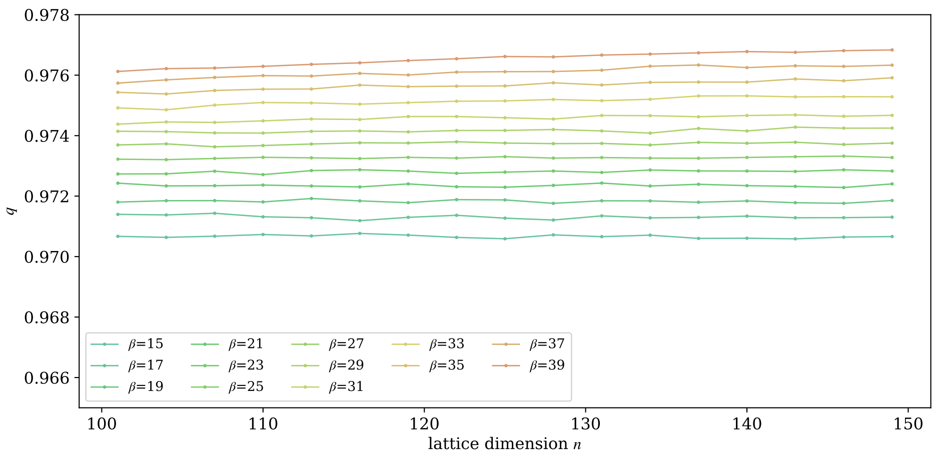

Figure 4: the relation between $q$ and lattice dimension $n$

Figure 5 illustrates the positive correlation of $q$ and $\beta$, which is consistent with the intuition: larger blocksize $\beta$ makes the lattice basis better, and the GS slope is milder, which means $q < 1$ goes closer to 1. Under GSA, we assume $q$ and $\beta$ satisfy $q = 1 - \exp(-a\beta + b)$, then the fitting function is

$$q_\beta = 1 - \exp(-0.0092200\beta - 3.3919) \tag{4.3}$$

The fitting curve is also illustrated in figure 5.

Table 1 gives estimated value of $q_\beta$.

| $\beta$ | 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 |
| $\beta$ | 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 |

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

By using $q_\beta$, 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 equations:

$$\begin{cases} \mathrm{vol}(\mathcal{L}) = \prod_{i=1}^n B_i \\ n \cdot \log B_1 + \frac{n(n-1)}{2} \log q_\beta = \log(\mathrm{vol}(\mathcal{L})) \\ \log B_i = (i-1) \cdot \log q_\beta + \log B_1,\ i = 1, \ldots, n \end{cases} \tag{4.4}$$

[3]In the experiments we heuristically restrict $\beta < n^{3/4}$.

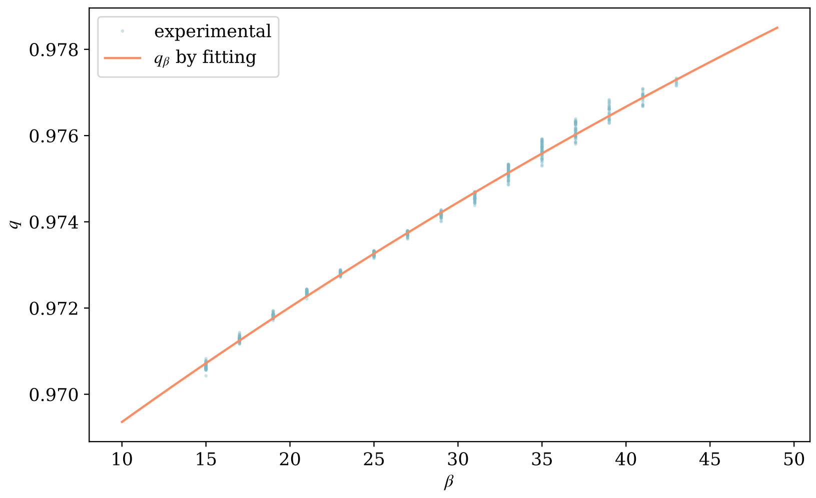

Figure 5: the relation between $q$ and BKZ blocksize $\beta$

## 4.2 Cost of Sub-algorithms

**Cost of Binary Search and Cell Enumeration.**

For cell enumeration algorithm (Alg. 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

$$T_{cell} \approx 2.4339nM + 108.74M - 17455n + 1334139 \tag{4.5}$$

For binary search algorithm (Alg. 3), theory 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 also retain accuracy, we only take the dominant term of the complete expansion, which is $n\log\left(n\det(L)^{\frac{2}{n}}\right) \cdot (2n-1)M$. In particular, for SVP challenge lattice, we have $\mathrm{vol}(\mathcal{L}) \sim 2^{10n}$. Then the fitting function of $T_{bin}$ is

$$T_{bin} \approx 0.11341Mn^2 + 13.155Mn\log n + 265.65M - 84679n + 15455380 \tag{4.6}$$

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

**Cost of Decoding.**

To decode one tag by running algorithm 4, the number of entering the for loops is $O(n)$, and in each loop it 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 generate 100 SVP lattices at random for each $n$. The expected running time $T_{decode}$ of decoding algorithm is fitted by

$$T_{decode} = 0.39045n^2 + 167.06n - 4350.4 \tag{4.7}$$

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

**Cost of reprocessing.**

The cost of 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 [1]. In one tour of BKZ$_\beta$, the total running time 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:

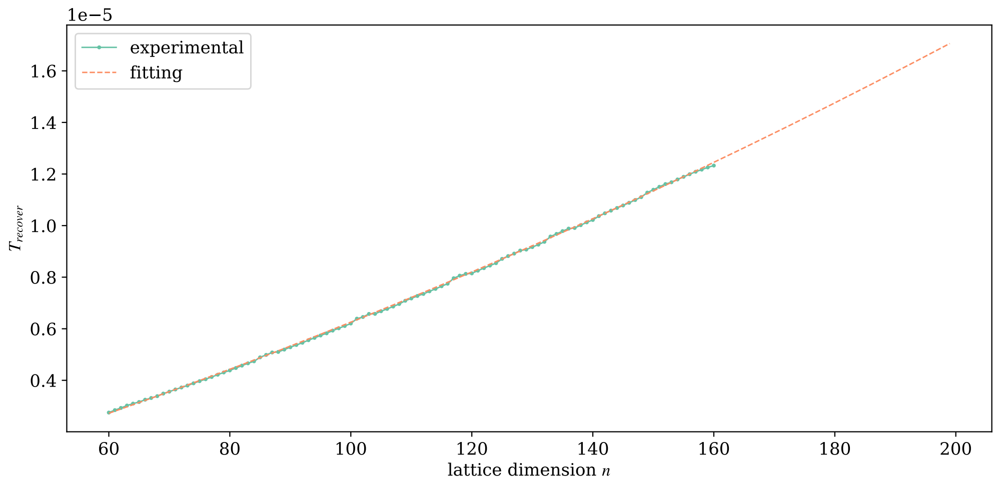

Figure 6: the relation between $T_{decode}$ and dimension $n$

$$\text{BlockCost}(i,j) = \text{BlockProcess}(j,n,\log A) + C_{node} \cdot \text{EnumCost}(i,j) \tag{4.8}$$

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

$$\text{BKZCost}(L) = k \cdot \sum_{i=1}^{n-1} \text{BlockCost}(i,j),\ \ j = \min(n, i+\beta-1) \tag{4.9}$$

In equation (4.8), BlockProcess$(j,n,\log A)$ is the cost of updating basis (Alg 5, line 12). The asymptotic time complexity of this part is $O(j^3 m \log A)$ which is mainly donated by LLL reduction [44], where $A \lesssim 2^{10n}$ for SVP challenge lattice. When $\beta$ is small, the cost of updating cannot be ignored. As 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 $C_{node} \approx 200$ [11]; EnumCost$(i,j)$ is the total amount of nodes that need to be traversed to find a short vector on $\mathcal{L}[i,j]$.

Let $n = 60,\ldots,150$, $\beta = 11,13,\ldots,43$ and $k=6$, we record the cost of each stages (including running time 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

$$\text{BlockProcess}(j,n,\log A) \approx 0.000904381 \times j^3 n^2 + 28752188 \tag{4.10}$$

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

$$\text{FullEnumCost}(i,j) = \frac{1}{2}\sum_{k=1}^{j-i+1} \frac{V_k(\|\mathbf{b}_i^*\|)}{\prod_{\ell=j-k+1}^{j} \|\mathbf{b}_\ell^*\|} \tag{4.11}$$

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 getting an explicit expression of $r_{\beta'}$. [4]

[4]An alternative (lowerbound) estimation of enumeration cost is provided by Chen and Nguyen [16]. The coefficients of their model are given in LWE estimator [6] . However their model is more suitable for $\beta$ is very high, and not very precise when the blocksize is small.

$$r_{\beta'} = \frac{\text{FullEnumCost}_{\beta'}}{\text{ExtremeEnumCost}_{\beta'}} \tag{4.12}$$

The value of $\text{FullEnumCost}_{\beta'}$ can be calculated by equation (4.11) with GS sequence $\{\|\mathbf{b}_i^*\|\}_{i=1}^n$, and the actual number of enumeration nodes $\text{ExtremeEnumCost}_{\beta'}$ is obtained from experiments. We recorded the number of enumeration nodes $\text{ExtremeEnumCost}_{\beta'}$ to calculate the speedup ratio data. For fitting $r_{\beta'}$, 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 are gathered and averaged. It is well known that the extreme pruning can offer an exponential speedup [25], and the tightening of radius also leads to a super-exponential speedup. We assume $r_\beta \sim \exp O(\beta' \log \beta')$, and by fitting we have

$$\log r_\beta = 0.35461\beta \log \beta - 1.5331\beta + 4.8982 \log \beta - 2.9084 \tag{4.13}$$

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

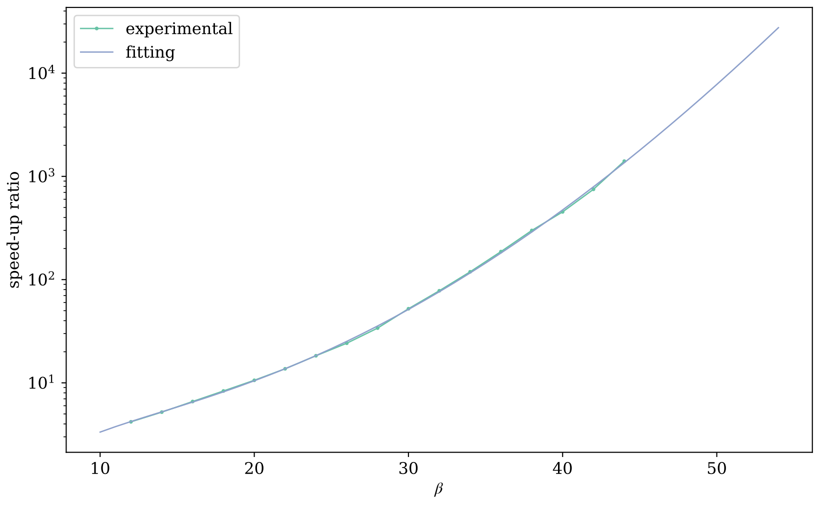

Figure 7: Speed-up Ratio of Extreme Pruning

To predict EnumCost$(i, j)$ without any information of a specific lattice basis, the GS sequence contained equation (4.12) should be replaced by the simulated values $\{B_1, B_2, \ldots, B_n\}$ derived by equations set (4.4).

**Algorithm 6** Calculating $T_{repro}$

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

```
1: $q \leftarrow 1 - \exp(-0.0092200\beta - 3.3919)$;    // Eq. (4.3)
2: $\log B_1 \leftarrow \frac{1}{n}\left(\log(\text{vol}(\mathcal{L})) - \frac{1}{2}n(n-1)\right)$    // Eq. (4.2)
3: $\log B_i \leftarrow (i-1)\cdot \log q + \log B_1$;    // Eq. (3.7)
4: $i \leftarrow 0$
5: $Cost \leftarrow 0$
6: while $i = 1$ to $n-1$ do
7:     $\beta' = \min(\beta, n-i+1)$
8:     $r_{\beta'} \leftarrow \exp(0.35461\beta \log \beta - 1.5331\beta + 4.8982 \log \beta - 2.9084)$    // Eq. (4.13)
9:     $Cost = Cost + \frac{1}{r_{\beta'}} \cdot \text{FullENUMCost}(B_i, \ldots, B_{i+\beta'-1}) + \text{BlockProcess}(i+\beta'-1, n)$    // by
       repalcing $\|\mathbf{b}_i^*\|$ with $B_i$ in equation(4.11)
10: end while
11: return $k \cdot C_{node} \cdot Cost$
```

## 4.3 Success Probability

Under Gaussian Heuristic, the success probability of pruned enumeration can be directly reduced to computing the volume of pruning set. For discrete pruning, the shape of pruning set has always been considered as a union of "ball-box intersections", which is not easy to compute. Aono et al. [8] 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 they use stratified sampling to deduce the total volume of the union.

However, the flaw in original randomness assumption (assumption 2) also causes the flaw in success probability model of 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 probability according to the model given by [8]. However, the lattice vectors contained in $\mathcal{C}(\mathbf{t})$ and $\mathcal{C}(\mathbf{t}')$ have exactly the same length.

Figure 8 depicts the paradox in 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, and compute $p_{succ,even}$ of their corresponding even-ended cells, both using the method provided by [8]. Then we run 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 gap between expectation value $1/p_{succ,odd}$ and $1/p_{succ,odd}$, which are estimated using odd-ended and even-ended cells respectively.

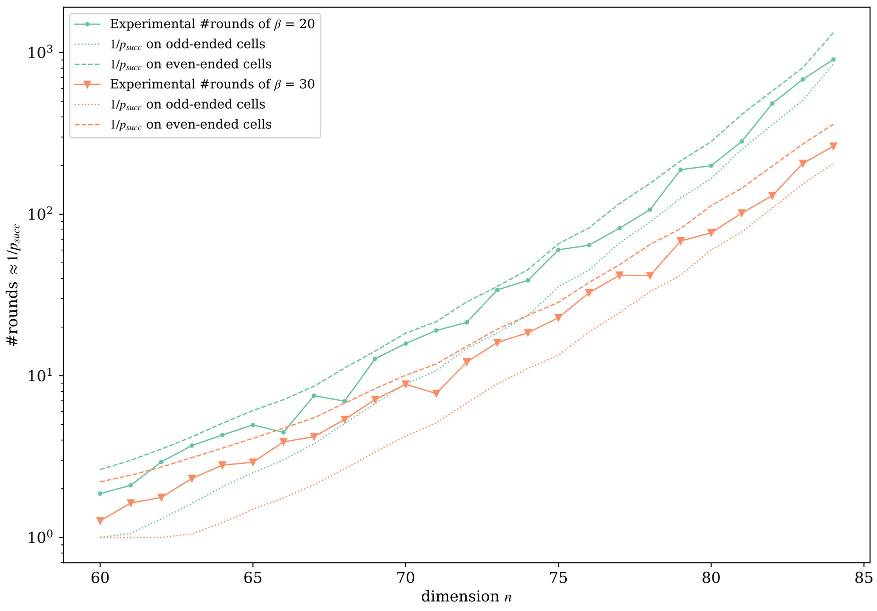

Figure 8: The difference of $p_succ$ estimated with odd-ended cells and even-ended cells

This phenomenon calls for a proper rectification on 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 hyper-plane contained in $\mathcal{C}(\mathbf{t}) \cup \mathrm{Ball}_n(R)$, 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*

$$\mathcal{C}'(\mathbf{t}) \overset{def}{=} \left\{ \sum_{i=1}^{n} x_i \mathbf{b}_i^*,\ x_i \in \mathbb{R} \text{ and } \begin{cases} x_i \in (-t_i + 1/2, -t_i/2] \cup (t_i/2, t_i + 1/2] & \textit{for } i < k \\ x_k = u_k & \textit{as defined in eq. (3.1)} \\ x_{k+1} = \ldots x_n = 0 \end{cases} \right\} \tag{4.14}$$

This assumption gives a more precise distribution of lattice vector in the cell. In fact $\mathcal{C}'(\mathbf{t})$ is the union of $2^{k-1}$ $k-1$ 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 $\mathbf{t} = [t_1, \dots, t_k \neq 0, 0, \dots, 0]$, 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

$$\begin{aligned} p_{succ}(\mathbf{t}) &\overset{def}{=} \underset{\mathbf{v} \leftarrow \mathcal{C}'(\mathbf{t})}{\mathrm{Prob}} \left(\|\mathbf{v}\|^2 \leq R^2\right) \\ &= \underset{\mathbf{v} \leftarrow \mathcal{C}'(\mathbf{t})}{\mathrm{Prob}} \left( \sum_{i=1}^{k-1} u_i^2 \|\mathbf{b}_i^*\|^2 < (R^2 - u_k^2 \|\mathbf{b}_k^*\|^2) \right) \\ &= \frac{\mathrm{vol}\left( Ball_{k-1}(\sqrt{R^2 - u_k^2 \|\mathbf{b}_k^*\|^2}) \cap \mathcal{C}_{k-1}(\mathbf{t}) \right)}{\mathrm{vol}(\mathcal{C}_{k-1}(\mathbf{t}))} \\ &= \frac{\mathrm{vol}\left( Ball_{k-1}(\sqrt{R^2 - u_k^2 \|\mathbf{b}_k^*\|^2}) \cap \mathcal{C}_{k-1}(\mathbf{t}) \right)}{\prod_{i=1}^{k-1} \|\mathbf{b}_i^*\|^2} \end{aligned} \tag{4.15}$$

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 (4.15) can be written as

$$\begin{aligned} &\mathrm{vol}\left(Ball_{k-1}(R') \cap \mathcal{C}_{k-1}(\mathbf{t})\right) \\ =&2^{k-1} \cdot R'^{k-1} \cdot \prod_{i=1}^{k-1} (\beta_i - \alpha_i) \cdot \underset{(x_1, \dots, x_{k-1}) \leftarrow \prod_{i=1}^{k-1} [\alpha_i, \beta_i]}{Pr} \left\{ \sum_{i=1}^{k-1} x_i^2 \leq 1 \right\} \end{aligned} \tag{4.16}$$

Then the calculation of $p_{succ}(\mathbf{t})$ is reduced to computing the sum distribution of $k-1$ independent and identically distributed variables $x_1^2, \dots, x_{k-1}^2$, which can be approximated by 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 sub-algorithm 5 in DP enumeration, the total success probability of finding a short lattice vector among $\mathcal{U}$ is

$$p_{succ} \approx \min \left( 1, \sum_{\mathbf{t} \in \mathcal{U}} p_{succ}(\mathbf{t}) \right) \tag{4.17}$$

To extrapolate the probability model to higher dimensional SVP instances without performing any time-consuming computation of real lattice reduction, the concrete value of GS-sequence involved in the calculation of $p_{succ}$ should be replaced by the simulated GS sequence $\{B_1, B_2, \dots, B_n\}$ derived by equations 4.4.

Figure 9 verifies the accuracy of rectified success probability model (eq. (4.15) and (4.17)). We take the SVP instances with $n = 60, \dots, 84, \beta = 20, 30$ and $M = 50000$ as examples, and we run the DP enumeration algorithm for solving SVP challenge on each SVP instance to record the total iteration rounds. Experiment on each parameter set is repeated for 30 times to get average value The dashed line shows the expected iteration rounds $1/p_{succ}$ calculated with the original $\{\|\mathbf{b}_i^*\|\}_{i=1}^n$ of the real reduced basis, and the dotted line is calculated only with the simulated GS sequence $\{B_i\}_{i=1}^n$. The results illustrates that the rectified model gives a more precise estimation of success probability than the original method provided in [8].

## 4.4 Simulator for DP Enumeration

Based on all the works in this section, the running time of DP enumeration can be estimated by algorithm 7 below. This simulator only needs minimum information of a lattice $\mathcal{L}$.

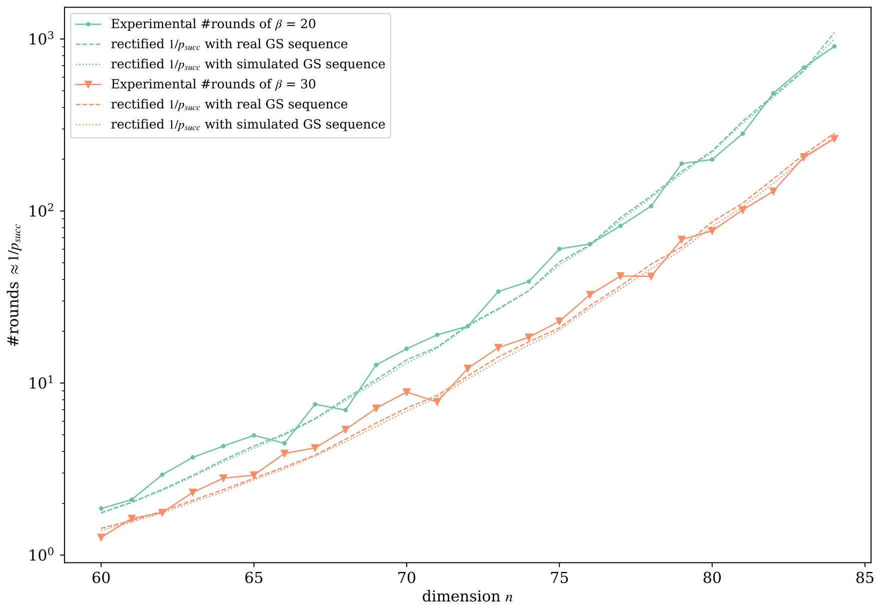

Figure 9: Verification of the Rectified success probability model

**Algorithm 7** DP-simulator

**Input:** Lattice dimension and volume $n, \mathrm{vol}(\mathcal{L})$, $k, \beta$, $M$, target length $R$ of SVP
**Output:** Expected running time (CPU cycles) of finding $\mathbf{v} \in \mathcal{L}$ such that $\|\mathbf{v}\| \le R$ by DP enumeration

```
1: Generate the simulated GS sequence $B_1, \dots, B_n$ by solving equations (4.4)
2: Calculate $r_\beta$ by equation (4.13)
3: Calculate $T_{repro}$ by calling Algorithm6 with parameters$(\beta, n, k, \mathrm{vol}(\mathcal{L}))$ as input
4: Calculate $T_{cell}$ by equation (4.5) with $M,\ n$
5: Calculate $T_{bin}$ by equation (4.6) with $M,\ n$
6: Calculate $T_{decode}$ by equation (4.7) with $n$
7: Call algorithm 3 and algorithm 2 with $B_1, \dots, B_n$ as the GS sequence, and output the $M$ tags with minimal value of $f(\mathbf{t})$
8: Calculate the total success probability $p_{succ}$ on the $M$ tags, with GS sequence $B_1, \dots, B_n$
9: return $\frac{T_{repro}+T_{bin}+T_{cell}+M \cdot T_{decode}}{p_{succ}}$
```

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

## 5 The Optimal Parameters for DP-ENUM

### 5.1 Finding the Optimal Parameter Setting by Simulator

To solve a certain SVP instance, the parameters of DP enumeration that need to be determined manually are: $\beta$ 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 $k$ very large because of the "diminishing returns" of lattice reduction, which means the improvement of basis quality would slow down with $k$ increases. We heuristically set $k = 6$ for SVP instances with $n \le 200$, which is also roughly consistent with the observation of [5] (Section 2.5). Then only $\beta$ and $M$ should be determined with restriction $0 < \beta \le n$ and $M > 0$. The two parameter should minimize the total cost of DP enumeration, i.e., the value of expression (4.1). This value is calculated by algorithm 7 and can barely be represented by a differentiable function. 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 Nelder-Mead method can converge quickly to the optimal solution.

Algorithm 8 gives the optimal parameter $\beta, M$ for a certain SVP instance based on the standard version of Nelder-Mead method.

**Algorithm 8** Finding optimal parameters for DP enumeration

**Input:** lattice dimension $n$, lattice volumn vol($\mathcal{L}$) and the target vector length $R$ of SVP
**Output:** $(\beta, M)$ that minimizes the output of DP-simulator$(n, \beta, M, R)$

```
1: $S(\beta, M) :=$ DP-simulator$(n, \beta, M, R)$ + $P(\beta, M)$
       // $P(\beta, M)$ is a penalty function to avoid that parameters exceed feasible region, i.e., $\beta > n$ or $M < 0$.
2: $N \leftarrow 2$                                        // 2 independent variables
3: Select initial points $\mathbf{x}_1 = [\beta_1, M_1], \ldots, \mathbf{x}_{N+1} = [\beta_{N+1}, M_{N+1}]$ at random
4: while true do
5:   reorder the $N+1$ points such that $S(\mathbf{x}_1) < \ldots < S(\mathbf{x}_{N+1})$
6:   $y_1 \leftarrow S(\mathbf{x}_1), \ldots, y_{N+1} \leftarrow S(\mathbf{x}_{N+1})$
7:   if $|\beta_1 - \beta_{N+1}| < 2$ and $|M_1 - M_{N+1}| < 1000$ then
8:     break;
9:   end if
10:  $\mathbf{x}_m \leftarrow \frac{1}{N}\sum_{i=0}^{N} \mathbf{x}_i$          // calculate the centroid (midpoint)
11:  $\mathbf{x}_r \leftarrow 2\mathbf{x}_m - \mathbf{x}_{N+1}$                        // reflection
12:  $y_r \leftarrow S(\mathbf{x}_r)$
13:  if $y_1 \le y_r < y_N$ then
14:    $\mathbf{x}_{N+1} \leftarrow \mathbf{x}_r$
15:    continue;
16:  else if $y_r < y_1$ then                                // expansion
17:    $\mathbf{x}_e \leftarrow \mathbf{x}_m + 2(\mathbf{x}_r - \mathbf{x}_m)$
18:    if $S(\mathbf{x}_e) < y_r$ then
19:      $\mathbf{x}_{N+1} \leftarrow \mathbf{x}_e$
20:    else
21:      $\mathbf{x}_{N+1} \leftarrow \mathbf{x}_r$
22:    end if
23:  else if $y_N \le y_r < y_{N+1}$ then                       // contraction
24:    $\mathbf{x}_c \leftarrow \mathbf{x}_m + (\mathbf{x}_r - \mathbf{x}_m)/2$
25:    if $S(\mathbf{x}_c) < y_r$ then
26:      $\mathbf{x}_{N+1} \leftarrow \mathbf{x}_c$
27:      continue;
28:    end if
29:  else
30:    $\mathbf{x}_c \leftarrow \mathbf{x}_m + (\mathbf{x}_{N+1} - \mathbf{x}_m)/2$
31:    if $S(\mathbf{x}_c) < y_r$ then
32:      $\mathbf{x}_{N+1} \leftarrow \mathbf{x}_c$
33:      continue;
34:    end if
35:  end if
36:  for $i = 2$ to $N+1$ do                                 // shrink
37:    $\mathbf{x}_i \leftarrow \mathbf{x}_1 + (\mathbf{x}_i - \mathbf{x}_1)/2$
38:  end for
39: end while
40: return The optimal parameters $\mathbf{x}_{min} \leftarrow \mathbf{x}_1 = [\beta_1, M_1]$ and the corresponding cost estimation $S(\mathbf{x}_{min})$
```

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 that $M \sim 10^5$ and $\beta < n/2$. Both of them increase not very rapidly with the growth of $n$.

Figure 10 compares the performance of extreme pruned enumeration and discrete pruned enumeration

| $n$ | $\beta$ | $M$ | Expected Time (CPU cycles) |
|---|---|---|---|
| 80 | 39 | 65000 | 9.08e+10 |
| 82 | 42 | 110000 | 1.22e+11 |
| 84 | 42 | 95000 | 1.96e+11 |
| 86 | 42 | 175000 | 2.92e+11 |
| 88 | 42 | 155000 | 4.83e+11 |
| 90 | 42 | 100000 | 8.90e+11 |
| 92 | 39 | 150000 | 1.52e+12 |
| 94 | 42 | 150000 | 2.09e+12 |
| 96 | 39 | 170000 | 6.88e+12 |
| 98 | 42 | 180000 | 1.08e+13 |
| 100 | 42 | 145000 | 2.15e+13 |
| 102 | 39 | 195000 | 4.16e+13 |
| 104 | 39 | 190000 | 1.11e+14 |
| 106 | 42 | 130000 | 1.34e+14 |
| 108 | 42 | 175000 | 4.24e+14 |
| 110 | 44 | 190000 | 1.23e+15 |

Table 2: Optimal parameters of DP ENUM for solving SVP challenge

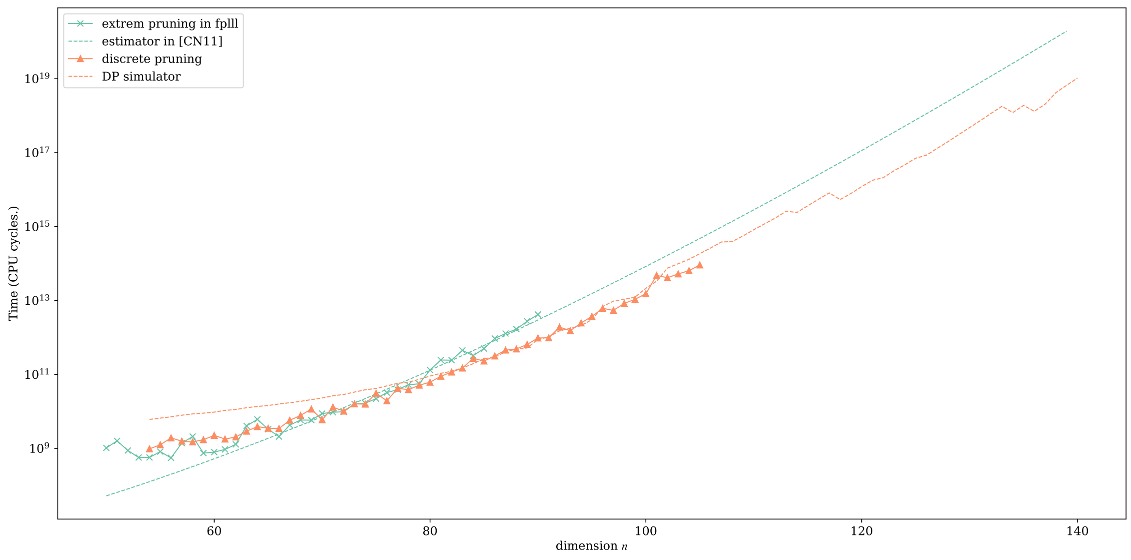

Figure 10: The performance of optimized discrete pruning vs. extreme pruning

for solving SVP challenge. For each $n$, the experiments are repeated on 30 different instances. In figure 10, the green broken line is the experimental running time of extreme pruned enumeration in fplll library [1] with the default pruning function up to $n = 90$. The green dashed line is a lower bound of extreme pruned enumeration, first proposed by Chen and Nguyen in 2011 [16], and it explicit fitting function is given by LWE estimator [6]:

$$T_{extreme} = C_{node} \times 2^{0.27019n \log(n) - 1.0192n + 16.103}$$

The orange dashed line is the running time estimation given by our simulator under the optimal parameter set given by algorithm 8, and the orange broken line is the experimental running time of our implementation of discrete pruned enumeration. For $n \lesssim 80$, DP enumeration algorithm sometimes find a solution before the first round ends, and therefor the actual running time is slightly smaller than the simulated time. And for $n > 80$, it shows that our implementation of DP enumeration (with optimal parameter set) coincides with the DP simulator and also outperformed extreme pruning method.

We also use the DP simulator with optimal parameter set to predict the running time of discrete pruning on high dimensional SVP challenge. The fitting result is [5]

$$T_{discrete} = \exp\left(0.23102n\log(n) - 1.0327n + 28.464\right)$$

Figure 11 compares the asymptotic behavior of classical enumeration with extreme pruning ($T_{extreme}$ [6]), and the fitting function of DP simulator $T_{discrete}$. Both the experimental and asymptotic comparison reveals that the discrete pruned enumeration might have more practical potential especially for solving high dimensional SVP.

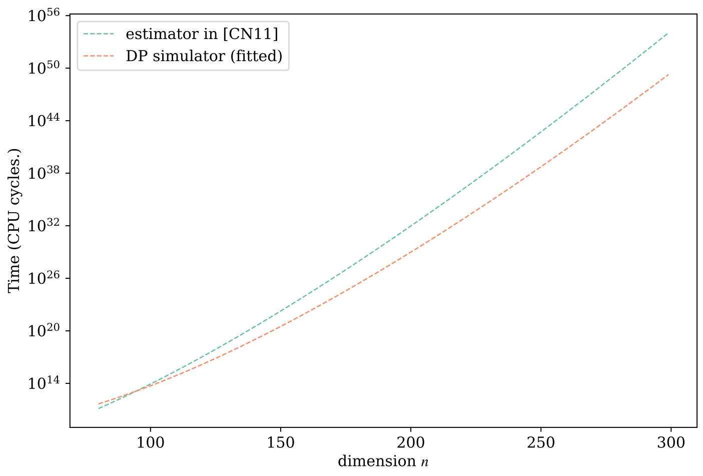

Figure 11: The asymptotic behavior of extreme pruning vs. discrete pruning

## 6 Conclusion

In this paper, we rectify the probability model of lattice point distribution in discrete pruning method to give a more rigorous description of discrete pruning. We also propose some improvements on DP enumeration algorithm to make it more practical. The most valuable part is our discrete pruning simulator combined theoretical analysis and many numerical techniques. For a certain SVP instance, we can use the DP simulator to find optimal parameters to minimize the running time of DP enumeration. The explicit time and space consumption is also given by the simulator. By simulation experiments, we believe that the time complexity of DP enumeration is still super-exponential and the space complexity is still linear, which does not change the conclusion of enumeration algorithm.

However, the experimental result shows our implementation of DP enumeration has higher efficiency than extreme pruning method implemented in fplll library, at least in medium dimensional SVP challenge ($80 < n < 110$). It also shows that the DP simulator can precisely predict the performance of DP enumeration. Then in higher dimension ($100 < n < 300$), we roughly compare the asymptotic behavior of DP enumeration and extreme pruned enumeration, which also shows the possible advantage of discrete pruning method.

There are several possible directions for improvement:

- Simulating the GS sequence more precisely. In some cases where GSA does not hold, for example, the blocksize $\beta \sim O(n)$ or the lattice is not a " random" one, some other efficient simulating techniques should be introduced. For $\beta \sim O(n)$, the BKZ simulator [13, 16] works well on random lattice, but for non-random lattice it remains to be studied and should be treated carefully.
- Using stronger reduction algorithm. As the result indicated, when $n \gtrsim 80$, DP enumeration outperforms classical enumeration with extreme pruning, which means the BKZ algorithm for preprocessing and reprocessing should call DP enumeration as SVP oracle to achieve higher efficiency, and

[5]We use the fitting function in this form to be consistent with [CN11]. Actually, the asymptotic analysis of DP enumeration is still insufficient, and we don't know whether the asymptotic cost function of DP enumeration is $\exp\left(O(n^2)\right)$ or $\exp(O(n\log n))$, and this fitting function can only be regarded as a lower bound estimation.

sieving is also an alternative SVP oracle. Besides, the structure of progressive BKZ algorithm [11] also shows strong power, although it has a very complicated running time estimator.

- Discussing the efficiency of many heuristic methods. Fukase and Kashiwabara [21] tried to improve the quality of basis by inserting short lattice vectors into basis, but it barely has theoretical proof. Since this method will influence the $p_{succ}$ in every round, the success probability model of FK algorithm should be modified.
- A parallelized implementation of DP enumeration.

## References

[1] Lattice algorithms using floating-point arithmetic(fplll). `https://github.com/fplll/fplll`.

[2] Ntl library. `http://www.shoup.net/ntl/`.

[3] Tu darmstadt, svp challenge. `https://www.latticechallenge.org/svp-challenge/`.

[4] Miklos Ajtai. 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*, STOC '03, page 396–406, New York, NY, USA, 2003. Association for Computing Machinery.

[5] Martin Albrecht. On dual lattice attacks against small-secret lwe and parameter choices in helib and seal. pages 103–129, 04 2017.

[6] Martin Albrecht, Florian Göpfert, Cedric Lefebvre, James Owen, and Rachel Player. Lwe estimator's documentation. online.

[7] Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. Post-quantum key exchange—a new hope. In *25th USENIX Security Symposium (USENIX Security 16)*, pages 327–343, Austin, TX, 8 2016. USENIX Association.

[8] Yoshinori Aono and Phong Q. Nguyen. Random sampling revisited: Lattice enumeration with discrete pruning. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology – EUROCRYPT 2017*, pages 65–102, Cham, 2017. Springer International Publishing.

[9] Yoshinori Aono, Phong Q. Nguyen, Takenobu Seito, and Junji Shikata. Lower bounds on lattice enumeration with extreme pruning. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018*, pages 608–637. Springer International Publishing, 2018.

[10] Yoshinori Aono, Phong Q. Nguyen, and Yixin Shen. Quantum lattice enumeration and tweaking discrete pruning. In Thomas Peyrin and Steven Galbraith, editors, *Advances in Cryptology – ASIACRYPT 2018*, pages 405–434. Springer International Publishing, 2018.

[11] Yoshinori Aono, Yuntao Wang, Takuya Hayashi, and Tsuyoshi Takagi. Improved progressive bkz algorithms and their precise cost estimation by sharp simulator. In *Advances in Cryptology – EUROCRYPT 2016*, pages 789–819, Berlin, Heidelberg, 05 2016. Springer Berlin Heidelberg.

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

[13] Shi Bai, Damien Stehlé, and Weiqiang Wen. Measuring, simulating and exploiting the head concavity phenomenon in bkz. In Thomas Peyrin and Steven Galbraith, editors, *Advances in Cryptology – ASIACRYPT 2018*, pages 369–404. Springer International Publishing, 2018.

[14] Johannes Buchmann and Christoph Ludwig. Practical lattice basis sampling reduction. In Florian Hess, Sebastian Pauli, and Michael Pohst, editors, *Algorithmic Number Theory*, pages 222–237, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[15] Yuanmi Chen. Réduction de réseau et sécurité concrète du chiffrement complètement homomorphe, 2013.

[16] Yuanmi Chen and Phong Q. Nguyen. Bkz 2.0: Better lattice security estimates. volume 7073, pages 1–20, 12 2011.

[17] Matthijs Coster, Antoine Joux, Brian Lamacchia, Andrew Odlyzko, Claus Schnorr, and Jacques Stern. Improved low-density subset sum algorithms. volume 2, 04 1999.

[18] Dan Ding and Guizhen Zhu. A random sampling algorithm for svp challenge based on y-sparse representations of short lattice vectors. 12 2014.

[19] Léo Ducas and Damien Stehlé. Assessing the security of lattice-based submissions: the 10 questions that nist should be asking the community. `https://www.h2020prometheus.eu/dissemination/blogs/assessing-security-lattice-based-submissions-10-questions-nist-should-be-asking`.

[20] U. Fincke and M. Pohst. Improved methods for calculating vectors of short length in a lattice. *Mathematics of Computation*, 1985.

[21] Masaharu Fukase and Kenji Kashiwabara. An accelerated algorithm for solving svp based on statistical analysis. *Journal of Information Processing*, 23(1):67–80, 2015.

[22] Masaharu Fukase and Kazunori Yamaguchi. Analysis of the extended search space for the shortest vector in lattice. *IMETI 2010 - 3rd International Multi-Conference on Engineering and Technological Innovation, Proceedings*, 2, 01 2010.

[23] Masaharu Fukase and Kazunori Yamaguchi. Finding a very short lattice vector in the extended search space. *Transactions of Information Processing Society of Japan*, 53(7):11–11, 2012.

[24] Nicolas Gama and Phong Q. Nguyen. Predicting lattice reduction. In Nigel Smart, editor, *Advances in Cryptology – EUROCRYPT 2008*, volume 4965, pages 31–51, Berlin, Heidelberg, 04 2008. Springer Berlin Heidelberg.

[25] Nicolas Gama, Phong Q. Nguyen, and Oded Regev. Lattice enumeration using extreme pruning. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, pages 257–278, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[26] Daniel Goldstein and Andrew Mayer. On the equidistribution of hecke points. *Forum Mathematicum - FORUM MATH*, 15:165–189, 01 2003.

[27] Guillaume Hanrot, Xavier Pujol, and Damien Stehlé. Terminating bkz. *IACR Cryptology ePrint Archive*, 2011:198, 01 2011.

[28] Guillaume Hanrot and Damien Stehlé. Improved analysis of kannan's shortest lattice vector algorithm. In Alfred Menezes, editor, *Advances in Cryptology - CRYPTO 2007*, pages 170–186, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[29] T. Hosono. Numerical inversion of laplace transform and some applications to wave optics. *RADIO SCIENCE*, 1981.

[30] Ravi Kannan. Improved algorithms for integer programming and related lattice problems. In *Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing*, STOC '83, page 193–206, New York, NY, USA, 1983. Association for Computing Machinery.

[31] E. Krätzel. A sum formula related to ellipsoids with applications to lattice point theory. *Abhandlungen Aus Dem Mathematischen Seminar Der Universitt Hamburg*, 71(1):143–159, 2001.

[32] Shuaigang Li, Shuqin Fan, and Xianhui Lu. Attacking ecdsa leaking discrete bits with a more efficient lattice. In *Inscrypt 2021*, 2021.

[33] Richard Lindner and Chris Peikert. Better key sizes (and attacks) for lwe-based encryption. pages 319–339, 02 2011.

[34] Mingjie Liu and Phong Q. Nguyen. Solving bdd by enumeration: An update. In Ed Dawson, editor, *Topics in Cryptology – CT-RSA 2013*, pages 293–309, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[35] Christoph Ludwig. *Practical Lattice Basis Sampling Reduction.* PhD thesis, der Technischen Universität Darmstadt, 2005.

[36] W Müller. Lattice points in large convex bodies. *Monatshefte Für Mathematik*, 128(4):315–330, 1999.

[37] Phong. Q Nguyen and Jacques Stern. Adapting density attacks to low-weight knapsacks. pages 41–58, 12 2005.

[38] Michael Pohst. On the computation of lattice vectors of minimal length, successive minima and reduced bases with applications. *SIGSAM Bull.*, 15(1):37–44, February 1981.

[39] C. P. Schnorr and M. Euchner. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. *Math. Program.*, 66(2):181–199, September 1994.

[40] C. P. Schnorr and H. H. Hörner. Attacking the chor-rivest cryptosystem by improved lattice reduction. In *Proceedings of the 14th Annual International Conference on Theory and Application of Cryptographic Techniques*, EUROCRYPT'95, page 1–12, Berlin, Heidelberg, 1995. Springer-Verlag.

[41] Claus Peter Schnorr. Lattice reduction by random sampling and birthday methods. In Helmut Alt and Michel Habib, editors, *Annual Symposium on Theoretical Aspects of Computer Science (STACS 2003)*, pages 145–156, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[42] Claus Peter Schnorr. *Factoring Integers by CVP Algorithms*, pages 73–93. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[43] Claus Peter Schnorr. Fast factoring integers by svp algorithms, corrected. Cryptology ePrint Archive, Report 2021/933, 2021. `https://ia.cr/2021/933`.

[44] Damien Stehlé. *Floating-Point LLL: Theoretical and Practical Aspects*, pages 179–213. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[45] Tadanori Teruya, Kenji Kashiwabara, and Goichiro Hanaoka. Fast lattice basis reduction suitable for massive parallelization and its application to the shortest vector problem. In Michel Abdalla and Ricardo Dahab, editors, *Public-Key Cryptography – PKC 2018*, pages 437–460, Cham, 2018. Springer International Publishing.

## A Details of Cell Enumeration

### A.1 Subalgorithm of Alg. 3

**Algorithm 9** CellENUM_probe

```
Input:  $R$, $M$, $\epsilon$
Output:  All $\mathbf{t} \in \mathbb{N}^n$ such that $\overline{f}(\mathbf{t}) \leq r$ where $\overline{f}(\mathbf{t})$ as defined in equation 3.4
1: $t_1 = t_2 = \ldots = t_n = 0$;
2: $c_1 = c_2 = \ldots = c_{n+1} = 0$;
3: $k \leftarrow 1$
4: $cnt \leftarrow 0$                                   // counter
5: while true do
6:   $c_k \leftarrow c_{k+1} + f(k, t_k)$;
7:   if $c_k < r$ then
8:     if $k = 1$ then
9:       $cnt++$
10:      if $cnt > (1+\epsilon)M$ then
11:        return 1                                   // $R$ is too large
12:      end if
13:      if $t_{k+1} = \ldots = t_n = 0$ then
14:        $t_k \leftarrow t_k + 2$;
15:      else
16:        $t_k \leftarrow t_k + 1$;
17:      end if
18:    else
19:      $k \leftarrow k - 1$;
20:      $t_k \leftarrow 0$;
21:    end if
22:  else
23:    $k \leftarrow k + 1$;
24:    if $k = n + 1$ then
25:      break;
26:    else
27:      if $t_{k+1} = \ldots = t_n = 0$ then
28:        $t_k \leftarrow t_k + 2$;
29:      else
30:        $t_k \leftarrow t_k + 1$;
31:      end if
32:    end if
33:  end if
34: end while
35: if $cnt < (1-\epsilon)M$ then
36:   return $-1$                                   // $R$ is too small
37: else
38:   return 0
39: end if
```

The time complexity of algorithm 9 is the same with algorithm 2, which is $O((2n-1)M)$.

### A.2 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 [10]. We only prove theory 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 satisfying 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

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

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 $(1-\epsilon)M_0 \leq M(\frac{R_{rk}+R_{lk}}{2}) \leq (1+\epsilon)M_0$ with $M_0$ being the input of algorithm 3. The target of our proof is to find a $\Delta R$ 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 $M(R_{lk}) < (1-\epsilon)M_0 \leq M(\frac{R_{rk}+R_{lk}}{2}) \leq (1+\epsilon)M_0 < M(R_{rk})$. Then we have

$$M(R_{rk}) - M(R_{lk}) > 2\epsilon M_0 \tag{A.1}$$

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 $A_n(x)$ [31,36]. $A_n(x)$ can be written as

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

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

$$\frac{A_n(x_{rk}) - A_n(x_{lk})}{x_{rk} - x_{lk}} = A'_n(x_\xi) \leq n {x_\xi}^{\frac{n}{2}-1} < n {x_{rk}}^{\frac{n}{2}-1} \tag{A.3}$$

Combining equation (A.1) and (A.3), we have

$$R_{rk} - R_{lk} = x_{rk} - x_{lk} = \frac{A_n(x_{rk}) - A_n(x_{lk})}{A'_n(x_\xi)} \gtrapprox \frac{2\epsilon M_0}{n {x_{rk}}^{n/2-1}} \tag{A.4}$$

Therefore the total rounds of iteration of algorithm 3 is at most

$$\begin{aligned}
&\log \frac{R_1 - R_0}{R_{rk} - R_{lk}} \\
&< \log n + \frac{n}{2} \log (R_1 + \frac{1}{4} \sum_{i=1}^{n} \|\mathbf{b}_i\|^2) - \log 2\epsilon M_0 \\
&= \log n + \frac{n}{2} \log \left( \left( \lceil M_0^{\frac{1}{n}} \rceil + \frac{1}{2} \right)^2 \mathrm{GSS}(\mathbf{B}) \right) - \log 2\epsilon M_0
\end{aligned} \tag{A.5}$$

where $\mathrm{GSS}(\mathbf{B}) = \sum_{i=1}^{n} \|\mathbf{b}_i\|^2 \leq n\det(L)^{\frac{2}{n}}$. 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. □

## B Calculating Success Probability by FILT and Euler Transformation

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

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

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

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 $f * g(z) \xlongequal{def} \int_0^z f(\tau) g(z - \tau)\, d\tau$.

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

**Step 1. Fast inverse Laplace transform**

**Theorem 2** *If 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

$$D(1) = Pr\left\{\sum_{i=1}^{n} x_i^2 \leq 1\right\} = \mathcal{L}^{-1}\left\{\frac{1}{s}\mathcal{L}\{\rho_{\sum_{i=1}^{n} x_i^2}\}(s)\right\}(t)\Big|_{t=1} \tag{B.1}$$

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

To calculate $D(1)$ in equation (B.1), we first do the Laplace transform

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

and then apply inverse Laplace transform

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

**Step 2: Approximate integral calculation with series**

Put the approximation of $\mathrm{e}^s$ in complex filed

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

into equation (B.2), [6] now notice that the integral has singularity points $s_m = a+(m-\frac{1}{2})\pi\mathrm{i},\ m=1,\dots,\infty$ [7]

According to the residue theorem and Jordan theorem, equation (B.2) can be approximated by

$$D(1) \approx \mathrm{e}^a \cdot \sum_{m=1}^{+\infty} \mathrm{Im}F\left(a+(m-\frac{1}{2})\pi\mathrm{i}\right) \tag{B.3}$$

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

Since the series in equation (B.3) 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 $F_m = \mathrm{Im}F\left(a+(m-\frac{1}{2})\pi\mathrm{i}\right)$, then the value of equation (B.3) 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 Wijingaarden transformation. In our implementation, we set $K=40, J=30$ by default.

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

[6] Here the value of $a$ should guarantee the convergence of series. For example, Hosono [29] claimed that the error is very small when $a \gg 1$, and Aono and Nugyen [8] recommended to use $a = \max(50, 30+3\sqrt{n})$.

[7] In equation (B.2), the integral path should be to the right of all singularities, i.e., $c > a$. Besides, 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.