Random Sampling Revisited: Lattice Enumeration with Discrete Pruning

Yoshinori Aono[1] and Phong Q. Nguyen[2,3(✉)]

[1] Security Fundamentals Laboratory, Cybersecurity Research Institute, National Institute of Information and Communications Technology, Tokyo, Japan
[2] Inria Paris, Paris, France
Phong.Nguyen@inria.fr
[3] CNRS/JFLI and the University of Tokyo, Tokyo, Japan

Abstract. In 2003, Schnorr introduced *Random sampling* to find very short lattice vectors, as an alternative to enumeration. An improved variant has been used in the past few years by Kashiwabara *et al.* to solve the largest Darmstadt SVP challenges. However, the behaviour of random sampling and its variants is not well-understood: all analyses so far rely on a questionable heuristic assumption, namely that the lattice vectors produced by some algorithm are uniformly distributed over certain parallelepipeds. In this paper, we introduce lattice enumeration with discrete pruning, which generalizes random sampling and its variants, and provides a novel geometric description based on partitions of the n-dimensional space. We obtain what is arguably the first sound analysis of random sampling, by showing how discrete pruning can be rigorously analyzed under the well-known Gaussian heuristic, in the same model as the Gama-Nguyen-Regev analysis of pruned enumeration from EUROCRYPT '10, albeit using different tools: we show how to efficiently compute the volume of the intersection of a ball with a box, and to efficiently approximate a large sum of many such volumes, based on statistical inference. Furthermore, we show how to select good parameters for discrete pruning by enumerating integer points in an ellipsoid. Our analysis is backed up by experiments and allows for the first time to reasonably estimate the success probability of random sampling and its variants, and to make comparisons with previous forms of pruned enumeration. Our work unifies random sampling and pruned enumeration and show that they are complementary of each other: both have different characteristics and offer different trade-offs to speed up enumeration.

1 Introduction

With the upcoming NIST standardization of post-quantum cryptography and the development of fully-homomorphic encryption, it is becoming increasingly important to provice convincing security estimates for lattice-based cryptosystems. To do so, we need to understand the best lattice algorithms, such as the ones used to solve the largest numerical challenges. For NTRU challenges [33] and

J.-S. Coron and J.B. Nielsen (Eds.): EUROCRYPT 2017, Part II, LNCS 10211, pp. 65–102, 2017.
DOI: 10.1007/978-3-319-56614-6_3

Darmstadt's lattice challenges [18], the largest records were solved (by respectively Ducas-Nguyen and Chen-Nguyen) using algorithms (pruned enumeration [11] with state-of-the-art BKZ [5]) which are reasonably well-understood. However, the seven largest records of Darmstadt's SVP challenges [28] have been solved by Kashiwabara and Teruya (using significant computational power, comparable for the largest challenge to that of RSA-768) using an algorithm which is partially secret: an incomplete description can be found in [8]. The core of the algorithm seems to be an improved variant of Schnorr's random sampling method [30], which was introduced as an alternative to enumeration [31] for finding extremely short lattice vectors.

Unfortunately, our understanding of random sampling and its variants is not satisfactory: all the analyses [4,8,9,20,30] so far rely on several heuristic assumptions, the most questionable being the *Randomness Assumption*, which says that the lattice vectors produced by some algorithm are uniformly distributed over certain parallelepipeds. As noted by Ludwig [20], this assumption cannot hold, and a gap between theoretical analyses and experimental results has been reported [8,20]. In some sense, the situation is reminiscent of enumeration with pruning, which was introduced and partially analyzed in the mid-nineties by Schnorr *et al.* [31,32], but arguably only well-understood in 2010, when Gama *et al.* [11] provided a novel geometric description and presented its first sound analysis, which provided much better parameters.

At this point, we do not know if random sampling is better or worse than pruned enumeration, neither in theory nor in practice, which is rather puzzling, considering their importance for lattice algorithms, which can be used to solve a wide range of problems, such as integer programming [15], factoring polynomials with rational coefficients [16], integer relation finding [13], as well as problems in communication theory (see [1,24] and references therein), and public-key cryptanalysis (see [22] and references therein). Pruned enumeration is used in state-of-the-art implementations of BKZ [2,5].

Our results. We introduce lattice enumeration with discrete pruning, which generalizes naturally Schnorr's random sampling and all its variants, and provides a novel geometric description based on partitions of the n-dimensional space. This new description allows us to rigorously analyze discrete pruning under the well-known Gaussian heuristic, in the same model as the Gama-Nguyen-Regev [11] analysis of pruned enumeration, albeit using different tools. This is the first sound analysis of random sampling and its variants, and our presentation unifies both pruned enumeration and random sampling, by viewing them as two different ways of speeding up the classical enumeration algorithm. In other words, we improve the understanding of random sampling to that of pruned enumeration.

To complement our theoretical analysis, we introduce three technical tools which allow, in practice, to estimate success probabilities and optimize parameters for discrete pruning: this is the most difficult aspect of discrete pruning, because given parameters, estimating the running time of discrete pruning is on the other hand very easy. The first two tools are combined to estimate accurately and efficiently the success probability: the first one computes efficiently

the volume of the intersection of an n-dimensional ball with a box, and the second one uses statistical inference to approximate efficiently large sums of such volumes without computing individually each volume. Finally, the third tool is an efficient algorithm to generate nearly-optimal parameters for discrete pruning in practice.

Our analysis is backed up by experiments, and allows us to make concrete comparisons with other forms of pruned enumeration. As an example, our analysis shows that the Fukase-Kashiwabara variant [8] outperforms Schnorr's original algorithm and its variants by Buchmann-Ludwig [4,20]. Experimentally, we find that discrete pruning is complementary with continuous pruning: whether one is more efficient than the other depends on the exact setting, such as what is the lattice dimension, the radius of the enumeration ball, the required time, *etc.*

Technical overview. A *lattice* is the set of all integer combinations of n linearly independent vectors $\mathbf{b}_1, \ldots, \mathbf{b}_n$ in $\mathbb{R}^n$. These vectors are known as a *basis* of the lattice. The most famous computational problem involving lattices is the *shortest vector problem* (SVP), which asks to find a nonzero lattice vector of smallest norm, given a lattice basis as input. A basic approach is *enumeration* which dates back to the early 1980s with work by Pohst [25], Kannan [15], and Fincke-Pohst [7] and is still actively investigated (*e.g.*, [1,11,12,31,35]): given a radius $R > 0$ and a basis $B = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ of a lattice L, enumeration computes all the points in $L \cap S$ where S is the zero-centered ball of radius R, which allows to find a shortest lattice vector by comparing their norms. Enumeration goes through all lattice vectors $\sum_{i=1}^{n} x_i \mathbf{b}_i$, exhaustively searching in order $x_n, x_{n-1} \ldots x_1 \in \mathbb{Z}$ by projecting L onto suitable subspaces of increasing dimension: for instance, bounds on x_n are found by projecting L onto a line, and intersecting it with S. The running time of enumeration depends on R and the quality of the basis, but is typically super-exponential in n.

To speed up the running time of enumeration, Schnorr, Euchner, and Hörner [31,32] suggested in the 1990s a modification of enumeration called *pruned enumeration*. We follow the geometric presentation of Gama, Nguyen and Regev [11], who revisited pruned enumeration. Essentially, pruned enumeration is a trade-off, defined by a pruning set $P \subseteq \mathbb{R}^n$: P is chosen in such a way that enumerating all the points in $L \cap S \cap P$ is much faster than over $L \cap S$, but this is only useful if $L \cap S \cap P$ is non-trivial $\not\subseteq \{0\}$, in which case we have found a short non-zero lattice vector in $L \cap S$. Under the Gaussian heuristic, $L \cap S \cap P$ is "expected" to be non-trivial when $\mathrm{vol}(S \cap P)$ is sufficiently large with respect to the lattice. Here, the set P is defined as an intersection of n cylinders such that $P \subseteq S$, thus $\mathrm{vol}(S \cap P) = \mathrm{vol}(P)$, and [11] shows how to approximate efficiently $\mathrm{vol}(P)$, which allows to choose good parameters: furthermore, if $\mathrm{vol}(S \cap P)$ turns out to be too small to hope for a solution, one can simply repeat the process with many choices of P, which may still be cheaper overall than the original enumeration.

We introduce *discrete pruning*, which follows the same framework, except that the pruning set P is completely different. Instead of a cylinder-intersection, we consider a set P formed by regrouping finitely many cells of suitable partitions

of $\mathbb{R}^n$, related to the lattice L: we show that random sampling and its variants all fall in this category. In practice, P can be rewritten as the union of finitely many non-overlapping boxes, where a box means a cube whose sides are not necessarily of equal length but are still perpendicular. To analyze the algorithm, it therefore suffices to be able to compute $\mathrm{vol}(S \cap H)$ where S is a ball and H is a box: we give exact formulas as infinite series, asymptotical estimations and efficient numerical methods to compute such volumes, which might be of independent interest. However, it actually suffices to approximate a large sum $\sum_i \mathrm{vol}(S \cap H_i)$: we introduce the use of statistical inference to approximate such a sum, without computing each term of the sum.

To select good parameters for the algorithm, we introduce a fast method based on enumerating integer points in an ellipsoid, which allows to select a nearly-optimal pruning set P without resorting to computations of $\mathrm{vol}(S \cap H)$.

Experimental results. Our experiments support our analysis of discrete pruning and explain very clearly why the Fukase-Kashiwabara variant [8] of random sampling outperforms Schnorr's random sampling [30] and its Buchmann-Ludwig variant [4]: for typical parameters, the cells selected by [8] turn out to have a much larger intersection with the ball S than the corresponding cells in [4,30]. Our algorithm for selecting discrete pruning parameters provides parameters which are in practice at least slightly better than [8]: we stress that our method is completely automatic, whereas [8] strongly relies on experiments and do not explain how to select parameters in arbitrary dimension, which makes comparisons difficult. Our experiments suggest that discrete pruning can be slower or faster than continuous pruning [11,31,32], depending on the exact setting. First, the performances are impacted by the lattice dimension, the enumeration radius, and the target running time. For instance, we find that the benefits of discrete pruning increase when the lattice dimension grows, the target running time is small and the basis is not too reduced. Potentially, this might improve BKZ-type algorithms, by speeding up the preprocessing of the enumeration subroutine, or by incorporating high-dimensional discrete pruning in the reduction process itself. Second, our analysis shows that the optimal basis reduction required by discrete pruning is different from the optimal basis reduction required by continuous pruning: roughly speaking, the smaller the sum of squared Gram-Schmidt norms, the better for discrete pruning. This means that to fully take advantage of discrete pruning, one may have to modify the reduction algorithm, and not simply use LLL or BKZ: in fact, some of the secret modifications of [8] may exactly target that, as it is clear that the reduction of [8] is a bit different from BKZ strategies. Third, there are implementation differences between discrete pruning and continuous pruning. It appears that discrete pruning is easier to parallelize, which may make it better suited to special hardware. Finding good parameters for discrete pruning is also a bit easier than for continuous pruning: in particular, there are less parameters to consider, which might be useful for blockwise reduction.

Future work. There are many interesting questions related to discrete pruning. First, one may wonder what is the most efficient form of pruned enumeration, either asymptotically or in practice: we now know continuous pruning [11,31,32] and discrete pruning, and we showed how to compare both in practice, but a theoretical asymptotical comparison is not easy. Can a combination of both, or another form of pruning be more efficient? Second, is it possible to efficiently reduce a basis in such a way that the power of discrete pruning is maximized? For instance, are there better ways to decrease the sum of squared Gram-Schmidt norms?

We presented discrete pruning in the SVP case, *i.e.* to find short lattice vectors. The methodology can be adapted to the CVP case, *i.e.* to find lattice vectors close to a given target: as a special case, [19] already noticed that the Lindner-Peikert algorithm [17] can be viewed as the BDD adaptation of Schnorr's random sampling [30]. However, in the general case of discrete pruning, there appears to be a few differences: the details will be investigated in future work.

Finally, our algorithm to select good discrete pruning parameters is based on enumerating integer points in an ellipsoid: though the algorithm is extremely efficient in practice, we do not know at the moment how to prove it, and it would be interesting to do so.

Related work. Liu and Nguyen [19] also tried to view Schnorr's random sampling as some form of pruning, in the context of bounded distance decoding: however, their formalization does not rely on partitions, and does not capture all the variants of random sampling, including the one of [8].

Roadmap. We start in Sect. 2 with some background and notation on lattices, and continue with a general description of enumeration (with or without pruning) in Sect. 3. In Sect. 4, we present lattice enumeration with discrete pruning based on partitions, show that it generalizes Schnorr's random sampling and its variants, and give our rigorous analysis. In Sect. 5, we address the technical problem of computing the volume of the intersection of a ball with a box, which is required by our discrete pruning analysis. In Sect. 6, we show how to select optimal parameters for discrete pruning. We present experimental results in Sect. 7. In the full version available on the eprint archive, we provide missing proofs and additional information.

2 Preliminaries

General. For any finite set U, we denote by $\#U$ its number of elements. For any measurable subset $S \subseteq \mathbb{R}^n$, we denote by $\mathrm{vol}(S)$ its volume. Throughout the paper, we use row representations of matrices. The Euclidean norm of a vector $\mathbf{v} \in \mathbb{R}^n$ is denoted $\|\mathbf{v}\|$. We denote by $\mathrm{Ball}_n(R)$ the n-dimensional zero-centered Euclidean ball of radius R, whose volume is $\mathrm{vol}(\mathrm{Ball}_n(R)) = R^n \frac{\pi^{n/2}}{\Gamma(n/2+1)}$.

Lattices. A *lattice* L is a discrete subgroup of $\mathbb{R}^m$. Alternatively, we can define a lattice as the set $L(\mathbf{b}_1, \ldots, \mathbf{b}_n) = \{\sum_{i=1}^{n} x_i \mathbf{b}_i \ : \ x_i \in \mathbb{Z}\}$ of all integer combinations of n linearly independent vectors $\mathbf{b}_1, \ldots, \mathbf{b}_n \in \mathbb{R}^m$. This sequence of vectors is known as a *basis* of the lattice L. All the bases of L have the same number n of elements, called the dimension or rank of L, and the n-dimensional volume of the parallelepiped $\{\sum_{i=1}^{n} a_i \mathbf{b}_i \ : \ a_i \in [0,1)\}$ they generate. We call this volume the co-volume, or determinant, of L, and denote it by $\mathrm{covol}(L)$. The lattice L is said to be *full-rank* if $n = m$. We denote by $\lambda_1(L)$ the first minimum of L, defined as the length of a shortest nonzero vector of L. The most famous lattice problem is the *shortest vector problem* (SVP), which asks to find a lattice vector of norm $\lambda_1(L)$.

Orthogonalization. For a basis $B = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ of a lattice L and $i \in \{1, \ldots, n\}$, we denote by π_i the orthogonal projection on $\mathrm{span}(\mathbf{b}_1, \ldots, \mathbf{b}_{i-1})^{\perp}$. The *Gram-Schmidt orthogonalization* of the basis B is defined as the orthogonal sequence of vectors $B^{\star} = (\mathbf{b}_1^{\star}, \ldots, \mathbf{b}_n^{\star})$, where $\mathbf{b}_i^{\star} := \pi_i(\mathbf{b}_i)$. For each i we can write $\mathbf{b}_i$ as $\mathbf{b}_i^{\star} + \sum_{j=1}^{i-1} \mu_{i,j} \mathbf{b}_j^{\star}$ for some unique $\mu_{i,1}, \ldots, \mu_{i,i-1} \in \mathbb{R}$. Thus, we may represent the $\mu_{i,j}$'s by a lower-triangular matrix μ with unit diagonal. The projection of a lattice may not be a lattice, but for all $i \in \{1, \ldots, n\}$, $\pi_i(L)$ is an $n + 1 - i$ dimensional lattice generated by the basis $\pi_i(\mathbf{b}_i), \ldots, \pi_i(\mathbf{b}_n)$, with $\mathrm{covol}(\pi_i(L)) = \prod_{j=i}^{n} \left\|\mathbf{b}_j^{\star}\right\|$.

Reduced bases. Lattice reduction algorithms aim to transform an input basis into a "high quality" basis. There are many ways to quantify the quality of bases produced by lattice reduction algorithms. One popular way is to consider the Gram-Schmidt norms $\|\mathbf{b}_1^{\star}\|, \ldots, \|\mathbf{b}_n^{\star}\|$. Intuitively speaking, a good basis is one in which this sequence does not decay too fast. In practice, it turns out that the Gram-Schmidt coefficients of bases produced by the main reduction algorithms (such as LLL or BKZ) have a certain "typical shape", assuming the input basis is sufficiently random. This property was thoroughly investigated in [10,23]. This typical shape is often used to estimate the running time of various algorithms. In particular, many theoretical asymptotic analyses (as introduced by Schnorr [30]) assume for simplicity that this shape is given by $\|\mathbf{b}_i^{\star}\|/\|\mathbf{b}_{i+1}^{\star}\| = q$ where q depends on the reduction algorithm; although less precise, this approximation called the *geometric series assumption (GSA)* is very close to the shape observed in practice.

Gaussian Heuristic. The classical Gaussian Heuristic provides an estimate on the number of lattice points inside a "nice enough" set:

Heuristic 1. *Given a full-rank lattice $L \subseteq \mathbb{R}^n$ and a measurable set $S \subseteq \mathbb{R}^n$, the number of points in $S \cap L$ is approximately* $\mathrm{vol}(S)/\mathrm{covol}(L)$.

If the heuristic holds, we would expect $\lambda_1(L)$ to be close to $\mathrm{GH}(L) = \mathrm{vol}(\mathrm{Ball}_n(1))^{-1/n}\mathrm{covol}(L)^{1/n}$, and that there about α^n points in L which have norm $\leq \alpha \mathrm{GH}(L)$. Some rigorous results along these lines are known. For instance,

for a fixed set S, if we consider random lattices of unit-covolume, and scale them to have $\mathrm{covol}(L)$, Siegel [34] shows that the average number of points in $S \cap L$ is exactly $\mathrm{vol}(S)/\mathrm{covol}(L)$. Furthermore, it is known that $\lambda_1(L)$ is in some sense close to $\mathrm{GH}(L)$ for a random lattice (see [26]). Note, however, that the heuristic can also be far off; for $L = \mathbb{Z}^n$, for instance, the heuristic estimates the number of lattice points inside a ball of radius $\sqrt{n}/10$ around the origin to be less than 1, yet there are exponentially many lattice points there (see [21] for more such examples). One should therefore experimentally verify the use of the heuristic, as we shall do later. This is particulary necessary, as pruned enumeration relies on strong versions of the heuristic, where the set S is not fixed, but actually depends on a basis of L.

Statistics. We denote by $\mathbb{E}()$ and $\mathbb{V}()$ respectively the expectation and the variance of a random variable.

Lemma 1. *Let X be a random variable uniformly distributed over $[\alpha, \beta]$. Then:*

$$\mathbb{E}(X^2) = \frac{\alpha^2 + \beta^2 + \alpha\beta}{3} \quad \textit{and} \quad \mathbb{V}(X^2) = \frac{4}{45}\alpha^4 - \frac{1}{45}\alpha^3\beta - \frac{2}{15}\alpha^2\beta^2 - \frac{1}{45}\alpha\beta^3 + \frac{4}{45}\beta^4$$

Proof. We have:

$$\mathbb{E}(X^2) = \frac{1}{\beta - \alpha}\int_\alpha^\beta x^2 dx = \frac{1}{\beta - \alpha}\left[x^3/3\right]_\alpha^\beta = \frac{\alpha^2 + \beta^2 + \alpha\beta}{3}$$

$$\mathbb{E}(X^4) = \frac{1}{\beta - \alpha}\int_\alpha^\beta x^4 dx = \frac{1}{\beta - \alpha}\left[x^5/5\right]_\alpha^\beta = \frac{\alpha^4 + \alpha^3\beta + \alpha^2\beta^2 + \alpha\beta^3 + \beta^4}{5}$$

Finally, $\mathbb{V}(X^2) = \mathbb{E}(X^4) - \mathbb{E}(X^2)^2$. □

Corollary 1. *Let $y \in \mathbb{R}$. Let X (resp. X') be a random variable uniformly distributed over $[y - 1/2, y + 1/2]$ (resp. $[y/2, (y+1)/2]$). Then:*

$$\mathbb{E}(X^2) = y^2 + \frac{1}{12}, \mathbb{E}(X'^2) = y^2/4 + y/4 + \frac{1}{12}, \quad \mathbb{V}(X^2) = \frac{y^2}{3} + \frac{1}{180}, \mathbb{V}(X'^2) = \frac{y^2}{48} + \frac{y}{48} + \frac{1}{180}.$$

In this paper, it is convenient to extend the expectation and variance this to any measurable set C of $\mathbb{R}^n$ by using the squared norm, to measure how short is a random vector of C:

$$\mathbb{E}\{C\} := \mathbb{E}_{\mathbf{x}\in C}(\|\mathbf{x}\|^2) \qquad \mathbb{V}\{C\} := \mathbb{V}_{\mathbf{x}\in C}(\|\mathbf{x}\|^2).$$

Normal distribution. The CDF of the normal distribution of expectation 0 and variance 1 is the error function

$$\mathrm{erf}(z) := \frac{2}{\sqrt{\pi}}\int_0^z e^{-t^2} dt.$$

3 Enumeration with Pruning

In this section, we give an overview of lattice enumeration and pruning, and revisit the analysis model of [11].

3.1 Enumeration

Let L be a full-rank lattice in $\mathbb{R}^n$. Enumeration [7,15,25] is an elementary algorithm which, given a basis $B = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ of L and a radius $R > 0$, outputs all the points in $L \cap S$ where $S = \mathrm{Ball}_n(R)$: by comparing all their norms, it is then possible to extract the shortest lattice vectors.

The main idea is to perform a recursive search using projections, which allows to reduce the dimension of the lattice: if $\|\mathbf{v}\| \leq R$, then $\|\pi_k(\mathbf{v})\| \leq R$ for all $1 \leq k \leq n$. We start with the one-dimensional lattice $\pi_n(L)$: it is trivial to enumerate all the points in $\pi_n(L) \cap S$. Assume that we enumerated all the points in $\pi_{k+1}(L) \cap S$ for some $k \geq 1$, then we can derive all the points in $\pi_k(L) \cap S$ by enumerating the intersection of a one-dimensional lattice with a suitable ball, for each point in $\pi_{k+1}(L) \cap S$. Concretely, it can be viewed as a depth first search of a gigantic tree called the enumeration tree. The running-time of enumeration depends on R and the quality of B, but it is typically super-exponential in n, even if $L \cap S$ is small.

We do not need to know more about enumeration: the interested reader is referred to [11] for more details.

3.2 Enumeration with Pruning

We note that in high dimension, enumeration is likely to be unfeasible in general for any radius $R \gg \mathrm{GH}(L)$: indeed, by the Gaussian heuristic, we expect $\#(L \cap \mathrm{Ball}_n(R))$ to have about $(R/\mathrm{GH}(L))^n$ points. For such large radius R, it is therefore more meaningful to just ask for one solution (or say, a bounded number of solutions) in $L \cap \mathrm{Ball}_n(R)$, rather than all the points.

Enumeration with pruning is a natural method to speed up enumeration, which goes back to the work of Schnorr *et al.* [31,32] in the 90s. We introduce its more general form: pruned enumeration uses an additional parameter, namely a pruning set $P \subseteq \mathbb{R}^n$, and outputs all points in $L \cap S \cap P$. The advantage is that for suitable choices of P, enumerating $L \cap S \cap P$ is much cheaper than enumerating $L \cap S$.

If $L \cap S \cap P \not\subseteq \{0\}$, then pruned enumeration provides non-trivial points in $L \cap S$, which is the first goal of pruned enumeration. Otherwise, it will return nothing or the zero vector, but we can simply repeat the process with many different P's until we find a non-trivial point, provided that there are many choices for P. In fact, by repeating sufficiently many times this process, one might even be able to recover all of $L \cap S$.

In order to analyze the algorithm, we need to predict when $L \cap S \cap P \subseteq \{0\}$: this is especially tricky, since for all choices of P considered in the past, the enumeration of $L \cap S \cap P$ was completely deterministic, which makes the probability space unclear. Gama *et al.*[11] provided the first sound analysis of enumeration with pruning, by viewing the pruning set P as a random variable: in practice, it depends on the choice of basis B. Then we define the success probability of pruned enumeration as:

$$\Pr_{\mathrm{succ}} = \Pr_{P}(L \cap S \cap P \not\subseteq \{0\}),$$

that is, the probability that it outputs a non-trivial point in $L \cap S$.

In general, this probability is very hard to compute. To estimate this probability, in the spirit of [11], we make the following heuristic assumption inspired by the Gaussian heuristic applied to the lattice L and the set $S \cap P$:

Heuristic 2. *For reasonable pruning sets P and lattices L, the success probability* $\Pr(L \cap S \cap P \not\subseteq \{0\})$ *is close to* $\min(1, \mathrm{vol}(S \cap P)/\mathrm{covol}(L))$*, and when this is close to 1, the cardinal of $L \cap S \cap P$ is close to* $\mathrm{vol}(S \cap P)/\mathrm{covol}(L)$.

We stress that this well-defined heuristic is only a heuristic: it is easy to select pruning sets P for which the heuristic cannot hold. But the experiments of [11] show that the heuristic typically holds for the pruning sets considered by [11], and our experiments show that it also holds for typical pruning sets corresponding to discrete pruning.

If the heuristic holds, then it suffices to be able to compute $\mathrm{vol}(S \cap P)$ to estimate the success probability of pruned enumeration. To estimate the running time of the full algorithm, we need more information:

- An estimate of the cost of enumerating $L \cap S \cap P$.
- An estimate of the cost of computing the (random) reduced basis B.

Gama *et al.* [11] introduced extreme pruning in which $\mathrm{vol}(S \cap P)/\mathrm{covol}(L)$ converges to zero, yet the global running time to find a non-zero short vector is much faster than enumeration, namely exponentially faster asymptotically for the choice of [11].

3.3 Continuous Pruning

Until now, the most general form of pruning set P that has been used is the following generalization [11] of pruned enumeration of [31,32], which was also concurrently used in [35]. There, P is defined by a function $f : \{1, \ldots, n\} \to [0, 1]$ and a lattice basis $B = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ as follows:

$$P = \{\mathbf{x} \in \mathbb{R}^n \text{ s.t. } \|\pi_{n+1-i}(\mathbf{x})\| \leq f(i)R \text{ for all } 1 \leq i \leq n\},$$

where the π_i's are the Gram-Schmidt projections defined by the basis B. We call *continuous pruning* this form of pruned enumeration, by opposition with *discrete pruning*, which is the topic of this paper.

By definition, $P \subseteq S$ so $\mathrm{vol}(S \cap P) = \mathrm{vol}(P)$. By suitable rotation, isometry and scaling, $\mathrm{vol}(P)$ can be derived from R, n and the volume of the cylinder intersection defined by f:

$$C_f = \left\{ (x_1, \ldots, x_n) \in \mathbb{R}^n \text{ s.t. } \sum_{j=1}^{i} x_j^2 \leq f(i)^2 \text{ for all } 1 \leq i \leq n \right\}.$$

Gama *et al.* [11] showed how to efficiently compute tight lower and upper bounds for $\mathrm{vol}(C_f)$, thanks to the Dirichlet distribution and special integrals. Using Heuristic 2, this allows to reasonably estimate the probability of success.

Using the shape of P, [11] also estimated of the cost of enumerating $L \cap S \cap P$, by using the Gaussian heuristic on projected lattices $\pi_i(L)$, as suggested in [12]: these estimates are usually accurate in practice.

To optimize the whole selection of parameters, one finally needs to take into account the cost of computing the (random) reduced basis of B. For instance, this is done in [2,5], which illustrates the power of continuous pruning in the context of lattice reduction.

Though continuous pruning has proved very successful, it is unknown if it is the most efficient form of pruned enumeration: there might be better choices of pruning sets P, and this is the starting point of discrete pruning, which we introduce next.

4 Enumeration with Discrete Pruning

We now introduce enumeration with discrete pruning, which generalizes Schnorr's random sampling [30] and its variants [4,8], and a provides a novel geometric description based on partitions, which is crucial for our analysis.

4.1 Lattice Partitions

Discrete pruning is based on what we call a *lattice partition*:

Definition 1. *Let L be a full-rank lattice in $\mathbb{Q}^n$. An L-partition is a partition $\mathcal{C}$ of $\mathbb{R}^n$ such that:*

- *The partition is countable: $\mathbb{R}^n = \cup_{t \in T} \mathcal{C}(t)$ where T is a countable set, and $\mathcal{C}(t) \cap \mathcal{C}(t') = \varnothing$ whenever $t \neq t'$.*
- *Each cell $\mathcal{C}(t)$ contains a single lattice point, which can be found efficiently: given any $t \in T$, one can compute in polynomial time the single point of $\mathcal{C}(t) \cap L$. We call this process the cell enumeration.*

We call $t \in T$ the tag of the cell $\mathcal{C}(t)$. Since $\mathbb{R}^n = \cup_{t \in T} \mathcal{C}(t)$, this means that any point in $\mathbb{R}^n$ also has a tag, because it belongs to a unique cell $\mathcal{C}(t)$. Any lattice partition induces a (bijective) encoding of lattice points onto T: any lattice point belongs to $\mathbb{R}^n$, and therefore has a tag; reciprocally, given a tag $t \in T$, one can compute the unique lattice point in $\mathcal{C}(t)$ by the cell enumeration.

The simplest examples of lattice partitions come from fundamental domains of lattices. In particular, one can easily check that any basis $B = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ of L gives rise to two trivial lattice partitions with $T = \mathbb{Z}^n$ as follows:

- $\mathcal{C}(\mathbf{t}) = \mathbf{t}B + \mathcal{D}$ where $\mathcal{D} = \{\sum_{i=1}^n x_i \mathbf{b}_i \text{ s.t. } -1/2 \leq x_i < 1/2\}$ is a parallelepiped.
- $\mathcal{C}(\mathbf{t}) = \mathbf{t}B + \mathcal{D}$ where $\mathcal{D} = \{\sum_{i=1}^n x_i \mathbf{b}_i^\star \text{ s.t. } -1/2 \leq x_i < 1/2\}$ is a box, *i.e.* a paralellepiped whose axes are pairwise orthogonal.

However, these lattice partitions are not very useful, because $\mathcal{C}(\mathbf{t}) \cap L = \{\mathbf{t}B\}$, which means that we know the lattice point directly from its tag: the cell enumeration is just a matrix/vector product.

The first non-trivial example of lattice partition is the following:

Lemma 2. *Let B be a basis of a full-rank lattice L in $\mathbb{Z}^n$. Let $T = \mathbb{Z}^n$ and for any $\mathbf{t} \in T$, $\mathcal{C}_{\mathbb{Z}}(\mathbf{t}) = \mathbf{t}B^\star + \mathcal{D}$ where $\mathcal{D} = \{\sum_{i=1}^n x_i \mathbf{b}_i^\star \text{ s.t. } -1/2 \leq x_i < 1/2\}$. Then $(\mathcal{C}_{\mathbb{Z}}(), T)$ with Algorithm 1 is an L-partition, which we call Babai's partition.*

Proof. We already know that $(\mathcal{C}_{\mathbb{Z}}(), T)$ is a $L(B^\star)$-partition because $B^\star$ is a basis of $L(B^\star)$. To show that it is also a L-partition, it suffices to show that $\mathcal{C}_{\mathbb{Z}}(t) \cap L$ is always a singleton, which can be found in polynomial time. To see this, note that Babai's nearest plane algorithm [3] implies that for any $\mathbf{t} \in T$, there is a unique $\mathbf{v} \in L$ such that $\mathbf{v} - \mathbf{t}B^\star \in \mathcal{D}$, and that $\mathbf{v}$ can be found in polynomial time. It follows that $\mathcal{C}_{\mathbb{Z}}(t) \cap L = \{\mathbf{v}\}$. □

The encoding of lattice points induced by Babai's partition is exactly the encoding onto $\mathbb{Z}^n$ introduced in [6]. The paper [8] defines a different encoding of lattice points onto $\mathbb{N}^n$, which implicitly uses a different lattice partition based on $T = \mathbb{N}^n$ rather than $\mathbb{Z}^n$, which we now define:

Algorithm 1. Cell enumeration for Babai's partition from Babai's Nearest Plane algorithm [3]

Input: A tag $\mathbf{t} \in \mathbb{Z}^n$ and a basis $B = (\mathbf{b}_1, \ldots, \mathbf{b}_n) \in \mathbb{Q}^n$ of a lattice L, with Gram-Schmidt orthogonalization $B^\star$.
Output: $\mathbf{v} \in L$ such that $\{\mathbf{v}\} = L \cap \mathcal{C}_{\mathbb{Z}}(\mathbf{t})$

$\mathbf{v} \leftarrow \mathbf{0}$ and $\mathbf{u} \leftarrow \mathbf{t}B^\star$
for $i := n$ downto 1 **do**
  Compute the integer c closest to $\langle \mathbf{b}_i^\star, \mathbf{u} \rangle / \langle \mathbf{b}_i^\star, \mathbf{b}_i^\star \rangle$
  $\mathbf{u} \leftarrow \mathbf{u} - c\mathbf{b}_i$ and $\mathbf{v} \leftarrow \mathbf{v} + c\mathbf{b_i}$
end for
Return $\mathbf{v}$

Algorithm 2. Tagging for the natural partition

Input: A vector $\mathbf{x} = \sum_{i=1}^n x_i \mathbf{b}_i^\star \in \mathbb{R}^n$ where the $\mathbf{b}_i^\star$'s are the Gram-Schmidt vectors of a basis $B = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ of a full-rank lattice L in $\mathbb{R}^n$.
Output: A tag $\mathbf{t} \in \mathbb{N}^n$ such that $\mathbf{x} \in \mathcal{C}_{\mathbb{N}}(\mathbf{t})$.

for $i := n$ downto 1 **do**
  if $x_i > 0$ **then**
    $t_i \leftarrow \lceil 2x_i \rceil - 1$
  else
    $t_i \leftarrow -\lfloor 2x_i \rfloor$
  end if
end for
Return $(t_1, \ldots, t_n)$

Algorithm 3. Cell enumeration for the natural partition

Input: A tag $\mathbf{t} \in \mathbb{N}^n$ and a basis $B = (\mathbf{b}_1, \ldots, \mathbf{b}_n) \in \mathbb{Q}^n$ of a lattice L, with Gram-Schmidt orthogonalization matrix μ.
Output: $\mathbf{v} \in L$ such that $\{\mathbf{v}\} = L \cap \mathcal{C}_{\mathbb{N}}(\mathbf{t})$

for $i := n$ downto 1 **do**
  $y \leftarrow -\sum_{j=i+1}^{n} u_j \mu_{j,i}$
  $u_i \leftarrow \lfloor y + 0.5 \rfloor$
  if $u_i < y$ **then**
    $u_i \leftarrow u_i - (-1)^{t_i} \lceil t_i/2 \rceil$
  else
    $u_i \leftarrow u_i + (-1)^{t_i} \lceil t_i/2 \rceil$
  end if
end for
Return $\sum_{i=1}^{n} u_i \mathbf{b}_i$

Lemma 3. *Let B be a basis of a full-rank lattice L in $\mathbb{Z}^n$. Let $T = \mathbb{N}^n$ and for any $\mathbf{t} = (t_1, \ldots, t_n) \in T$, $\mathcal{C}_{\mathbb{N}}(\mathbf{t}) = \{\sum_{i=1}^{n} x_i \mathbf{b}_i^\star$ s.t. $-(t_i+1)/2 < x_i \leq -t_i/2$ or $t_i/2 < x_i \leq (t_i+1)/2\}$. Then $(\mathcal{C}_{\mathbb{N}}(), T)$ with Algorithm 3 is an L-partition, which we call the natural partition.*

Proof. The fact that the cells $\mathcal{C}_{\mathbb{N}}(\mathbf{t})$ form a partition of $\mathbb{R}^n$ is obvious: the cells are clearly disjoint and any point of $\mathbb{R}^n$ belongs to a cell (see Algorithm 2). The only difficulty is to show that any cell contains one and only one lattice point, and that it can be found efficiently: this is achieved by Algorithm 3, which is a variant of Babai's nearest plane algorithm. □

Figure 1 displays Babai's partition and the natural partition (with tags) in dimension two. The encoding of lattice points derived from the natural partition is exactly the encoding introduced by Fukase and Kashiwabara in [8]. What is remarkable is that every cell of the natural partition is not connected, except the zero cell: each cell $\mathcal{C}_{\mathbb{N}}(\mathbf{t})$ is the union of 2^k boxes, where k is the number of non-zero coefficients in $\mathbf{t}$, as illustrated by Fig. 1.

To compare the Babai partition and the natural partition, we study the moments of their cells, which follow from Corollary 1:

Fig. 1. Babai's partition and the natural partition in dimension two: different cells are coloured differently. (Color figure online)

Corollary 2 (Moments of Babai's partition). *Let B be a basis of a full-rank lattice L in $\mathbb{R}^n$. Let $\mathbf{t} = (t_1, \ldots, t_n) \in \mathbb{Z}^n$ and the cell $\mathcal{C}_{\mathbb{Z}}(\mathbf{t}) = \mathbf{t}B^\star + \mathcal{D}$ where $\mathcal{D} = \{\sum_{i=1}^n x_i \mathbf{b}_i^\star \text{ s.t. } -1/2 \leq x_i < 1/2\}$. Then:*

$$\mathbb{E}\{\mathcal{C}_{\mathbb{Z}}(\mathbf{t})\} = \sum_{i=1}^{n} \left(t_i^2 + \frac{1}{12}\right) \|\mathbf{b}_i^\star\|^2 \quad \text{and} \quad \mathbb{V}\{\mathcal{C}_{\mathbb{Z}}(\mathbf{t})\} = \sum_{i=1}^{n} \left(\frac{t_i^2}{3} + \frac{1}{180}\right) \|\mathbf{b}_i^\star\|^4$$

Corollary 3 (Moments of the natural partition). *Let B be a basis of a full-rank lattice L in $\mathbb{R}^n$. Let $\mathbf{t} = (t_1, \ldots, t_n) \in \mathbb{N}^n$ and the cell $\mathcal{C}_{\mathbb{N}}(\mathbf{t}) = \{\sum_{i=1}^n x_i \mathbf{b}_i^\star \text{ s.t. } -(t_i+1)/2 < x_i \leq -t_i/2 \text{ or } t_i/2 < x_i \leq (t_i+1)/2\}$. Then:*

$$\mathbb{E}\{\mathcal{C}_{\mathbb{N}}(\mathbf{t})\} = \sum_{i=1}^{n} \left(\frac{t_i^2}{4} + \frac{t_i}{4} + \frac{1}{12}\right) \|\mathbf{b}_i^\star\|^2 \quad \text{and} \quad \mathbb{V}\{\mathcal{C}_{\mathbb{N}}(\mathbf{t})\} = \sum_{i=1}^{n} \left(\frac{t_i^2}{48} + \frac{t_i}{48} + \frac{1}{180}\right) \|\mathbf{b}_i^\star\|^4$$

This suggests that the natural partition is better than Babai's partition: we will return to this topic in Sect. 6.

4.2 Discrete Pruning from Lattice Partitions

Any L-partition $(\mathcal{C}, T)$ defines a partition $\mathbb{R}^n = \cup_{t \in T} \mathcal{C}(t)$. Discrete pruning is simply obtained by choosing a finite number of cells $\mathcal{C}(t)$ to enumerate, as done by Algorithm 4: discrete pruning is parametrized by a finite set $U \subseteq T$, which specifies which cells to enumerate. Discrete pruning is therefore a pruned enumeration with pruning set:

$$P = \cup_{\mathbf{t} \in U} \mathcal{C}(\mathbf{t})$$

Algorithm 4. Discrete Pruning from Lattice Partitions

Input: A lattice partition $(\mathcal{C}(), T)$, a finite subset $U \subseteq T$ and a radius R.
Output: $L \cap S \cap P$ where $S = \text{Ball}_n(R)$ and $P = \cup_{t \in U} \mathcal{C}(t)$.

1: $\mathcal{R} = \varnothing$
2: **for** $t \in U$ **do**
3: Enumerate $L \cap \mathcal{C}(t)$: if the output vector has norm $\leq R$, add the vector to the set $\mathcal{R}$.
4: **end for**

The algorithm performs exactly k partition-enumerations, where $k = \#U$ is the number of cells of discrete pruning, and each partition-enumeration runs in polynomial time by definition of the lattice partition. So the running time is $\#U$ polynomial-time operations: one can decide how much time should be spent.

Since the running time is easy to evaluate, the only difficulty is to estimate the probability of success. Based on Heuristic 2, the probability can be derived from:

$$\text{vol}(S \cap P) = \sum_{\mathbf{t} \in U} \text{vol}(S \cap \mathcal{C}(\mathbf{t})). \tag{1}$$

Thus, the problem is reduced to computing the volume $\mathrm{vol}(S \cap \mathcal{C}(\mathbf{t}))$ of the intersection of a ball with a cell. If we want to maximize the probability of success for a given effort (*i.e.* for a fixed number k of cells), it suffices to select the k cells which maximize $\mathrm{vol}(S \cap \mathcal{C}(\mathbf{t}))$ among all the cells: we will study this topic in Sect. 6 for the natural partition.

The hardness of computing $\mathrm{vol}(S \cap \mathcal{C}(\mathbf{t}))$ depends on the lattice partition, but for Babai's partition and the natural partition, this can be reduced to computing the volume $\mathrm{vol}(S \cap H)$ where S is a ball and H is a box, which is exactly the topic of Sect. 5:

- In the case of Babai's partition, each cell $\mathcal{C}_{\mathbb{Z}}(\mathbf{t})$ is already a box.
- In the case of the natural partition, each cell $\mathcal{C}_{\mathbb{Z}}(\mathbf{t})$ is the union of 2^j symmetric (non-overlapping) boxes, where j is the number of non-zero coefficients of $\mathbf{t}$. It follows that $\mathrm{vol}(\mathcal{C}_{\mathbb{Z}}(\mathbf{t}) \cap S) = 2^j \mathrm{vol}(H \cap S)$, where H is any of these 2^j boxes.

Interestingly, in Sect. 6.3, we show how to approximate well the sum of (1) without computing all the terms, using only a constant number of terms.

4.3 Revisiting Schnorr's Random Sampling and the Fukase-Kashiwabara Variant

Here, we show that Schnorr's random sampling and its variants, including the Fukase-Kashiwabara variant, can all be viewed as special cases of discrete pruning.

Schnorr's Random Sampling. Let $B = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ be a basis of a full-rank lattice L in $\mathbb{Z}^n$: denote by $B^\star = (\mathbf{b}_1^\star, \ldots, \mathbf{b}_n^\star)$ its Gram-Schmidt orthogonalization. Let $1 \leq u \leq n-1$ be an integer parameter. Schnorr's random sampling [30] outputs all points $\mathbf{v} \in L$ of the form $\sum_{i=1}^{n} \mu_i \mathbf{b}_i^\star$ such that $\mu_n = 1$ and the remaining $\mu_i \in \begin{cases} [-1/2, 1/2(& \text{if } i \leq n-(u+1) \\ [-1, 1(& \text{if } n-u \leq i \leq n-1 \end{cases}$

This is equivalent to pruned enumeration with a pruning set defined as the following (non-centered) box of parameter u:

$$P_u = \left\{ \sum_{i=1}^{n} x_i \mathbf{b}_i^\star \text{ s.t. } \begin{array}{c} -1/2 \leq x_i < 1/2, \text{if } i \leq n-(u+1) \\ -1 \leq x_i < 1, \text{if } n-u \leq i \leq n-1 \\ 1/2 \leq x_i < 3/2, \text{if } i = n \end{array} \right\} \tag{2}$$

Clearly $\mathrm{vol}(P_u) = 2^u \mathrm{covol}(L)$. Curiously, the box P_u has slightly bigger moments than $\cup_{\mathbf{t} \in U_u} \mathcal{C}_{\mathbb{N}}(\mathbf{t})$ where $U_u = \{(0, ..., 0, t_{n-u}, ..., t_{n-1}, 1) \in \{0,1\}^n\}$ is a finite set defining discrete pruning with the natural partition: the corresponding pruning set also has volume $2^u \mathrm{covol}(L)$. Schnorr's box actually corresponds to discrete pruning with the same finite set U_u of tags, but using a different hybrid lattice partition, which matches the natural partition for the first $n-1$ coordinates,

and Babai's partition for the n-th coordinate: namely, $T = \mathbb{N}^{n-1} \times \mathbb{Z}$ and for any $\mathbf{t} \in T$

$$\mathcal{C}_{\text{Schnorr}}(\mathbf{t}) = \left\{ \sum_{i=1}^{n} x_i \mathbf{b}_i^\star \text{ s.t. } \begin{array}{r} -(t_i+1)/2 < x_i \leq -t_i/2 \text{ or } t_i/2 < x_i \leq (t_i+1)/2 \text{ if } i \leq n-1 \\ -1/2 \leq x_n - t_n < 1/2 \end{array} \right\}.$$

Based on Heuristic 2, the success probability of random sampling can be deduced from $\text{vol}(P_u \cap S)$, where P_u is a non-centered box and S is a ball: the computation of such volumes is exactly the topic of Sect. 5. There, the following computations will be useful:

Lemma 4 (Moments of Schnorr's box). *Let $B = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ be a basis of a full-rank lattice L in $\mathbb{R}^n$, and P_u be Schnorr's box of parameter u defined by (2). Then:*

$$\mathbb{E}\{P_u\} = \frac{\sum_{i=1}^{n-(u+1)} \|\mathbf{b}_i^\star\|^2}{12} + \frac{\sum_{i=n-u}^{n-1} \|\mathbf{b}_i^\star\|^2}{3} + \frac{13\|\mathbf{b}_n^\star\|^2}{12}$$

$$\mathbb{V}\{P_u\} = \frac{\sum_{i=1}^{n-(u+1)} \|\mathbf{b}_i^\star\|^4}{180} + \frac{\sum_{i=n-u}^{n-1} \|\mathbf{b}_i^\star\|^4}{45} + \frac{61\|\mathbf{b}_n^\star\|^4}{180}$$

In some variants of random sampling (see [4,30]), one actually considers a random subset of U_u of size k for some $k \ll 2^u$: based on Heuristic 2, the success probability of random sampling can be deduced from $\text{vol}(\mathcal{C}(\mathbf{t}) \cap S)$, where $\mathcal{C}(\mathbf{t})$ is the union of symmetric non-overlapping boxes. Again, the problem can be reduced to the volume computation of Sect. 5.

It can easily be checked that the other variants by Buchmann-Ludwig [4] can also be viewed as discrete pruning.

The Fukase-Kashiwabara Variant. As mentioned previously, Fukase and Kashiwabara [8] recently introduced an encoding of lattice points onto $\mathbb{N}^n$, which turns out to be the encoding derived from the natural partition. Their variant proceeds by enumerating all lattice points having certain encodings. In our terminology, this can immediately be rewritten as discrete pruning with the natural partition, where the finite set of tags has size approximately 5×10^7.

They do not provide a general algorithm to select tags: however, they explain which tags they selected to solve the SVP challenges, so this only applies to certain settings and fixed dimensions. Here is an example in dimension n in the range $120 - 140$. The selection proceeds in two stages:

- First, they select a large set of candidates $V \subseteq \mathbb{N}^n$, formed by all tags $(t_1, \ldots, t_n) \in \mathbb{N}^n$ such that:
  - all $t_i \in \{0, 1, 2\}$
  - the total number of indexes i such that $t_i = 1$ is ≤ 13, and the indexes i such that $t_i = 1$ all belong to $\{n - 55 + 1, \ldots, n\}$.
  - the total number of indexes i such that $t_i = 2$ is ≤ 1, and the indexes i such that $t_i = 2$ all belong to $\{n - 15 + 1, \ldots, n\}$.
- For each $\mathbf{t} \in V$, they compute $\mathbb{E}\{\mathcal{C}_{\mathbb{N}}(\mathbf{t})\}$ using the formula of Corollary 3.
- The final set U is formed by the 5×10^7 tags $\mathbf{t} \in V$ which have the smallest $\mathbb{E}\{\mathcal{C}_{\mathbb{N}}(\mathbf{t})\}$.

4.4 Optimizations

If the discrete pruning set U has exactly k elements, then the running time is k polynomial-time operations. However, from a practical point of view, it is important to decrease as much as possible the cost of the polynomial-time operation.

First, one can abort the enumeration of a cell if we realize that the lattice vector inside the cell will be outside the ball S: this is similar to what is done during enumeration.

Second, we can speed up the computation by regrouping cells. A good example is Schnorr's random sampling. We can view Schnorr's pruning set P_u as the union of 2^u cells, but when we want to enumerate all lattice points inside $S \cap P_u$, it is better to view it as a single box: discrete pruning can be rewritten here as some variant of enumeration. More generally, depending on the set U of tags, we can recycle some computations: similar tricks were used for pruned enumeration [11].

Third, we note that all the cell enumerations can be performed in parallel: discrete pruning is easier to parallelize than continuous pruning. It seems that discrete pruning should be better suited to special hardware than continuous pruning.

5 Ball-Box Intersections

In this section, we are interested in computing the volume of the intersection between a ball and a box, either exactly by a formula or approximately by an algorithm. More precisely, we are interested in $\mathrm{vol}(B \cap H)$, where B is the ball of center $\mathbf{c} \in \mathbb{R}^n$ and radius R, and H is the following box:

$$H = \{(x_1, \ldots, x_n) \in \mathbb{R}^n \text{ s.t. } \alpha_i \leq x_i \leq \beta_i\},$$

where the α_i's and β_i's are given. Without loss of generality, we may assume that $\mathbf{c} = 0$ after suitable translation, and $R = 1$ after suitable scaling. Hence, we are interested in:

$$\begin{aligned} \mathrm{Vol}\left(\mathrm{Ball}_n(1) \cap H\right) &= \mathrm{Vol}\left(\mathrm{Ball}_n(1) \cap \prod_{i=1}^{n} [\alpha_i, \beta_i]\right) \\ &= \prod_{i=1}^{n} (\beta_i - \alpha_i) \Pr_{(x_1,\ldots,x_n) \leftarrow \prod_{i=1}^{n} [\alpha_i, \beta_i]} \left[\sum_{i=1}^{n} x_i^2 \leq 1\right] \end{aligned} \tag{3}$$

The section is organized as follows. In Sect. 5.1, we give a rigorous asymptotical estimate of (3) when the box is sufficiently balanced, but we show that it is ill-suited to the case of discrete pruning. In Sect. 5.2, we provide two exact formulas for (3) as infinite series, due to respectively Constales and Tibken [27]: these infinite series give rise to approximation algorithms by truncating the series. In Sect. 5.3, we give a heuristic method to approximate (3), based on fast inverse

Laplace transforms: this method is referred as FILT in the remaining of the paper. Section 5.4 provides an experimental comparison of the running time of the three methods of Sects. 5.2 and 5.3, in the context of discrete pruning: it turns out that in high dimension, the FILT method outperforms the other two.

We note that Buchmann and Ludwig [4, Theorem 1] (more details in [20, Theorem 18]) implicitly adressed the computation of (3): the main part of [4, Algorithm 3] can be viewed as a heuristic approximation algorithm based on the Discrete Fourier transform, but no experimental result seems to be reported in [4,20].

5.1 Asymptotical Analysis

The following result shows that the volume of the intersection of a ball with a "balanced" box can be asymptotically computed, because the right-hand probability of (3) can be derived from the central limit theorem:

Theorem 3. *Let $C_1, C_2 > 0$ be constants. Let $H = \{\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n$ s.t. $\alpha_i \leq x_i \leq \beta_i\}$, where $C_1 \leq \beta_i - \alpha_i$ and $\max(|\alpha_i|, |\beta_i|) \leq C_2$. Then $Y = (\|\mathbf{x}\|^2 - \mathbb{E}\{H\})/\sqrt{\mathbb{V}\{H\}}$ has zero mean and variance one, and converges in distribution to the normal distribution,* i.e. *for all $y > 0$:*

$$\lim_{n\to\infty} \Pr_{\mathbf{x}\in H}\left(\|\mathbf{x}\|^2 \leq \mathbb{E}\{H\} + y\sqrt{\mathbb{V}\{H\}}\right) = \frac{1}{2}\left(1 + \mathrm{erf}(y/\sqrt{2})\right),$$

where

$$\mathbb{E}\{H\} = \sum_{i=1}^{n}\left(\frac{\alpha_i^2 + \beta_i^2 + \alpha_i\beta_i}{3}\right) \quad \textit{and}$$

$$\mathbb{V}\{H\} = \sum_{i=1}^{n}\left(\frac{4}{45}\alpha_i^4 - \frac{1}{45}\alpha_i^3\beta_i - \frac{2}{15}\alpha_i^2\beta_i^2 - \frac{1}{45}\alpha_i\beta_i^3 + \frac{4}{45}\beta_i^4\right).$$

Unfortunately, we cannot apply Theorem 3 when the box H is Schnorr's box which, after suitable rotation, corresponds to:

- If $i \leq n - (u+1)$, then $\beta_i = \|\mathbf{b}_i^\star\|/2$ and $\alpha_i = -\beta_i$;
- If $n - u \leq i \leq n - 1$, then $\beta_i = \|\mathbf{b}_i^\star\|$ and $\alpha_i = -\beta_i$;
- $\alpha_n = \|\mathbf{b}_n^\star\|/2$ and $\beta_n = 3/2\|\mathbf{b}_n^\star\|$

If the basis B is LLL-reduced, then the $\|\mathbf{b}_i^\star\|$'s typically decrease geometrically, which means that the (α_i, β_i)'s do not satisfy the assumptions of Theorem 3. Furthermore, experiments show that in practice, the distribution of the Y defined in Theorem 3 is not normal (see Fig. 2) as its left-tail is below that of the normal distribution and its right-tail is over. Worse, the distance with the normal distribution actually increases with the dimension: see Fig. 3. However, if we fix the dimension and apply stronger and stronger lattice reduction, we expect the box to become more and more "balanced": this is confirmed by Fig. 4, which shows that the more reduced the basis is, the closer Y is to the normal distribution.

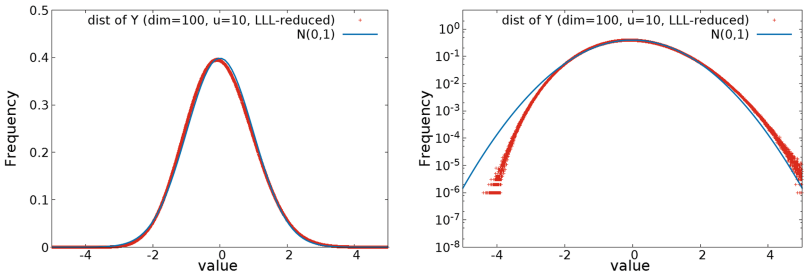

Fig. 2. Comparison of the normal distribution with the experimental distribution of Y defined in Theorem 3 with Schnorr's box (with $u = 10$) over an LLL-reduced basis in dimension 100. Both tails significantly deviate from normal tails. On the right, there is a log-scale.

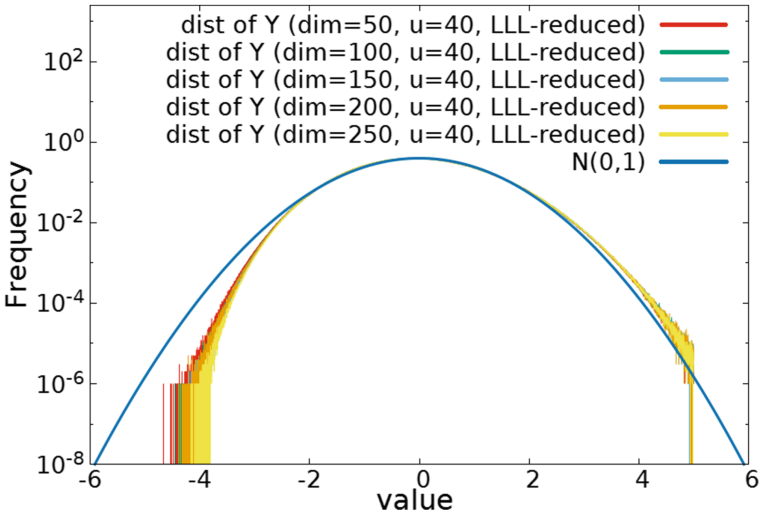

Fig. 3. Evolution of the distribution of Y defined in Theorem 3 with Schnorr's box (with $u = 10$) over an LLL-reduced basis, as the dimension increases: it becomes less and less normal.

If the box is "unbalanced", it is always possible to upper bound and lower bound the right-hand probability of (3). For instance, an upper bound follows from Hoeffding's bound:

Lemma 5. *Let* $H = \{\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n \text{ s.t. } \alpha_i \leq x_i \leq \beta_i\}$. *Then for any* $y > 0$*:*

$$\Pr_{\mathbf{x} \in H}\left(\|\mathbf{x}\|^2 \leq \mathbb{E}\{H\} - y\right) \leq e^{-2y^2/\sum_{i=1}^{n}(\beta_i - \alpha_i)^2}.$$

Proof. It suffices to apply Hoeffding's bound with $y > 0$ and the n independent variables $-X_i = -x_i^2$. □

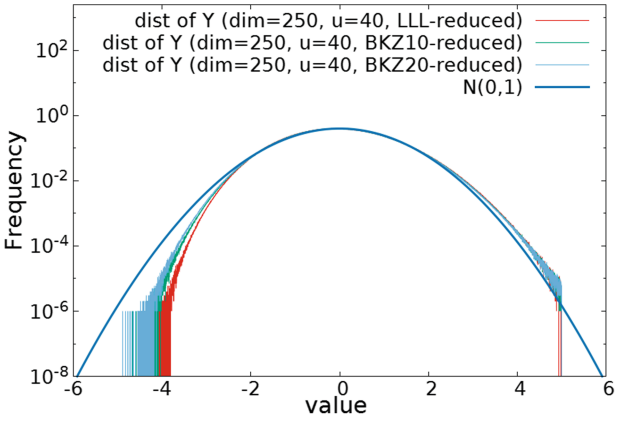

Fig. 4. Evolution of the distribution of Y defined in Theorem 3 with Schnorr's box (with $u = 40$) in dimension 250, with varying reductions: it becomes closer to normal as the basis becomes more reduced

However, these upper bounds are typically very pessimistic for the type of boxes we are interested in. Reciprocally, in the spirit of Schnorr [30] (see also [4,9]), it is also possible to give a lower bound on the right-hand probability of (3), by considering the largest sub-box J of H which is contained in $\mathrm{Ball}_n(1)$, in which case $\mathrm{vol}(\mathrm{Ball}_n(1) \cap H) \geq \mathrm{vol}(J)$. This can lead to an asymptotic lower bound if we are able to conveniently bound the side-lengths of H, *i.e.* the $\|\mathbf{b}_i^\star\|$'s. This is why Schnorr [30] introduced the Geometric Series Assumption (GSA), but this only holds in an approximate sense, which creates some problems (see [4]): for instance, the GSA implies that all $\|\mathbf{b}_i^\star\|$ are always $\leq \|\mathbf{b}_1\|$ for $i \geq 2$, but in practice, it can happen that some $\mathbf{b}_i^\star$ is larger than $\mathbf{b}_1$. To prevent this problem, one can use instead absolute bounds: for instance, Fig. 5 shows that in practice, for a random LLL-reduced basis, $\max_B \|\mathbf{b}_i^\star\|/\|\mathbf{b}_1\|$ can be upper bounded by a geometric sequence indexed by i, with parameters independent of n. However, the lower bound obtained is again typically very pessimistic for the type of boxes we are interested in.

5.2 Exact Formulas

Here, we provide two exact formulas for the intersection volume as infinite series, by sligthly generalizing the works of respectively Constales and Tibken [27], who studied the special case of a zero-centered cube.

Fourier Series. We first generalize Constales' method based on Fourier series. Let $S(x) = \int_0^x \sin(t^2)dt$ and $C(x) = \int_0^x \cos(t^2)dt$ be the Fresnel integrals.

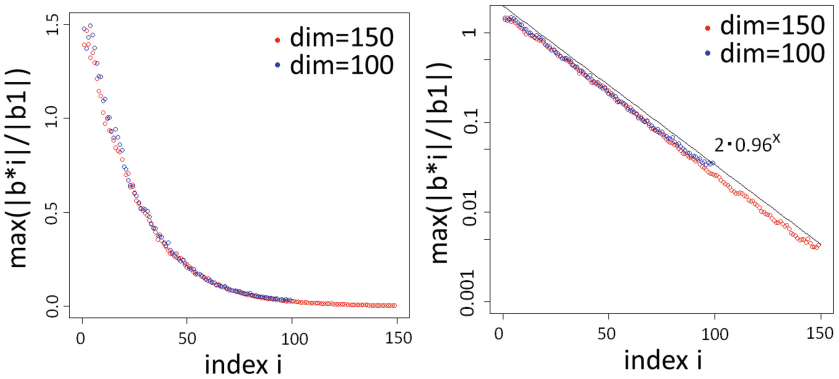

Fig. 5. Evolution of $\max_B \|\mathbf{b}_i^\star\|/\|\mathbf{b}_1\|$ over hundreds thousands LLL-reduced bases, as $i \geq 2$ increases, in dimensions 100 (blue) and 150 (red): the right-hand graph is in log-scale. (Color figure online)

Theorem 4. *Let $\alpha_j \leq \beta_j$ for $1 \leq j \leq n$. Let $\ell = \sum_{j=1}^n \max(\alpha_j^2, \beta_j^2)$. Then:*

$$\mathrm{vol}(\mathrm{Ball}_n(1) \cap \prod_{j=1}^n [\alpha_j, \beta_j]) = \begin{cases} \prod_{j=1}^n |\beta_j - \alpha_j| & \text{if } \ell \leq 1 \\ \left(\frac{1}{2} - \frac{\sum_{j=1}^n \alpha_j^2 + \beta_j^2 + \alpha_j\beta_j}{3\ell} + \frac{1}{\ell} + \frac{1}{\pi}\mathrm{Im}\sum_{k=1}^{\infty} \frac{\Phi(-2\pi k/\ell)}{k} e^{2i\pi k/\ell}\right) \prod_{j=1}^n (\beta_j - \alpha_j) & \text{if } \ell > 1 \end{cases} \quad (4)$$

where Φ is defined as

$$\Phi(\omega) = \prod_{j=1}^n \frac{(C(\beta_j\sqrt{|\omega|}) - C(\alpha_j\sqrt{|\omega|})) + i\mathrm{sgn}(\omega)(S(\beta_j\sqrt{|\omega|}) - S(\alpha_j\sqrt{|\omega|}))}{(\beta_j - \alpha_j)\sqrt{|\omega|}}.$$

A proof can be found in the full version: it is just a slight generalization of the proof of Constales [27]. An approximation algorithm can be derived by truncating the infinite series of (4) to N terms: computing each term costs approximately n operations, where an operation is dominated by the computation of a constant number of Fresnel integrals. The computation of Fresnel integrals is classical: for instance, it can be done by reduction to erf computations.

Multidimensional Fourier Transforms. We now generalize Tibken's formula based on the n-dimensional Fourier transform.

Theorem 5. *Let $\alpha_j \leq \beta_j$ for $1 \leq j \leq n$. Then:*

$$\mathrm{vol}(\mathrm{Ball}_n(1) \cap \prod_{j=1}^{n}[\alpha_j, \beta_j]) = \frac{1}{(4\pi)^{n/2}} \left(\frac{I(\frac{1}{2n+4}, 0)}{\Gamma(n/2+1)} + \sum_{k=2}^{\infty} \frac{L_k^{n/2}(n/2+1) I(\frac{1}{2n+4}, k)}{\Gamma(k+n/2+1)(2n+4)^k} \right) \tag{5}$$

where $L_k^{\alpha}(x)$ denotes the generalized Laguerre polynomial, Γ is the classical Gamma function and:

$$I(\lambda,0) = \pi^n \prod_{j=1}^{n} \left(\mathrm{erf}\left(\frac{\beta_j}{2\sqrt{\lambda}} \right) - \mathrm{erf}\left(\frac{\alpha_j}{2\sqrt{\lambda}} \right) \right) \quad \text{and} \quad I(\lambda, k) = (-1)^k \left(\frac{\partial}{\partial \lambda} \right)^k I(\lambda, 0)$$

Again, an approximation algorithm can be derived by truncating the infinite series. The first term of (5) is easy to compute from the erf, and turns out to give a much better approximation than the central limit theorem for Schnorr's box. But the right-hand sum terms are trickier to compute (see the full version for details), due to the derivative in the definition of $I(\lambda, k)$ and the presence of Laguerre polynomials.

5.3 Numerical Approximation from Fast Inverse Laplace Transforms (FILT)

We now present the method we used in practice to approximate the volume, which is based on the Laplace transform. The starting point is similar to the Fourier series approach, but it deviates afterwards. For a function $f(x)$ defined over $x \geq 0$, its *Laplace transform* $F = \mathcal{L}\{f\}$ is defined over $\mathbb{C}$ as $F(s) = \int_0^{\infty} f(t) e^{-st} dt$. The inverse transform is given by the Bromwich integral

$$f(t) = \frac{1}{2\pi i} \int_{c-\infty i}^{c+\infty i} F(s) e^{st} ds, \tag{6}$$

where the integration is done along the vertical line $\mathrm{Re}(s) = c$ in the complex plane such that c is greater than the real part of all singularities of $F(s)$. If $g(t) = \int_0^t f(\tau) d\tau$, then $\mathcal{L}\{g\}(s) = \frac{1}{s}\{f\}$. Thus, if X is a non-negative random variable with probability density function $f(x)$, then its cumulative distribution function $F_X(x)$ satisfies: $F_X(x) = \mathcal{L}^{-1}\{\frac{1}{s}\mathcal{L}\{f\}(s)\}(x)$. Thus, if we denote by $\rho_{\sum_{j=1}^n x_j^2}$ the probability density function of $\sum_{i=1}^n x_i^2$, then the right-hand probability of (3) is given by:

$$\Pr_{(x_1,\ldots,x_n) \leftarrow \prod_{i=1}^n [\alpha_i, \beta_i]} \left[\sum_{i=1}^{n} x_i^2 \leq 1 \right] = \mathcal{L}^{-1}\left\{ \frac{1}{s} \mathcal{L}\{\rho_{\sum_{i=1}^n x_i^2}\}(s) \right\}(1). \tag{7}$$

When x_i is uniform over $[\alpha_i, \beta_i]$, the p.d.f. of x_i^2 is

$$\rho_{x_i^2}(z) = \begin{cases} \dfrac{1}{2(\beta_i - \alpha_i)\sqrt{z}} & (\alpha_i^2 \leq z \leq \beta_i^2) \\ 0 & \text{otherwise} \end{cases}.$$

Thus, the p.d.f. of $x_1^2 + \cdots + x_n^2$ is given by their convolution: $\rho_{x_1^2+\cdots+x_n^2}(z) = \rho_{x_1^2}(z) * \cdots * \rho_{x_n^2}(z)$, where

$$(f * g)(t) = \int_{\tau=0}^{t} f(\tau)g(t-\tau)d\tau.$$

However, the Laplace transform of a convolution is simply the product of the transforms: if $f_1(t)$ and $f_2(t)$ be two functions with Laplace transform $F_1(s)$ and $F_2(s)$ respectively, then

$$\mathcal{L}\{f_1 * f_2\}(s) = F_1(s) \cdot F_2(s).$$

In our case, the Laplace transform of each individual pdf is given by

$$\mathcal{L}\left\{\rho_{x_i^2}\right\}(s) = \frac{\sqrt{\pi}\left(\mathrm{erf}(\beta_i\sqrt{s}) - \mathrm{erf}(\alpha_i\sqrt{s})\right)}{2(\beta_i - \alpha_i)\sqrt{s}}. \tag{8}$$

Thus,

$$\mathcal{L}\left\{\rho_{\sum_{i=1}^n x_i^2}\right\}(s) = \left(\frac{\pi}{s}\right)^{n/2} \prod_{i=1}^{n} \frac{\left(\mathrm{erf}(\beta_i\sqrt{s}) - \mathrm{erf}(\alpha_i\sqrt{s})\right)}{2(\beta_i - \alpha_i)}, \tag{9}$$

and computing (7) is reduced to computing the inverse Laplace transform of (9) at 1, namely:

$$\frac{\pi^{n/2-1}}{2i} \int_{c-\infty i}^{c+\infty i} \frac{e^s}{s^{n/2+1}} \prod_{j=1}^{n} \frac{\left(\mathrm{erf}(\beta_i\sqrt{s}) - \mathrm{erf}(\alpha_i\sqrt{s})\right)}{2(\beta_i - \alpha_i)} ds \tag{10}$$

for a real number c within a certain range.

We used Hosono's method [14] to invert the Laplace transform: given $F(s)$, we want to compute its inverse $f(t)$. Let $\gamma > \gamma_0$ in the region of convergence. Hosono used the following approximation of the exponential function, for $\gamma_1 > \gamma_0$:

$$e^s \approx E(s, \gamma_1) := \frac{e^{\gamma_1}}{2\cosh(\gamma_1 - s)}$$

which has singularity points at

$$s_m = \gamma_1 + \left(m - \frac{1}{2}\right)\pi i \text{ for } m \in \mathbb{N}.$$

Considering the integral along the sides of the box $\{\gamma_1 + x + yi : |x| < a, |y| < R\}$ for some small number a and large number R. Letting $R \to \infty$, and by the residue theorem, we have

$$f(t) = \frac{1}{2\pi i} \int_{c-\infty i}^{c+\infty i} F(s)e^{st} ds \approx \frac{e^{\gamma_1}}{t} \sum_{m=1}^{\infty} (-1)^m \mathrm{Im} F\left(\frac{\gamma_1 + (m - 1/2)\pi i}{t}\right). \tag{11}$$

Here, we used the fact that $\mathrm{Re}F$ and $\mathrm{Im}F$ are respectively odd and even functions when $f(t)$ is a real function. Thus, by choosing a suitable γ_1 and truncating the sum of (11) to N terms, we can obtain an approximation of the inverse Laplace transform $f(t)$, and therefore approximate (10).

Choice of Preimages. Since our function includes $\sqrt{s}$, we need to specify which square root over $\mathbb{C}$, in such a way that the function is continuous along the path $(c - \infty i, c + \infty i)$. Since the path is in $\mathrm{Re} > 0$, we choose the primal value of $\sqrt{s}$ in the area $|\arg z| < \pi/4$.

Speeding up the convergence by Euler's series transform. To approximate the sum

$$\sum_{m=1}^{\infty}(-1)^m F_m := \sum_{m=1}^{\infty}(-1)^m \mathrm{Im} F\left(\frac{\gamma_1 + (m-1/2)\pi i}{t}\right)$$

by a small number of terms, we apply the Euler's series transform to its last 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 $\Delta^{j-1}F_{k+1}$ is the forward difference $\sum_{i=0}^{j-1}(-1)^i\binom{j-1}{i}F_{j+k-i}$. The van Wijngaarden transformation gives us a simple algorithm. Let $s_{0,j} = \sum_{m=k+1}^{k+j}(-1)^m F_m$ for $j = 0, 1, \ldots J$ and compute $s_{\ell+1,j} = (s_{\ell,j} + s_{\ell,j+1})/2$ for all $\ell \geq j$. Finally, $s_{J,0}$ is an approximation of the partial sum.

5.4 Experimental Comparison

We give experimental results to compare the running time and accuracy required by the previous three methods (namely, Constales' formula (4), Tibken's formula (5), and our FILT method (11) to approximate the intersection volume (3): in each case, it depends on a number of terms, and only experiments can tell how many terms are required in practice.

Accuracy of the FILT Method. We first report on the accuracy of the FILT method, which turns out to require the least number of terms in practice. To check accuracy, we compared with a very good approximation of the volume, which we denote by "convolution" in all the graphs: it was obtained with the Fourier series method with a huge number of terms. To visualize results more easily, most of the graphs display an approximation of a pdf, instead of a cdf: Fig. 6 shows the accuracy of FILT on (8) in dimension one, when the function is the pdf of x_j^2:

$$\frac{e^{\gamma_1}}{t}\sum_{m=1}^{N}(-1)^m \mathrm{Im} F\left(\frac{\gamma_1 + (m-1/2)\pi i}{t}\right) \approx \begin{cases} \dfrac{1}{2(\beta_j - \alpha_j)\sqrt{t}} & (\alpha_j^2 \leq z \leq \beta_j^2) \\ 0 & \text{otherwise} \end{cases}. \tag{12}$$

for γ_1 and N, where

$$F(s) = \frac{\sqrt{\pi}\,(\mathrm{erf}(\beta_j\sqrt{s}) - \mathrm{erf}(\alpha_j\sqrt{s}))}{2(\beta_j - \alpha_j)\sqrt{s}}.$$

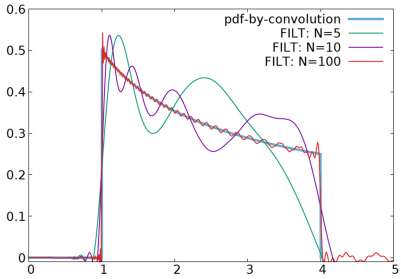

Fig. 6. Accuracy of (12) for (8) as the number N of terms increases.

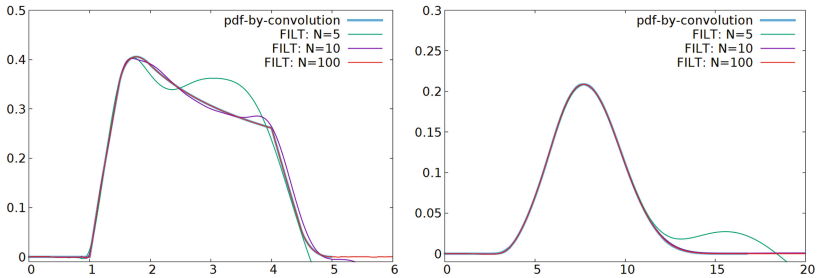

Fig. 7. Accuracy of (11) for evaluating (9) as the number N of terms increases. On the left, $n = 3$. On the right, $n = 10$.

For higher dimensions $n = 3$ and $n = 10$, Fig. 7 shows the accuracy of the method for evaluating (9).

When we apply (10) to (7) to compute the volume of the intersection, the method is very efficient: Fig. 8 shows the accuracy of the method in dimension 140 for the pdf and the CDF of the target random variable, as the number N increases. Here, the box comes from a random natural cell of tag $(0, \ldots, 0, t_{131}, \ldots, t_{140})$ where $t_j \in \{0, 1, 2\}$.

Comparison with the Methods of Constales and Tibken. For each dimension, we generated a random LLL basis and considered the box H corresponding to the tag which as the 1000-th smallest expectation. Then we compute the intersection volume $V = \mathrm{Vol}(Ball_n(0, 1.2 \cdot \mathrm{GH}(L)) \cap H)$ using the FILT method with sufficiently many terms as the reference value r. Table 1 shows what is the number of required terms to achieve a small relative error in practice, that is, the minimum k such that

$$\left| \frac{r - s_k}{r} \right| < 10^{-5}$$

where s_k is the approximation computed with k terms. All computations were done using `cpp_dec_float` $<$ `50` $>$ of the `boost` library which can use 50-decimal floating-point numbers. For the computation of the Constales and FILT method,

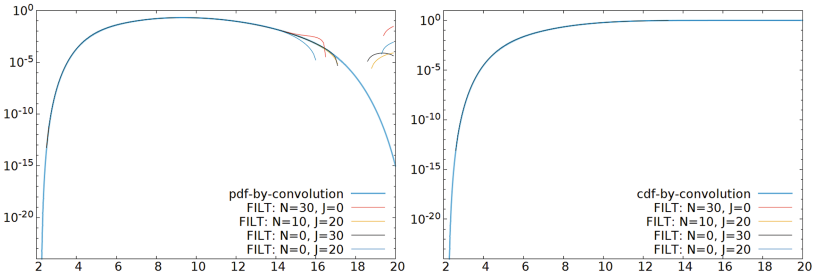

Fig. 8. Accuracy of the van Wijngaarden transformation in dimension 140. On the left, pdf; and on the right, CDF.

Table 1. Required number of terms and running times for the three volume methods

Method/Dimension		40	50	60	80	100	150
Constales	♯ terms	34	34	42	67	109	890
	Time[sec.]	0.55	0.65	0.92	1.9	3.0	45.1
Tibken	♯ terms	34	39	54	-	-	-
	Time[sec.]	5.9	19.5	362.5	-	-	-
FILT	♯ terms	39	46	46	46	43	34
	Time[sec.]	1.06	1.54	1.72	1.96	2.06	1.88

the van Wijngaarden transformation (Sect. 5.3) is used to speed up the convergence. In our FILT method, we used a heuristic value $\gamma_1 = \max(50, 30+3\sqrt{dim})$.

From Table 1, we conclude that the FILT method is the best method in practice for discrete pruning: its running time is around 2 s across dimension 40–150, and its required number of terms stays around 40. On the other hand, Tibken's method gets quickly impractical: our implementation requires a huge precomputation table (see the full version for details), which was not taken into account, and we see that the number of terms increases, which creates a sharp increase in the running time. Constales' method is very competitive with FILT until dimension 80, and even faster in dimension ≤ 60, but its number of terms starts to increase from dimension 60, to the point of making the method significantly slower than FILT in dimension 150.

We note that the running time of FILT (resp. Constales' method) is dominated by the computation of $\mathrm{erf}(\cdot)$ (resp. $S(\cdot)$ and $C(\cdot)$): interestingly, in the context of discrete pruning, when we want to compute the intersection volumes for many tags over a fixed basis, we can recycle some computations because $\mathrm{erf}(\beta_i\sqrt{s})$ only depends on t_i. Since tags of interest typically have few non-zero coefficients, and that these non-zero coefficients are small, there is a lot of overlap: given two tags $\mathbf{t}$ and $\mathbf{u}$, there are many indices i such that $t_i = u_i$. In practice, when computing the intersection volumes for sufficiently many tags over a common basis, the total running time is decreased by a factor 10–20.

So the running times of Table 1 can be further decreased when it is applied to discrete pruning: the amortized running time for FILT is a fraction of a second.

6 Optimizing Discrete Pruning with the Natural Partition

We saw in Sect. 4.2 that to optimize discrete pruning for a given effort (*i.e.* for a fixed number M of cells), it suffices to select the M cells which maximize $\mathrm{vol}(S \cap \mathcal{C}(\mathbf{t}))$ among all the cells. In the case of the natural partition, the computation of each $\mathrm{vol}(S \cap \mathcal{C}(\mathbf{t}))$ can be reduced to the computation of $\mathrm{vol}(S \cap H)$ where H is a sub-box of $\mathcal{C}(\mathbf{t})$. And we would like to identify which tags maximize $\mathrm{vol}(S \cap H)$.

Section 5 gave methods to compute $\mathrm{vol}(S \cap H)$ very efficiently, say less than one second. Unfortunately, this is too inefficient to make an online selection, since we may want to process a huge number of cells. Even if we preprocess the computation of tags by using a profile of the Gram-Schmidt norms $\|\mathbf{b}_i^\star\|$, this will be too slow.

In this section, we study practical heuristic methods to select optimal tags for the natural partition: we use our volume computations of Sect. 5 to check the quality of our selection, so the results of Sect. 5 are very useful. The section is organized as follows. In Sect. 6.1, we observe a strong correlation between $\mathrm{vol}(S \cap \mathcal{C}(\mathbf{t}))$ and $\mathbb{E}\{\mathcal{C}_{\mathbb{N}}(\mathbf{t})\}$ for most tags $\mathbf{t}$. This leads us to select the cells which minimize $\mathbb{E}\{\mathcal{C}_{\mathbb{N}}(\mathbf{t})\}$, which avoids any computation of $\mathrm{vol}(S \cap \mathcal{C}(\mathbf{t}))$: Sect. 6.1 explains how to do so efficiently. In Sect. 6.3, we show how to check much more quickly the quality of a discrete pruning set: we show how to speed-up the approximation of a large sum of intersection volumes $\mathrm{vol}(S \cap \mathcal{C}(\mathbf{t}))$, based on statistical inference, which is crucial to estimate the success probability of discrete pruning. Finally, in Sects. 6.4 and 6.5, we compare the quality of cells, depending on the discrete pruning set, and explain how to avoid bad cells.

6.1 Correlation Between Intersection Volumes and Cell Expectations

Inspired by the tag selection of [8] (see Sect. 4.3), we experimentally studied the relationship between the intersection volume $\mathrm{vol}(S \cap \mathcal{C}(\mathbf{t}))$ and the cell expectation $\mathbb{E}\{\mathcal{C}_{\mathbb{N}}(\mathbf{t})\}$. We found that for a fixed-radius centered ball S, for random cells $\mathcal{C}_{\mathbb{N}}(\mathbf{t})$ of the natural partition, there exists a strong negative correlation between $\mathrm{vol}(S \cap \mathcal{C}_{\mathbb{N}}(\mathbf{t}))$ and $\mathbb{E}\{\mathcal{C}_{\mathbb{N}}(\mathbf{t})\}$: roughly speaking, the bigger the expectation $\mathbb{E}\{\mathcal{C}_{\mathbb{N}}(\mathbf{t})\}$, the smaller the volume $\mathrm{vol}(S \cap \mathcal{C}_{\mathbb{N}}(\mathbf{t}))$, as shown in Fig. 9.

More precisely, for a BKZ-20 reduced basis of a 70-dimensional lattice, we computed the best $2^{14} = 16,384$ cells with respect to $\mathbb{E}\{\mathcal{C}_{\mathbb{N}}(\mathbf{t})\}$ (using Corollary 3), and $\mathrm{vol}(S \cap \mathcal{C}_{\mathbb{N}}(\mathbf{t}))$ where S is the ball of radius $\|\mathbf{b}_1\|$. The correlation coefficient of Pearson, Spearman and Kendall are 0.9974191, 0.83618071 and 0.72659780 respectively. The left-hand graph of Fig. 9 shows the intersection volumes of the best cells, sorted by expectation: apart from a few exceptions,

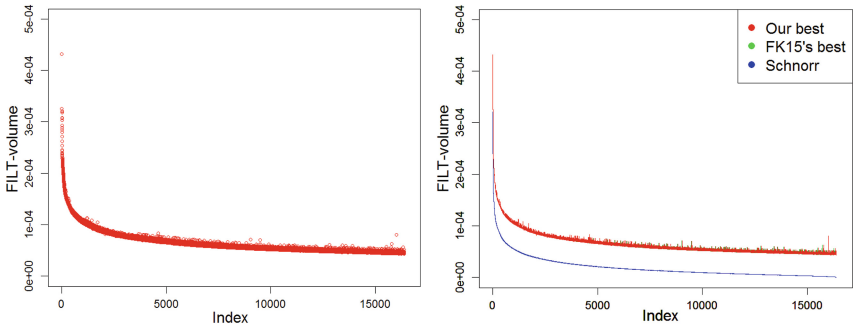

Fig. 9. Graphs of approximated cell-ball intersection volumes sorted by expectation

we see that the curve always decreases, which means that most of the time, the bigger the expectation of the cell, the smaller is the intersection volume.

In other words, if we are only interested in selecting the cells with the largest intersection volumes, we do not need to compute the volume, a near-optimal solution can be found by selecting the cells with the smallest expectation (where the expectation is trivial to compute using Corollary 3), which we will do very efficiently in the next subsection: the volume computation is only necessary when we want to estimate the success probability, not for the selection of tags. We say near-optimal because though the cells with the smallest expectation may not exactly be the cells with the largest intersection volume, most of them are. If we use many cells but miss only a few good cells, it will not affect much the success probability. Note that the selection of cells with the smallest expectation is independent of the radius of the ball S.

As an application of this heuristic supported by experiments, we give evidence that the natural partition is better than Babai's partition: Algorithm 5 shows how to transform efficiently any discrete pruning U for Babai's partition into a discrete pruning V (of the same size) for the natural partition, in such a way that $\sum_{\mathbf{t}\in U} \mathbb{E}(\mathcal{C}_{\mathbb{Z}}(\mathbf{t})) \geq \sum_{\mathbf{t}\in V} \mathbb{E}(\mathcal{C}_{\mathbb{N}}(\mathbf{t}))$. Thus, if we only consider the expectation of cells, we can ignore Babai's partition, and restrict ourselves to the natural partition.

Another interesting application is that for a fixed tag, one can decrease the expectation of its cell by decreasing $\sum_{i=1}^{n} \|\mathbf{b}_i^\star\|^2$: this suggests that the smaller this quantity, the better for discrete pruning.

6.2 Finding the Cells of Lowest Expectations

Recall that the expectation of a cell of the natural partition is given by:

$$\mathbb{E}\{\mathcal{C}_{\mathbb{N}}(\mathbf{t})\} = \sum_{i=1}^{n} \left(\frac{t_i^2}{4} + \frac{t_i}{4} + \frac{1}{12}\right) \|\mathbf{b}_i^\star\|^2$$

We now explain how to compute the tags $\mathbf{t} \in \mathbb{N}^n$ which minimize this expectation.

Algorithm 5. Discrete Pruning: From Babai's partition to the natural partition

Input: A finite set $U \subseteq \mathbb{Z}^n$ defining a discrete pruning with Babai's partition $(\mathcal{C}_{\mathbb{Z}}(), \mathbb{Z}^n)$ over a basis B of a lattice L.

Output: A finite set $V \subseteq \mathbb{N}^n$ of the same cardinal as U, defining a discrete pruning with the natural partition $(\mathcal{C}_{\mathbb{N}}(), \mathbb{N}^n)$ over B such that the finite union of cells has lower $\mathbb{E}$ than for U.

Partition the set U as $U_1, \ldots, U_m$ so that m is minimal and within any subset U_i, all the tags $\mathbf{t} \in U_i$ are the same in absolute value, *i.e.* $(|t_1|, \ldots, |t_n|)$ is constant.
Define the bijection $\nu : \mathbb{Z} \to \mathbb{N}$ by $\nu(z) = 2|z| - 1$ if z is odd, and $\nu(z) = 2|z|$ otherwise.
for $i = 1$ to m **do**
  for $j = 1$ to n **do**
    Compute the number e_j of tags $\mathbf{t} \in U_i$ such that $t_j < 0$
    Compute the number f_j of tags $\mathbf{t} \in U_i$ such that $t_j > 0$
    if $e_j \geq f_j$ **then**
      $\varepsilon_j \leftarrow 1$
    else
      $\varepsilon_j \leftarrow -1$
    end if
  end for
  $V_i \leftarrow \varnothing$
  for $\mathbf{t} \in U_i$ **do**
    for $j = 1$ to n **do**
      $t'_j \leftarrow \nu(\varepsilon_j t_j)$
    end for
    $V_i \leftarrow V_i \cup (t'_1, \ldots, t'_n)$
  end for
end for
Return $\cup_{i=1}^{n} V_i$.

For a fixed sequence $(\|\mathbf{b}_i^\star\|)_{1 \leq i \leq n}$, minimizing $\mathbb{E}\{\mathcal{C}_{\mathbb{N}}(\mathbf{t})\}$ is equivalent to minimizing the function $q(t_1, \ldots, t_n) = \sum_{i=1}^{n} (t_i + 1/2)^2 \|\mathbf{b}_i^\star\|^2$ over $\mathbb{N}^n$, which is the same as minimizing $\|\sum_{i=1}^{n} t_i \mathbf{b}_i^\star + \sum_{i=1}^{n} \mathbf{b}_i^\star/2\|^2$: the minimal value is $\sum_{i=1}^{n} \|\mathbf{b}_i^\star\|^2/4$ reached at 0.

Let $L^\star$ be the lattice spanned by the $\mathbf{b}_i^\star$'s, which are pairwise orthogonal. Then finding the M cells with the smallest expectation $\mathbb{E}\{\mathcal{C}_{\mathbb{N}}(\mathbf{t})\}$ is equivalent to finding the M lattice points $\sum_{i=1}^{n} t_i \mathbf{b}_i^\star \in L^\star$ with positive coefficients $t_i \in \mathbb{N}$ which are the closest to $\mathbf{u} = -\sum_{i=1}^{n} \mathbf{b}_i^\star/2$. To solve this problem, we solve a related problem: find all the lattice points $\sum_{i=1}^{n} t_i \mathbf{b}_i^\star \in L^\star$ whose distance to $\mathbf{u}$ is less than a given bound, with the restriction that all $t_i \in \mathbb{N}$. This is a special case of lattice enumeration in which the coefficients are positive and the input basis vectors are orthogonal: it can be done by slightly modifying enumeration, as shown by Algorithm 6. Given a bound $r > 0$ and $r_1 \leq r_2 \leq \cdots \leq r_n$, Algorithm 6 generates all non-zero $(x_1, \ldots, x_n) \in \mathbb{N}^n$ such that $\sum_{i=1}^{n} (x_i + 1/2)^2 r_i \leq r$. Algorithm 6 finds all the integer points $\in \mathbb{Z}^n$ which are inside some ellipsoid and the positive orthant (because each $x_i \geq 0$). We will explain later

why require that $r_1 \leq r_2 \leq \cdots \leq r_n$, but we note that this is actually not a constraint: if the r_i's are not increasing, sort them by finding a permutation π of $\{1, \ldots, n\}$ such that $r_{\pi(1)} \leq r_{\pi(2)} \leq \cdots \leq r_{\pi(n)}$, then call Algorithm 6 on $(r_{\pi(1)} \leq r_{\pi(2)} \leq \cdots \leq r_{\pi(n)}$ and r, and post-process any output $(x_1, \ldots, x_n)$ of Algorithm 6 by returning instead $(x_{\pi^{-1}(1)}, \ldots, x_{\pi^{-1}(n)})$. Thus, by choosing the r_i's as the $\|\mathbf{b}_i^\star\|^2$'s after suitable reordering, we can indeed use Algorithm 6 to find all the lattice points $\sum_{i=1}^n t_i \mathbf{b}_i^\star \in L^\star$ whose distance to $\mathbf{u}$ is less than a given bound, with the restriction that all $t_i \in \mathbb{N}$.

Now we claim that if we call Algorithm 6 with a suitable value of r we can indeed find the M cells with the smallest expectation $\mathbb{E}\{\mathcal{C}_{\mathbb{N}}(\mathbf{t})\}$ as follows:

- There are small intervals I such that for any $r \in I$, Algorithm 6 will output only slightly more than M solutions. Then we can sort all the solutions by expectation, and only output the best M tags.
- To find such an interval I slightly modify Algorithm 6 to obtain an algorithm that decides if the number of non-zero $(x_1, \ldots, x_n) \in \mathbb{N}^n$ such that $\sum_{i=1}^n (x_i + 1/2)^2 r_i \leq r$ is larger or less than a given number, with an early-abort as soon as it has found enough solutions. This immediately gives us a suitable I by binary search, using a logarithmic number of calls to the modified version of Algorithm 6. In practice, a non-optimized implementation typically takes a few seconds to find say the best 50 millions tags.

To make this approach practical, it is crucial that Algorithm 6 is very efficient. At first, this looks rather surprising: Algorithm 6 is doing enumeration with a lattice basis $(\mathbf{b}_1^\star, \ldots, \mathbf{b}_n^\star)$ whose Gram-Schmidt norms are identical to that of $(\mathbf{b}_1, \ldots, \mathbf{b}_n)$, with a non-trivial radius beyond the $\mathrm{GH}(L^\star)$. Even if we restrict to positive coefficients, it looks impossible, because the running time of enumeration is typically predicted well by the Gaussian heuristic (see [11]), using values that depend only the Gram-Schmidt norms: this means that, naively, we might expect enumeration in $L^\star$ to be as expensive as enumeration in L, in which case the whole approach would be meaningless, because the goal of discrete pruning is to speed up enumeration. Fortunately, it turns out that the usual predictions for the running time of enumeration do not apply to $L^\star$, because the basis $(\mathbf{b}_1^\star, \ldots, \mathbf{b}_n^\star)$ we use has a very special property: all its vectors are orthogonal, and it is known that in lattices generated by orthogonal vectors, the number of lattice points in a ball can behave significantly differently than usual (see [21]). Yet, that alone would not be sufficient in practice to guarantee efficiency: this is where the constraint $r_1 \leq r_2 \leq \cdots \leq r_n$ matters. In our experiments with that constraint, if ℓ is the number of solutions (*i.e.* the number of $(x_1, \ldots, x_n) \in \mathbb{N}^n$ such that $\sum_{i=1}^n (x_i + 1/2)^2 r_i \leq R$), the running time appears to be polynomial in ℓ. More precisely, like all enumeration algorithms, Algorithm 6 can be viewed as a depth-first search of a tree, and the running time is less than $O(L)$ polynomial-time operations, where L is the total number of nodes of the tree. In practice, at least in the context of discrete pruning, the number L seems to be bounded by $O(\ell \times n)$, and even $\ell \times n$. This is contrast in the usual situation for which the number of nodes of the enumeration tree is exponentially larger than the

Algorithm 6. Enumeration of cells of low expectations

Input: $(r_1, \ldots, r_n) \in \mathbb{R}^n$ such that $0 \leq r_1 \leq r_2 \leq \cdots \leq r_n$ and a bound $r > 0$.
Output: All $(x_1, \ldots, x_n) \in \mathbb{N}^n \setminus \{0\}$ such that $\sum_{i=1}^{n}(x_i + 1/2)^2 r_i \leq r$ and all $x_i \geq 0$.

$v_2 = \cdots = v_n = \rho_{n+1} = 0$ *// current coefficients*
for $k = n$ downto 2 **do**
  $\rho_k = \rho_{k+1} + (v_k + 1/2)^2 \cdot r_k$ *// partial squared norms*
end for
$k = v_1 = 1$;
while true **do**
  $\rho_k = \rho_{k+1} + (v_k + 1/2)^2 \cdot r_k$ *// compute squared norm of current node*
  if $\rho_k \leq r$ **then**
    if $k = 1$ **then**
      return $(v_1, \ldots, v_n)$; (*solution found*)
      $v_k \leftarrow v_k + 1$
    else
      $k \leftarrow k - 1$ *// going down the tree*
      $v_k \leftarrow 0$
    end if
  else
    $k \leftarrow k + 1$ *// going up the tree*
    if $k = n + 1$ **then**
      exit (*no more solutions*)
    else
      $v_k \leftarrow v_k + 1$
    end if
  end if
end while

number of solutions. We leave it as an open problem to show the efficiency of Algorithm 6.

6.3 Faster Approximation of the Success Probability by Statistical Inference

As seen in Sect. 4.2, estimating the success probability of discrete pruning requires from (1) the computation of $\sum_{\mathbf{t} \in U} \mathrm{vol}(S \cap \mathcal{C}(\mathbf{t}))$, where U is the set of tags. We saw in the previous section how to compute efficiently $\mathrm{vol}(S \cap \mathcal{C}(\mathbf{t}))$ for the natural partition by reducing to the case of $\mathrm{vol}(H \cap S)$ where H is box. Although this computation is efficient, it is not sufficiently efficient to be applied billions of times within a few seconds.

Fortunately, the classical theory of statistical inference allows us to approximate $\sum_{\mathbf{t} \in U} \mathrm{vol}(S \cap \mathcal{C}(\mathbf{t}))$ reasonably well without computing each term of the sum separately: it turns out that even a constant number of terms is sufficient to obtain a good approximation in practice, and a good approximation is sufficient for our heuristic estimate of the success probability of discrete pruning, since Heuristic 2 is only a heuristic estimate. To illustrate the presentation, we report experiments for a 70-dimensional LLL-reduced basis of a random lattice L, radius

$R = 1.2GH(L)$, and a typical set of tags, namely 5,000,000 tags generated by Algorithm 6. Using the FILT method, we find that the sum $\sum_{\mathbf{t}\in U} \mathrm{vol}(S \cap \mathcal{C}(\mathbf{t}))$ of 5,000,000 volumes is approximately $35.03688\mathrm{covol}(L)$ by computing each volume, and we will show how to approximate this value using only 1,000 volumes.

We want to approximate the sum $\mu = \sum_{\mathbf{t}\in U} \mathrm{vol}(S \cap \mathcal{C}(\mathbf{t}))$, which is exactly the expectation of the discrete random variable $X = \#U \times \mathrm{vol}(S \cap \mathcal{C}(\mathbf{t}))$, when $\mathbf{t}$ ranges over the finite set U.

Let $X_1, \ldots, X_m$ be random variables chosen independently from the same distribution as X. Then the sample mean $\overline{X} = (X_1 + \cdots + X_m)/m$. Its expectation $\mathbb{E}(\overline{X})$ is exactly the target $\mu = \sum_{\mathbf{t}\in U} \mathrm{vol}(S \cap \mathcal{C}(\mathbf{t}))$, and its variance $\mathbb{V}(\overline{X})$ is $\mathbb{V}(X)/m$. We want to know how close is $\overline{X}$ to its expectation μ.

Since the X_i's are discrete, the central limit theorem implies that $\sqrt{m}(\overline{X} - \mu)$ converges in distribution (as m grows) to the centered normal distribution of standard deviation σ: Fig. 10 shows the distribution of X and $\overline{X}$ (with $m = 1000$) for our example. This means that in practice, we can already expect to find a reasonable approximation of μ by simpling selecting m tags $\mathbf{t}_1, \ldots, \mathbf{t_m}$ independently and uniformly at random from the set U: the estimate would be $\frac{\#U}{m} \sum_{i=1}^{m} \mathrm{vol}(S \cap \mathcal{C}(\mathbf{t}_i))$, with an absolute error of magnitude $\pm\sigma/\sqrt{m}$.

The strategy we described is known as *simple sampling*, because it is the simplest method to estimate the expectation of X, but there are many other methods. In practice, we used a better yet still simple strategy, known as *stratified sampling*, because we can take advantage of properties of the cells we select in practice.

In the simplest form of stratified sampling, the set of tags U can be partitioned into subsets, and one selects a tag uniformly at random inside each subset and "extrapolate", as before. More precisely, if the subsets are $U_1, \ldots U_m$, then the (randomized) estimate is

$$\overline{X'} = \frac{1}{m} \sum_{i=1}^{m} \#U_i \times \mathrm{vol}(S \cap \mathcal{C}(\mathbf{t}_i)), \tag{13}$$

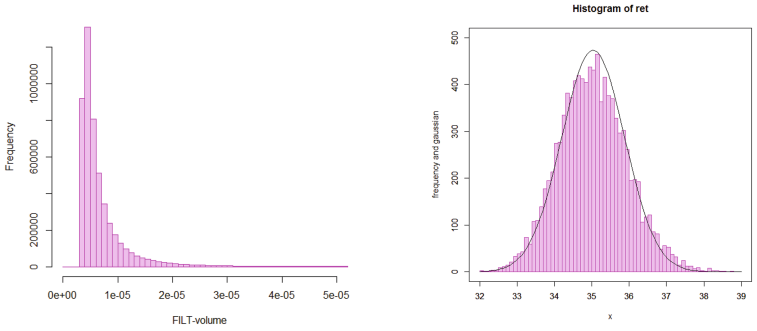

Fig. 10. Distribution of $X/(\#U\mathrm{covol}(L))$ (on the left) and the sample mean $\overline{X}$ (on the right) for $m = 1000$.

where each $\mathbf{t}_i$ is selected uniformly at random from U_i. In our case, assume that we select the set U of tags formed by the best M tags $\mathbf{t}$ with respect to $\mathbb{E}\{\mathcal{C}_{\mathbb{N}}(\mathbf{t})\}$ for some M. Then we sort all these tags by increasing expectation, and split the ordered sequence into m subsets $U_1, \ldots, U_m$ of nearly equal size. Thus, for all $(\mathbf{t}_i, \mathbf{t}_j) \in U_i \times U_j$ and $i < j$, we have: $\mathbb{E}\{\mathcal{C}_{\mathbb{N}}(\mathbf{t}_i)\} \leq \mathbb{E}\{\mathcal{C}_{\mathbb{N}}(\mathbf{t}_j)\}$.

Figure 11 shows that for the same value of m, the distribution of $\overline{X'}$ is narrower than that of $\overline{X}$, and therefore stratified sampling gives a better estimate than simple sampling. Figure 12 shows how accurate is (13), for increasing values of m, where there are 10,000 trials for each value of m. Using these sampling strategies from statistical inference, it is now possible to estimate rather precisely the sum of volumes efficiently in practice. A value of the order $m = 1000$ appears to be sufficient, even for much larger sets of tags. This is consistent with common practice in statistical surveys.

By combining this fast approximation algorithm with Sect. 6.2, we can efficiently find out what is the best trade-off for discrete pruning: by increasing order of magnitude of M, and starting with a small one, we use Sect. 6.2 to identify the

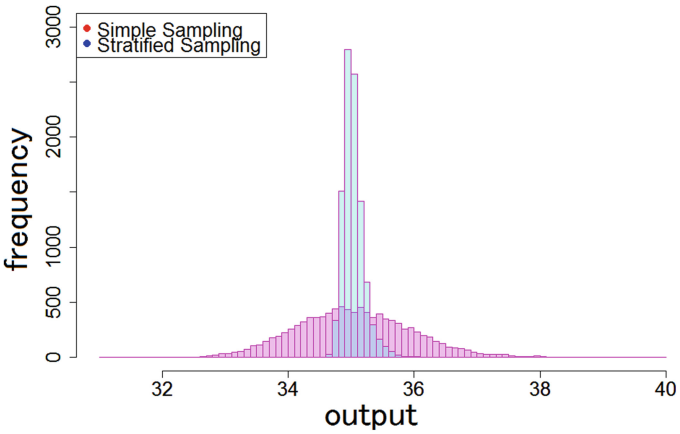

Fig. 11. Distributions of the estimate of simple sampling and stratified sampling.

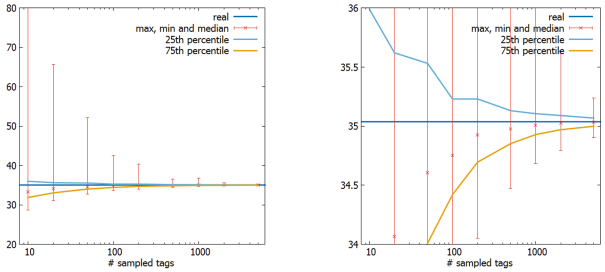

Fig. 12. Accuracy of stratified sampling with $m = 10, \ldots, 5000$. The right-hand side zooms in the region $y \in [34, 36]$.

nearly-best M cells, and apply the fast approximation algorithm to estimate the success probability for these M cells. We then deduce which M offers the best trade-off, after taking into account the time of basis reduction, like in continuous pruning [11]. However, this optimization is easier than for continuous pruning: the only parameter is M (or even its order of magnitude).

6.4 Comparison of Cells

As we mentioned in Sect. 4.3, Schnor's random sampling is essentially discrete pruning with the set of tags $U_u = \{(0, ..., 0, t_1, ..., t_u, 1) \in \{0, 1\}^n\}$. We have computed the intersection volume and the cell expectations of these tags. The blue curve in Fig. 9 shows the intersection volume sorted by cell expectation. We compared Schnorr's cells with the cells of Fukase-Kashiwabara [8], as selected by the process described in Sect. 4.3. The experiments show that the intersection volumes are much bigger for the FK cells than for Schnorr's cells: the Fukase-Kashiwabara variant [8] outperforms Schnorr's random sampling. We also computed the best cells in terms of expectation, using our algorithm: in this limited experiments, we can see in Fig. 9 that the FK tags are very close to our tags, but that tey are still different. For instance, if we consider a typical BKZ-20 basis of a 250-dimensional random lattice, then among the best 5×10^7 tags selected by Algorithm 6:

- about 70% of the tags have at least one coefficient $\notin \{0, 1\}$ and cannot therefore be selected by Schnorr's random sampling.
- about 25% of the tags have at least two coefficients $\notin \{0, 1\}$ and cannot therefore be selected by the method outlined in [8].

Accordingly, we obtained several experiments in which the shortest vector found by discrete pruning had a tag which did not mach the FK selection nor of course Schnorr's selection.

6.5 Bad Cells

To conclude this section, we note that discrete pruning can be slightly improved by removing what we call bad cells. For instance, the lattice point inside the cell $\mathcal{C}_{\mathbb{N}}(\mathbf{0})$ is zero, which is useless. When selecting a set of tags by lowest expectation, the set usually contains "a trivial" tag of the form of $\mathbf{e}_i = (0, \ldots, 0, 1, 0, \ldots, 0)$. When the basis B is size-reduced, it turns out that the lattice point inside the cell $\mathcal{C}_{\mathbb{N}}(\mathbf{t})$ is simply the trivial vector $\mathbf{b}_i$, which is usually not very short for usual reduction algorithms, and can anyways be tested separately. Although the cells of these tags have small expectation, they are not useful for discrete pruning: they can be removed by precomputation.

7 Experiments

Most of our experiments were performed by a standard server with two Intel Xeon E5-2660 CPUs and 256-GB RAMs. In this section, by random reduced

basis, we mean a basis obtained by reducing the Hermite normal form of lattices generated by the SVP challenge generator [28] with different seeds: each seed selects a different lattice, and we only consider one reduced basis per lattice for a given reduction strength. The LLL and BKZ reduced bases are computed by the NTL and progressive BKZ library [2], respectively.

Typical graphs of $\|\mathbf{b}_i^\star\|^2$ of LLL, PBKZ-20 and 60 are shown in Fig. 13.

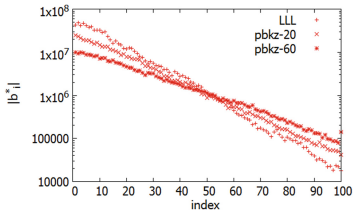

Fig. 13. Typical graphs of $\|\mathbf{b}_i^\star\|$ of LLL, PBKZ-20 and 60 reduced bases of a random 100-dimensional lattice.

For a lattice basis B, a set of tags $T = \{\mathbf{t}_1, \ldots, \mathbf{t}_M\}$, and bounding radius R, define the symbol

$$V(B, T, R) = \sum_{j=1}^{M} \frac{\mathrm{vol}(\mathcal{C}_{\mathbb{N}}(\mathbf{t_i}) \cap \mathrm{Ball}_n(R))}{\mathrm{covol}(L)}$$

which, by Heuristic 1, is a heuristic estimate of the number of lattice points contained in the union of cells.

7.1 Verifying Heuristics on the Success Probability and Number of Solutions

For a lattice basis and a set $\{\mathbf{t}_1, \ldots, \mathbf{t}_M\}$ of tags, consider the union of corresponding cells, that is, the pruning set: $P = \cup_{i=1}^{N} \mathcal{C}_{\mathbb{N}}(\mathbf{t_i})$. Heuristic 1 suggests that the number of lattice points inside P and shorter than R is roughly $\mathrm{vol}(P \cap \mathrm{Ball}_n(R))/\mathrm{covol}(L)$.

We verified this estimate by using random LLL-reduced bases in dimension 100. For each basis B_i, we generated 1,000 tags by Algorithm 6 and selected the best tags so that the total ratio $\mathrm{vol}(P \cap \mathrm{Ball}_n(R))/\mathrm{covol}(L)$ (with radius $R = 1.2GH(L)$) was larger than 0.001. About 300 or 400 tags are selected. For 17,386 bases and 6,281,800 tags generated as above, we display the result in Fig. 14. By the red curve the relation between accumulated volume, that is, for the set of first i tags $T_i = \{\mathbf{t}_1, \ldots, \mathbf{t}_i\}$, $\sum_{\mathbf{t}_i \Leftarrow L_j} V(L, T_i, R = 1.2GH(L_j))$. Here, the notation $\mathbf{t}_i \Leftarrow L_j$ means that $\mathbf{t}_i$ is generated by Algorithm 6 for the basis L_j. The blue curve is the number of non-zero vectors which are contained in the boxes of the tag set T_i.

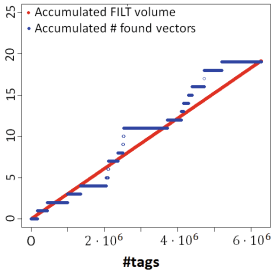

Fig. 14. Comparison between accumulated volume and actual number of found vectors

7.2 Comparison with Classical Pruned Enumeration

We give experiments comparing discrete pruning with continuous pruning [11]. The parameters are: the lattice dimension n, the number of tags M, the enumeration radius $R = \alpha \cdot \mathrm{GH}(L)$ and the blocksize β of BKZ lattice reduction.

The outline of the comparison is as follows: Generate an n-dimensional random lattice and reduce it by BKZ-β. Generate the M best tags $T = \{\mathbf{t}_1, \ldots, \mathbf{t}_M\}$ by Algorithm 6 and compute the intersection volume $V(L, T, R = \alpha \cdot \mathrm{GH}(L))$. For $M > 1000$, we use stratified sampling with $m = 1000$.

To compare with continuous pruning, we adapt Gama-Nguyen-Regev's optimization method to minimize the expected number of processed nodes

$$N = \frac{1}{2}\sum_{k=1}^{n} \frac{\mathrm{vol}\{(x_1, \ldots, x_k) \in \mathbb{R}^k : \sum_{i=1}^{\ell} x_i^2 < (Rf(\ell) \cdot \alpha GH(L))^2 \text{ for all } \ell \in [k]\}}{\prod_{i=n-k+1}^{n} \|\mathbf{b}_i^\star\|}.$$

subject to the volume of cylinder intersections

$$\frac{\mathrm{vol}\{(x_1, \ldots, x_n) \in \mathbb{R}^n : \sum_{i=1}^{\ell} x_i^2 < (R \times f(\ell) \cdot \alpha GH(L))^2 \text{ for } \forall\ \ell \in [n]\}}{\mathrm{covol}(L)}$$

is larger than $V(L, T, R)$ by optimizing the bounding coefficients $f(1), \ldots, f(n)$. Note that in the original paper [11], their condition is the probability defined by the surface area of a cylinder intersection.

The cost estimation for discrete pruning is easy: the number of operations is essentially $M \cdot n^2/2$ floating-point operations since one conversion defined in Algorithm 1 requires $n^2/2$ floating-point operations, like Babai's nearest plane algorithm. But the implementation can be massively parallelized (see [29] for the special case of Schnorr's random sampling). On the other hand, the cost estimation for continuous pruning is a bit more tricky, since the actual cost to process one node, i.e., cost to decide whether the node is alive or pruned, is proportional to the depth in the searching tree. To make a rigid comparison, we counted up the actual number N_i of nodes at depth i processed during the enumeration by experiments and we define the cost as $\sum_{i=1}^{n} i \cdot N_i$.

With these settings, we tried 20 random bases for the same parameter set $(n, M, R = \alpha \cdot \mathrm{GH}(L), \beta)$: Fig. 15 shows the average ratio of the costs

$$\frac{1}{20} \sum_{seed=0}^{19} \frac{M \cdot n^2/2}{\sum_{i=1}^{n} i \cdot N_i}$$

which indicates when discrete pruning is faster or slower than continuous pruning, depending on whether the ratio is ≤ 1 or ≥ 1.

The trends are clear from the figures. We find that the discrete pruning is faster than continuous pruning when:

1. The number M of tags is small. This might be useful in extreme enumeration for approximating the shortest vector problem.
2. The lattice dimension is high. Besides the speed factor, it might be useful because it is not easy to run continuous pruning with a very low success probability in high dimension, as it is harder to find suitable optimized bounding functions.
3. The lattice basis is not strongly BKZ-reduced.

On the other hand, the α parameter that sets the enumeration radius does not affect the trends.

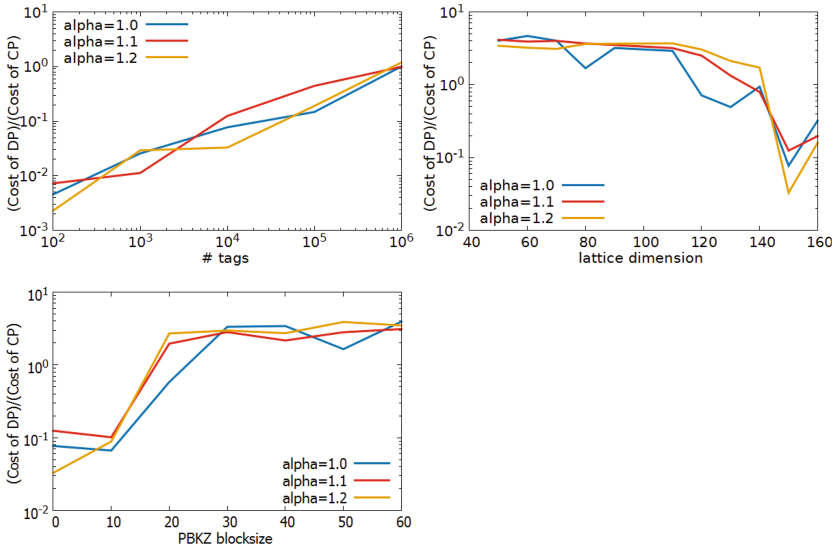

Fig. 15. Discrete pruning vs Continuous pruning: above the virtual horizontal line 10^0, discrete pruning is more expensive. (Upper left) 150-dim LLL-reduced bases with increasing α and M; (Upper right) LLL-reduced bases with increasing dimension and α. $M = 10^4$ is fixed; (Lower left) 150-dim with increasing blocksize β of progressive BKZ and α. $M = 10^4$ is fixed. Note that $\beta = 0$ corresponds to LLL.

Acknowledgements. This work was supported by JSPS KAKENHI Grant Numbers 16H02780 and 16H02830. We thank the reviewers for their comments.

References

1. Agrell, E., Eriksson, T., Vardy, A., Zeger, K.: Closest point search in lattices. IEEE Trans. Info. Theory **48**(8), 2201–2214 (2002)
2. Aono, Y., Wang, Y., Hayashi, T., Takagi, T.: Improved progressive BKZ algorithms and their precise cost estimation by sharp simulator. In: Fischlin, M., Coron, J.-S. (eds.) EUROCRYPT 2016. LNCS, vol. 9665, pp. 789–819. Springer, Heidelberg (2016). doi:10.1007/978-3-662-49890-3_30
3. Babai, L.: On Lovász' lattice reduction and the nearest lattice point problem. In: Mehlhorn, K. (ed.) STACS 1985. LNCS, vol. 182, pp. 13–20. Springer, Heidelberg (1985). doi:10.1007/BFb0023990
4. Buchmann, J., Ludwig, C.: Practical lattice basis sampling reduction. In: Hess, F., Pauli, S., Pohst, M. (eds.) ANTS 2006. LNCS, vol. 4076, pp. 222–237. Springer, Heidelberg (2006). doi:10.1007/11792086_17
5. Chen, Y., Nguyen, P.Q.: BKZ 2.0: better lattice security estimates. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 1–20. Springer, Heidelberg (2011). doi:10.1007/978-3-642-25385-0_1
6. Ding, D., Zhu, G., Wang, X.: A genetic algorithm for searching the shortest lattice vector of SVP challenge. In: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2015, pp. 823–830 (2015)
7. Fincke, U., Pohst, M.: Improved methods for calculating vectors of short length in a lattice, including a complexity analysis. Math. Comput. **44**(170), 463–471 (1985)
8. Fukase, M., Kashiwabara, K.: An accelerated algorithm for solving SVP based on statistical analysis. JIP **23**(1), 67–80 (2015)
9. Fukase, M., Yamaguchi, K.: Analysis of the extended search space for the shortest vector in lattice. J. Syst. Cybern. Inf. **9**(6), 42–46 (2011)
10. Gama, N., Nguyen, P.Q.: Predicting lattice reduction. In: Smart, N. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 31–51. Springer, Heidelberg (2008). doi:10.1007/978-3-540-78967-3_3
11. Gama, N., Nguyen, P.Q., Regev, O.: Lattice enumeration using extreme pruning. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 257–278. Springer, Heidelberg (2010). doi:10.1007/978-3-642-13190-5_13
12. Hanrot, G., Stehlé, D.: Improved analysis of kannan's shortest lattice vector algorithm. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 170–186. Springer, Heidelberg (2007). doi:10.1007/978-3-540-74143-5_10
13. Håstad, J., Just, B., Lagarias, J.C., Schnorr, C.-P.: Polynomial time algorithms for finding integer relations among real numbers. SIAM J. Comput. **18**(5), 859–881 (1989)
14. Hosono, T.: Numerical inversion of laplace transform and some applications to wave optics. Radio Sci. **16**(6), 1015–1019 (1981)
15. Kannan, R.: Improved algorithms for integer programming and related lattice problems. In: Proceedings of the 15th ACM Symposium on Theory of Computing (STOC), pp. 193–206 (1983)
16. Lenstra, A.K., Lenstra Jr., H.W., Lovász, L.: Factoring polynomials with rational coefficients. Math. Ann. **261**, 513–534 (1982)

17. Lindner, R., Peikert, C.: Better key sizes (and Attacks) for LWE-based encryption. In: Kiayias, A. (ed.) CT-RSA 2011. LNCS, vol. 6558, pp. 319–339. Springer, Heidelberg (2011). doi:10.1007/978-3-642-19074-2_21
18. Lindner, R., Rückert, M.: TU Darmstadt lattice challenge. http://www.latticechallenge.org/
19. Liu, M., Nguyen, P.Q.: Solving BDD by enumeration: an update. In: Dawson, E. (ed.) CT-RSA 2013. LNCS, vol. 7779, pp. 293–309. Springer, Heidelberg (2013). doi:10.1007/978-3-642-36095-4_19
20. Ludwig, C.: Practical Lattice Basis Sampling Reduction. Ph.D. thesis, Technische Universität Darmstadt (2005)
21. Mazo, J.E., Odlyzko, A.M.: Lattice points in high dimensional spheres. Monatsh. Math. **17**, 47–61 (1990)
22. Nguyen, P.Q.: Public-key cryptanalysis. In: Luengo, I. (ed.) Recent Trends in Cryptography. Contemporary Mathematics, vol. 477. AMS-RSME (2009)
23. Nguyen, P.Q., Stehlé, D.: LLL on the average. In: Hess, F., Pauli, S., Pohst, M. (eds.) ANTS 2006. LNCS, vol. 4076, pp. 238–256. Springer, Heidelberg (2006). doi:10.1007/11792086_18
24. Nguyen, P.Q., Vallée, B. (eds.): The LLL Algorithm: Survey and Applications. Information Security and Cryptography. Springer, Heidelberg (2009)
25. Pohst, M.: On the computation of lattice vectors of minimal length, successive minima and reduced bases with applications. SIGSAM Bull. **15**(1), 37–44 (1981)
26. Rogers, C.A.: The number of lattice points in a set. Proc. London Math. Soc. **6**(3), 305–320 (1956)
27. Rousseau, C.C., Ruehr, O.G.: Problems and solutions. SIAM Review **39**(4), 779–786 (1997). Subsection: The Volume of the Intersection of a Cube and a Ball in N-space. Two solutions by Bernd Tibken and Denis Constales
28. Schneider, M., Gama, N.: SVP challenge. http://www.latticechallenge.org/svp-challenge/
29. Schneider, M., Göttert, N.: Random sampling for short lattice vectors on graphics cards. In: Preneel, B., Takagi, T. (eds.) CHES 2011. LNCS, vol. 6917, pp. 160–175. Springer, Heidelberg (2011). doi:10.1007/978-3-642-23951-9_11
30. Schnorr, C.P.: Lattice reduction by random sampling and birthday methods. In: Alt, H., Habib, M. (eds.) STACS 2003. LNCS, vol. 2607, pp. 145–156. Springer, Heidelberg (2003). doi:10.1007/3-540-36494-3_14
31. Schnorr, C.-P., Euchner, M.: Lattice basis reduction: improved practical algorithms and solving subset sum problems. Math. Program. **66**, 181–199 (1994)
32. Schnorr, C.P., Hörner, H.H.: Attacking the Chor-Rivest cryptosystem by improved lattice reduction. In: Guillou, L.C., Quisquater, J.-J. (eds.) EUROCRYPT 1995. LNCS, vol. 921, pp. 1–12. Springer, Heidelberg (1995). doi:10.1007/3-540-49264-X_1
33. Innovation, S.: NTRU challenge. https://www.securityinnovation.com/products/ntru-crypto/ntru-challenge
34. Siegel, C.L.: A mean value theorem in geometry of numbers. Ann. Math. **46**(2), 340–347 (1945)
35. Stehlé, D., Watkins, M.: On the extremality of an 80-dimensional lattice. In: Hanrot, G., Morain, F., Thomé, E. (eds.) ANTS 2010. LNCS, vol. 6197, pp. 340–356. Springer, Heidelberg (2010). doi:10.1007/978-3-642-14518-6_27