A $2^{n/2}$-Time Algorithm for $\sqrt{n}$-SVP and $\sqrt{n}$-Hermite SVP, and an Improved Time-Approximation Tradeoff for (H)SVP

Abstract. We show a $2^{n/2+o(n)}$-time algorithm that, given as input a basis of a lattice $\mathcal{L} \subset \mathbb{R}^n$, finds a (non-zero) vector in whose length is at most $\widetilde{O}(\sqrt{n}) \cdot \min\{\lambda_1(\mathcal{L}), \det(\mathcal{L})^{1/n}\}$, where $\lambda_1(\mathcal{L})$ is the length of a shortest non-zero lattice vector and $\det(\mathcal{L})$ is the lattice determinant. Minkowski showed that $\lambda_1(\mathcal{L}) \leq \sqrt{n}\det(\mathcal{L})^{1/n}$ and that there exist lattices with $\lambda_1(\mathcal{L}) \geq \Omega(\sqrt{n}) \cdot \det(\mathcal{L})^{1/n}$, so that our algorithm finds vectors that are as short as possible relative to the determinant (up to a poly-logarithmic factor).

The main technical contribution behind this result is new analysis of (a simpler variant of) a $2^{n/2+o(n)}$-time algorithm from [ADRS15], which was only previously known to solve less useful problems. To achieve this, we rely crucially on the "reverse Minkowski theorem" (conjectured by Dadush [DR16] and proven by [RS17]), which can be thought of as a partial converse to the fact that $\lambda_1(\mathcal{L}) \leq \sqrt{n}\det(\mathcal{L})^{1/n}$.

Previously, the fastest known algorithm for finding such a vector was the $2^{.802n+o(n)}$-time algorithm due to [LWXZ11], which actually found a non-zero lattice vector with length $O(1) \cdot \lambda_1(\mathcal{L})$. Though we do not show how to find lattice vectors with this length in time $2^{n/2+o(n)}$, we do show that our algorithm suffices for the most important application of such algorithms: basis reduction. In particular, we show a modified version of Gama and Nguyen's slide-reduction algorithm [GN08], which can be combined with the algorithm above to improve the time-length tradeoff for shortest-vector algorithms in nearly all regimes—including the regimes relevant to cryptography.

1 Introduction

A lattice $\mathcal{L} \subset \mathbb{R}^n$ is the set of integer linear combinations

$$\mathcal{L} := \mathcal{L}(\mathbf{B}) = \{z_1\mathbf{b}_1 + \cdots + z_n\mathbf{b}_n \ : \ z_i \in \mathbb{Z}\}$$

of linearly independent basis vectors $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n) \in \mathbb{R}^{n \times n}$. We define the length of a shortest non-zero vector in the lattice as $\lambda_1(\mathcal{L}) := \min_{\mathbf{x} \in \mathcal{L}_{\neq \mathbf{0}}} \|\mathbf{x}\|$. (Throughout this paper, $\|\cdot\|$ is the Euclidean norm.)

The Shortest Vector Problem (SVP) is the computational search problem whose input is a (basis for a) lattice $\mathcal{L} \subseteq \mathbb{R}^n$, and the goal is to output a shortest non-zero vector $\mathbf{y} \in \mathcal{L}$ with $\|\mathbf{y}\| = \lambda_1(\mathcal{L})$. For $\delta \geq 1$, the δ-approximate variant of SVP (δ-SVP) is the problem of finding a non-zero vector $\mathbf{y} \in \mathcal{L}$ of length at most $\delta \cdot \lambda_1(\mathcal{L})$ given a basis of $\mathcal{L}$.

δ-SVP and its many relatives have found innumerable applications over the past forty years. More recently, many cryptographic constructions have been discovered whose security is based on the (worst-case) hardness of δ-SVP or closely related lattice problems. See [Pei16] for a survey. Such lattice-based cryptographic constructions are likely to be used in practice on massive scales (e.g., as part of the TLS protocol) in the not-too-distant future [NIS18], and it is therefore crucial that we understand this problem as well as we can.

For most applications, it suffices to solve δ-SVP for superconstant approximation factors. E.g., cryptanalysis typically requires $\delta = \text{poly}(n)$. However, our best algorithms for δ-SVP work via (non-trivial) reductions to δ'-SVP for much smaller δ' *over lattices with smaller rank*, typically $\delta' = 1$ or $\delta' = O(1)$. E.g., one can reduce n^c-SVP with rank n to $O(1)$-SVP with rank $n/(c+1)$ for constant $c \geq 1$ [GN08, ALNS20]. Such reductions are called *basis reduction algorithms* [LLL82, Sch87, SE94].

Therefore, even if one is only interested in δ-approximate SVP for large approximation factors, algorithms for $O(1)$-SVP are still relevant. (We make little distinction between exact SVP and $O(1)$-SVP in the introduction. Indeed, many of the algorithm that we call $O(1)$-SVP algorithms actually solve exact SVP.)

1.1 Sieving for constant-factor-approximate SVP

There is a very long line of work (e.g., [Kan83, AKS01, NV08, PS09, MV13, LWXZ11, WLW15, ADRS15, AS18, AUV19]) on this problem.

The fastest known algorithms for $O(1)$-SVP run in time $2^{O(n)}$. With one exception ([MV13]), all known algorithms with this running time are variants of *sieving algorithms*. These algorithms work by sampling $2^{O(n)}$ not-too-long lattice vectors $\mathbf{y}_1, \ldots, \mathbf{y}_M \in \mathcal{L}$ from some nice distribution over the input lattice $\mathcal{L}$, and performing some kind of sieving procedure to obtain $2^{O(n)}$ shorter vectors $\mathbf{x}_1, \ldots, \mathbf{x}_m \in \mathcal{L}$. They then perform the sieving procedure again on the $\mathbf{x}_k$, and repeat this process many times.

The most natural sieving procedure was originally studied by Ajtai, Kumar, and Sivakumar [AKS01]. This procedure simply takes $\mathbf{x}_k := \mathbf{y}_i - \mathbf{y}_j \in \mathcal{L}$, where i, j are chosen so that $\|\mathbf{y}_i - \mathbf{y}_j\| \leq (1-\varepsilon) \min_\ell \|\mathbf{y}_\ell\|$. In particular, the resulting sieving algorithm clearly finds progressively shorter lattice vectors at each step. So, it is trivial to show that this algorithm will eventually find a short lattice vector. Unfortunately (and maddeningly), it seems very difficult to say nearly anything else about the distribution of the vectors when this very simple sieving technique is used, and in particular, while we know that the vectors must be short, we do not know how to show that they are *non-zero*. [AKS01] used clever tricks to modify the above procedure into one for which they could prove correctness, and the current state-of-the-art is a $2^{0.802n}$-time algorithm for γ-SVP for a sufficiently large constant $\gamma > 1$ [LWXZ11, WLW15, AUV19].

Another line of research [NV08, Laa15, MW16, BDGL16, Duc18] focuses on improving the time complexity of practical SVP algorithms by introducing various experimentally verified heuristics. These heuristic algorithms are thus more directly relevant for cryptanalysis. The fastest known heuristic algorithm

for solving SVP has time complexity $(3/2)^{(n/2)+o(n)}$, illustrating a large gap between provably correct and heuristic algorithms. (In this regard, this work contributes to the ultimate goal of closing this gap.)

In this work, we are more interested in the "sieving by averages" technique, introduced in [ADRS15] to obtain a $2^{n+o(n)}$-time algorithm for exact SVP. This sieving procedure takes $\mathbf{x}_k := (\mathbf{y}_i + \mathbf{y}_j)/2$ to be the average of two lattice vectors. Of course, $\mathcal{L}$ is not closed under taking averages, so one must choose i, j so that that $(\mathbf{y}_i + \mathbf{y}_j)/2 \in \mathcal{L}$. This happens if and only if $\mathbf{y}_i, \mathbf{y}_j$ lie in the same *coset* of $2\mathcal{L}$, $\mathbf{y}_i = \mathbf{y}_j \bmod 2\mathcal{L}$. Equivalently, the coordinates of $\mathbf{y}_i$ and $\mathbf{y}_j$ in the input basis should have the same parities. So, these algorithms pair vectors according to their cosets (and ignore all other information about the vectors) and take their averages $\mathbf{x}_k = (\mathbf{y}_i + \mathbf{y}_j)/2$.

The analysis of these algorithms centers around the *discrete Gaussian distribution* $D_{\mathcal{L},s}$ over a lattice, given by

$$\Pr_{\mathbf{X} \sim D_{\mathcal{L},s}}[\mathbf{X} = \mathbf{y}] \propto e^{-\pi \|\mathbf{y}\|^2 / s^2}$$

for a parameter $s > 0$ and any $\mathbf{y} \in \mathcal{L}$. When the starting vectors come from this distribution, we are able to say quite a bit about the distribution of the vectors at each step. (Intuitively, this is because this algorithm only uses algebraic properties of the vectors—their cosets—and entirely ignores the geometry.) In particular, [ADRS15] used a careful rejection sampling procedure to guarantee that the vectors at each step are distributed exactly as $D_{\mathcal{L},s}$ for some parameter $s > 0$. Specifically, in each step the parameter lowers by a factor of $\sqrt{2}$, which is exactly what one would expect, taking intuition from the continuous Gaussian. More closely related to this work is [AS18], which showed that this rejection sampling procedure is actually unnecessary.

In addition to the above, [ADRS15, Ste17] also present a $2^{n/2+o(n)}$-time algorithm that samples from $D_{\mathcal{L},s}$ as long as the parameter $s > 0$ is not too small. In particular, we need s to be "large enough that $D_{\mathcal{L},s}$ looks like a continuous Gaussian." This algorithm is similar to the $2^{n+o(n)}$-time algorithms in that it starts with independent discrete Gaussian vectors with some high parameter, and it gradually lowers the parameter using a rejection sampling procedure together with a procedure that takes the averages of pairs of vectors that lie in the same coset modulo some sublattice (with index $2^{n/2+o(n)}$). But, it fails for smaller parameters because the rejection sampling procedure that it uses must throw out too many vectors in this case. (In [Ste17], a different rejection sampling procedure is used that never throws away too many vectors, but it is not clear how to implement it in $2^{n/2+o(n)}$ time for small parameters $s < \sqrt{2}\eta_{1/2}(\mathcal{L})$.) It was left as an open question whether there is a suitable variant of this algorithm that works for small parameters, which would lead to an algorithm to solve SVP in $2^{n/2+o(n)}$ time. For example, perhaps we could show that the simple algorithm that solves SVP without doing any rejection sampling at all (similar to what was shown for the $2^{n+o(n)}$-time algorithm in [AS18]).

1.2 Hermite SVP

We will also be interested in a variant of SVP called Hermite SVP (HSVP). HSVP is defined in terms of the determinant $\det(\mathcal{L}) := |\det(\mathbf{B})|$ of a lattice $\mathcal{L}$ with basis $\mathbf{B}$. (Though a lattice can have many bases, one can check that $|\det(\mathbf{B})|$ is the same for all such bases, so that this quantity is well-defined.) Minkowski's celebrated theorem says that $\lambda_1(\mathcal{L}) \leq O(\sqrt{n}) \cdot \det(\mathcal{L})^{1/n}$, and Hermite's constant $\gamma_n = \Theta(n)$ is the maximal value of $\lambda_1(\mathcal{L})^2 / \det(\mathcal{L})^{2/n}$. (Hermite SVP is of course named in honor of Hermite and his study of γ_n. It is often alternatively called Minkowski SVP.)

For $\delta \geq 1$, it is then natural to define δ-HSVP as the variant of SVP that asks for any non-zero lattice vector $\mathbf{x} \in \mathcal{L}$ such that $\|\mathbf{x}\| \leq \delta \det(\mathcal{L})^{1/n}$. One typically takes $\delta \geq \sqrt{\gamma_n} \geq \Omega(\sqrt{n})$, in which case the problem is total. In particular, there is a trivial reduction from $\delta\sqrt{\gamma_n}$-HSVP to δ-SVP. (There is also a non-trivial reduction from δ^2-SVP to δ-HSVP for $\delta \geq \sqrt{\gamma_n}$ [Lov86].)

δ-HSVP is an important problem in its own right. In particular, the random lattices most often used in cryptography typically satisfy $\lambda_1(\mathcal{L}) \geq \Omega(\sqrt{n}) \cdot \det(\mathcal{L})^{1/n}$, so that for these lattices δ-HSVP is equivalent to $O(\delta/\sqrt{n})$-SVP. This fact is quite useful as the best known basis reduction algorithms [GN08, MW16, ALNS20] yield solutions to both δ_S-SVP and δ_H-HSVP with, e.g.,

$$\delta_H := \gamma_k^{\frac{n-1}{2(k-1)}} \approx k^{n/(2k)} \qquad \delta_S := \gamma_k^{\frac{n-k}{k-1}} \approx k^{n/k-1}\,, \tag{1}$$

when given access to an oracle for (exact) SVP in dimension $k \leq n/2$. Notice that δ_H is significantly better than the approximation factor $\sqrt{\gamma_n}\delta_S \approx \sqrt{n}k^{n/k-1}$ that one obtains from the trivial reduction to δ_S-SVP. (Furthermore, the approximation factor δ_H in Eq. (1) is achieved even for $n/2 < k \leq n$.)

In fact, it is easy to check that we will achieve the same value of δ_H if the reduction is instantiated with a $\sqrt{\gamma_k}$-HSVP oracle in dimension k, rather than an SVP oracle. More surprisingly, a careful reading of the proofs in [GN08, ALNS20] shows that a $\sqrt{\gamma_k}$-HSVP oracle is "almost sufficient" to even solve δ_S-SVP. (We make this statement a bit more precise below.)

1.3 Our results

Our main contribution is a simplified version of the $2^{n/2+o(n)}$-time algorithm from [ADRS15] and a novel analysis of the algorithm that gives an approximation algorithm for both SVP and HSVP.

Theorem 1.1 (Informal, approximation algorithm for (H)SVP)**.** *There is a $2^{n/2+o(n)}$-time algorithm that solves δ-SVP and δ-HSVP for $\delta \leq \widetilde{O}(\sqrt{n})$.*

Notice that this algorithm almost achieves the best possible approximation factor δ for HSVP since there exists a family of lattices for which $\lambda_1(\mathcal{L}) \geq \Omega(\sqrt{n}\det(\mathcal{L})^{1/n})$ (i.e., $\gamma_n \geq \Omega(n)$). So, δ is optimal for HSVP up to a polylogarithmic factor.

Problem	Approximation factor	Previous Best	This work
SVP	Exact	2^n [*] [ADRS15]	—
	$O(1)$	$2^{0.802n}$ [*] [WLW15]	—
	n^c for $c \in (0.5, 0.802]$	$2^{\frac{0.401n}{c}}$ [ALNS20]	$2^{\frac{n}{2}}$ [*]
	n^c for $c \in (0.802, 1]$	$2^{\frac{0.401n}{c}}$ [ALNS20]	—
	n^c for $c > 1$	$2^{\frac{0.802n}{c+1}}$ [ALNS20]	$2^{\frac{n}{2c+1.24}}$
HSVP	$\sqrt{n}$	$2^{0.802n}$ [*] [WLW15]	$2^{\frac{n}{2}}$ [*]
	n^c for $c \geq 1$	$2^{\frac{0.401n}{c}}$ [ALNS20]	$2^{\frac{n}{4c}}$

Table 1. Proven running times for solving (H)SVP. We mark results that do not use basis reduction with [*]. We omit $2^{o(n)}$ factors in the running time, and except in the first two rows, polylogarithmic factors in the approximation factor.

As far as we know, this algorithm might actually solve exact or near-exact SVP, but we do not know how to prove this. However, by adapting the basis reduction algorithms of [GN08, ALNS20], we show that Theorem 1.1 is nearly as good (when combined with known results) as a $2^{k/2}$-time algorithm for exact SVP in k dimensions, in the sense that we can already nearly match Eq. (1) in time $2^{k/2+o(k)}$ with this.

In slightly more detail, basis reduction procedures break the input basis vectors $\mathbf{b}_1, \ldots, \mathbf{b}_n$ into blocks $\mathbf{b}_{i+1}, \ldots, \mathbf{b}_{i+k}$ of length k. They repeatedly call their oracle on (projections of) the lattices generated by these blocks and use the result to update the basis vectors. We observe that the procedures in [GN08, ALNS20] only need to use an SVP oracle on the last block $\mathbf{b}_{n-k+1}, \ldots, \mathbf{b}_n$. For all other blocks, an HSVP oracle suffices. Since we now have a faster algorithm for HSVP than we do for SVP, we make this last block a bit smaller than the others, so that we can solve (near-exact) SVP on the last block in time $2^{k/2+o(k)}$.

When we apply the $2^{0.802n}$-time algorithm for $O(1)$-SVP from [LWXZ11, WLW15, AUV19] to instantiate this idea, it yields the following result, which gives the fastest known algorithm for δ-SVP for all $\delta \gtrsim n^c$.

Theorem 1.2 (Informal)**.** *There is a* $(2^{k/2+o(k)} \cdot \mathrm{poly}(n))$*-time algorithm that solves* δ_H^**-HSVP with*

$$\delta_H^* \approx k^{n/(2k)} ,$$

for $k \leq .99n$ *and*

$$\delta_S^* \approx k^{(n/k)-0.62} ,$$

for $k \leq n/1.63$*.*

Notice that Theorem 1.2 matches Eq. (1) with block size k exactly for δ_H, and up to a factor of $k^{0.37}$ for δ_S. This small loss in approximation factor comes from the fact that our last block is slightly smaller than the other blocks.

Together, Theorems 1.1 and 1.2 give the fastest proven running times for n^c-HSVP for all $c > 1/2$ and for n^c-SVP for all $c > 1$, as well as $c \in (1/2, 0.802)$. Table 1 summarizes the current state of the art.

1.4 Our techniques

Summing vectors over a tower of lattices Like the $2^{n/2+o(n)}$-time algorithm in [ADRS15], our algorithm for $\widetilde{O}(\sqrt{n})$-(H)SVP constructs a tower of lattices $\mathcal{L}_0 \supset \mathcal{L}_1 \supset \cdots \supset \mathcal{L}_\ell = \mathcal{L}$ such that for every $i \geq 1$, $2\mathcal{L}_{i-1} \subset \mathcal{L}_i$. The idea of using a tower of lattices was independently developed in [BGJ14] (see also [GINX16]) for heuristic algorithms. The index of $\mathcal{L}_i$ over $\mathcal{L}_{i-1}$ is 2^α for an integer $\alpha = n/2 + o(n)$, and $\ell = o(n)$. For the purpose of illustrating our ideas, we make a simplifying assumption here that $\ell\alpha$ is an integer multiple of n, and hence $\mathcal{L}_0 = \mathcal{L}/2^{\alpha\ell/n}$ is a scalar multiple of $\mathcal{L}$.

And, as in [ADRS15], we start by sampling $\mathbf{X}_1, \ldots, \mathbf{X}_N \in \mathcal{L}_0$ for $N = 2^{\alpha+o(n)}$ from $D_{\mathcal{L}_0,s}$. This can be done efficiently using known techniques, as long as s is large relative to, e.g., the length of the shortest basis of $\mathcal{L}_0$ [GPV08, BLP+13]. Since $\mathcal{L}_0 = \mathcal{L}/2^{\alpha\ell/n}$, the parameter s can still be significantly smaller than, e.g., $\lambda_1(\mathcal{L})$. In particular, we can essentially take $s \leq \mathrm{poly}(n)\lambda_1(\mathcal{L})/2^{\alpha\ell/n}$.

The algorithm then takes disjoint pairs of vectors that are in the same coset of $\mathcal{L}_0/\mathcal{L}_1$, and adds the pairs together. Since $2\mathcal{L}_0 \subset \mathcal{L}_1$, for any such pair $\mathbf{X}_i, \mathbf{X}_i$, $\mathbf{Y}_k = \mathbf{X}_i + \mathbf{X}_j$ is in $\mathcal{L}_1$. (This adding is analogous to the averaging procedure from [ADRS15, AS18] described above. In that case, $\mathcal{L}_1 = 2\mathcal{L}_0$, so that it is natural to divide vectors in $\mathcal{L}$ by two, while here adding seems more natural.) We thus obtain approximately $N/2$ vectors in $\mathcal{L}_1$ (up to the loss due to the vectors that could not be paired), and repeat this procedure many times, until finally we obtain vectors in $\mathcal{L}_\ell = \mathcal{L}$, each the sum of 2^ℓ of the original $\mathbf{X}_i$.

To prove correctness, we need to prove that with high probability some of these vectors will be *both* short and non-zero. It is actually relatively easy to show that the vectors are short—at least in expectation. To prove this, we first use the fact that the expected squared norm of the $\mathbf{X}_i$ is bounded by ns^2 (which is what one would expect from the continuous Gaussian distribution). And, the original $\mathbf{X}_i$ are distributed symmetrically, i.e., $\mathbf{X}_i$ is as likely to equal $-\mathbf{x}$ as it is to equal $\mathbf{x}$).

Furthermore, our pairing procedure is symmetric, i.e., if we were to replace $\mathbf{X}_i$ with $-\mathbf{X}_i$, the pairing procedure would behave identically. (This is true precisely because $2\mathcal{L}_0 \subset \mathcal{L}_1$—we are using the fact that $\mathbf{x} = -\mathbf{x} \bmod \mathcal{L}_1$ for any $\mathbf{x} \in \mathcal{L}_0$.) This implies that

$$\mathbb{E}[\langle \mathbf{X}_i, \mathbf{X}_j \rangle \mid E_{i,j}] = \mathbb{E}[\langle \mathbf{X}_i, -\mathbf{X}_j \rangle \mid E_{i,j}] = 0 \,,$$

where $E_{i,j}$ is the event that $\mathbf{X}_i$ is paired with $\mathbf{X}_j$. Therefore, $\mathbb{E}[\|\mathbf{X}_i + \mathbf{X}_j\|^2 \mid E_{i,j}]$ is equal to

$$\mathbb{E}[\|\mathbf{X}_i\|^2 \mid E_{i,j}] + \mathbb{E}[\|\mathbf{X}_j\|^2 \mid E_{i,j}] + 2\,\mathbb{E}[\langle \mathbf{X}_i, \mathbf{X}_j \rangle \mid E_{i,j}] \approx 2\,\mathbb{E}[\|\mathbf{X}_i\|^2] \,.$$

The same argument works at every step of the algorithm. So, (if we ignore the subtle distinction between $\mathbb{E}[\|\mathbf{X}_i\|^2 \mid E_{i,j}]$ and $\mathbb{E}[\|\mathbf{X}_i\|^2]$), we see that our final vectors have expected squared norm

$$2^\ell\,\mathbb{E}[\|\mathbf{X}_i\|^2] \leq 2^\ell n s^2 \leq \mathrm{poly}(n) 2^{\ell(1-2\alpha n)} \cdot \lambda_1(\mathcal{L})^2 \,. \tag{2}$$

By taking, e.g., $\alpha = n/2 + n/\log n < n + o(n)$ and $\ell = \log^2 n$, we see that we can make this expectation small relative to $\lambda_1(\mathcal{L})$.

The difficulty, then, is "only" to show that the distribution of the final vectors is not heavily concentrated on zero. Of course, we can't hope for this to be true if, e.g., the expectation in Eq. (2) is much smaller than $\lambda_1(\mathcal{L})^2$. And, as we will discuss below, if we choose α and ℓ so that this expectation is sufficiently large, then techniques from prior work can show that the probability of zero is low. Our challenge is therefore to bound the probability of zero for the largest choices of α and ℓ (and therefore the lowest expectation in Eq. (2)) that we can manage.

Gaussians over unknown sublattices Peikert and Micciancio (building on prior work) showed what they called a "convolution theorem" for discrete Gaussians. Their theorem said that the sum of discrete Gaussian vectors is statistically close to a discrete Gaussian (with parameter increased by a factor of $\sqrt{2}$), provided that the parameter s is a bit larger than the *smoothing parameter* $\eta(\mathcal{L})$ of the lattice $\mathcal{L}$ [MP13]. This (extremely important) parameter $\eta(\mathcal{L})$, was introduced by Micciancio and Regev [MR07], and has a rather technical (and elegant) definition. (See Section 2.4.) Intuitively, $\eta(\mathcal{L})$ is such that for any $s > \eta(\mathcal{L})$, $D_{\mathcal{L},s}$ "looks like a continuous Gaussian distribution." E.g., for $s > \eta(\mathcal{L})$, the moments of the discrete Gaussian distribution are quite close to the moments of the continuous Gaussian distribution (with the same parameter).

In fact, [MP13] showed a convolution theorem for lattice *cosets*, not just lattices, i.e., the sum of a vector sampled from coset $D_{\mathcal{L}+\mathbf{t}_1,s}$ and a vector sampled from $D_{\mathcal{L}+\mathbf{t}_2,s}$ yields a vector with a distribution that is statistically close to $D_{\mathcal{L}+\mathbf{t}_1+\mathbf{t}_2,\sqrt{2}s}$. Since our algorithm sums vectors sampled from a discrete Gaussian over $\mathcal{L}_0$, conditioned on their cosets modulo $\mathcal{L}_1$, it is effectively summing discrete Gaussians over cosets of $\mathcal{L}_1$. So, as long as we stay above the smoothing parameter of $\mathcal{L}_1 \supset \mathcal{L}$, our vectors will be statistically close to discrete Gaussians, allowing us to easily bound the probability of zero.

However, [ADRS15] already showed how to use a variant of this algorithm to obtain samples from *exactly* the discrete Gaussian above smoothing. And, more generally, there is a long line of work that uses samples from the discrete Gaussian above smoothing to find "short vectors" from a lattice, but the length of these short vectors is always proportional to $\eta(\mathcal{L})$. The problem is that in general $\eta(\mathcal{L})$ can be arbitrarily larger than $\lambda_1(\mathcal{L})$ and $\det(\mathcal{L})^{1/n}$. (To see this, consider the two-dimensional lattice generated by $(T, 0), (0, 1/T)$ for large T, which has $\eta(\mathcal{L}) \approx T$, $\lambda_1(\mathcal{L}) = 1/T$ and $\det(\mathcal{L}) = 1$.) So, this seems useless for solving (H)SVP, instead yielding a solution to another variant of SVP called SIVP.[1]

[1] It is not known how to use an SIVP oracle for basis reduction, which makes it significantly less useful than SVP. [MR07, MP13] and other works used these ideas to reduce SIVP to the problem of breaking a certain cryptosystem, in order to argue that the cryptosystem is secure. They were therefore primarily interested in SIVP as an example of a hard lattice problem, rather than as a problem that one might actually wish to solve.

Our solution is essentially to apply these ideas from [MP13] to an *unknown* sublattice $\mathcal{L}' \subseteq \mathcal{L}$. (Here, one should imagine a sublattice generated by fewer than n vectors. Jumping ahead a bit, the reader might consider the example $\mathcal{L}' = \mathbb{Z}\mathbf{v} = \{\mathbf{0}, \pm\mathbf{v}, \pm 2\mathbf{v}, \dots, \}$ the rank-one sublattice generated by $\mathbf{v}$, shortest non-zero vector in the lattice.) Indeed, the discrete Gaussian over $\mathcal{L}$, $D_{\mathcal{L},s}$, can be viewed as a *mixture of discrete Gaussians* over $\mathcal{L}'$, $D_{\mathcal{L},s} = D_{\mathcal{L}'+\mathbf{C},s}$, where $\mathbf{C} \in \mathcal{L}/\mathcal{L}'$ is some random variable over cosets of $\mathcal{L}'$. (Put another way, one could obtain a sample from $D_{\mathcal{L},s}$ by first sampling a coset $\mathbf{C} \in \mathcal{L}/\mathcal{L}'$ from some appropriately chosen distribution and then sampling from $D_{\mathcal{L}'+\mathbf{C},s}$.)

The basic observation behind our analysis is that we can now apply (a suitable variant of) [MP13]'s convolution theorem in order to see that the sum of two mixtures of Gaussians over $\mathcal{L}'$, $\mathbf{X}_1, \mathbf{X}_2 \sim D_{\mathcal{L}'+\mathbf{C},s}$, yields a new mixture of Gaussians $D_{\mathcal{L}'+\mathbf{C}',\sqrt{2}s}$ for *some* $\mathbf{C}'$, provided that s is sufficiently large *relative to* $\eta(\mathcal{L}')$.

Ignoring *many* technical details, this shows that our algorithm can be used to output a distribution of the form $D_{\mathcal{L}'+\mathbf{C},s}$ for some random variable $\mathbf{C} \in \mathcal{L}/\mathcal{L}'$ provided that $s \gg \eta(\mathcal{L}')$. Crucially, we only need to consider $\mathcal{L}'$ in the analysis; the algorithm does not need to know what $\mathcal{L}'$ is for this to work. Furthermore, we do not care at all about the distribution of $\mathbf{C}$! We already know that our algorithm samples from a distribution that is short in expectation (by the argument above), so that the only thing we need from the distribution $D_{\mathcal{L}'+\mathbf{C},s}$ is that it is not zero too often. Indeed, when $\mathbf{C}$ is not the zero coset (i.e., $\mathbf{C} \notin \mathcal{L}'$), then $D_{\mathcal{L}'+\mathbf{C},s}$ is never zero, and when $\mathbf{C}$ is zero, then we get a sample from $D_{\mathcal{L}',s}$ for $s \gg \eta(\mathcal{L})$, in which case well-known techniques imply that we are unlikely to get zero.

Smooth sublattices So, in order to prove that our algorithm finds short vectors, it remains to show that there exists some sublattice $\mathcal{L}' \subseteq \mathcal{L}$ with low smoothing parameter—a "smooth sublattice." In more detail, our algorithm will find a non-zero vector with length less than $\sqrt{n} \cdot \eta(\mathcal{L}')$ for any sublattice $\mathcal{L}'$. Indeed, as one might guess, taking $\mathcal{L}' = \mathbb{Z}\mathbf{v} = \{\mathbf{0}, \pm\mathbf{v}, \pm 2\mathbf{v}, \dots, \}$ to be the lattice generated by a shortest non-zero vector $\mathbf{v}$, we have $\eta(\mathcal{L}') = \text{polylog}(n)\|\mathbf{v}\| = \text{polylog}(n)\lambda_1(\mathcal{L})$ (where the polylogarithmic factor arises because of "how smooth we need $\mathcal{L}'$ to be"). This immediately yields our $\widetilde{O}(\sqrt{n})$-SVP algorithm.

To solve $\widetilde{O}(\sqrt{n})$-HSVP, we must argue that every lattice has a sublattice $\mathcal{L}' \subseteq \mathcal{L}$ with $\eta(\mathcal{L}') \leq \text{polylog}(n) \cdot \det(\mathcal{L})^{1/n}$. In fact, for very different reasons, Dadush conjectured *exactly* this statement (phrased slightly differently), calling it a "reverse Minkowski conjecture" [DR16]. (The reason for this name might not be clear in this context, but one can show that this is a partial converse to Minkowski's theorem.) Later, Regev and Stephens-Davidowitz proved the conjecture [RS17]. Our HSVP result then follows from this rather heavy hammer.

1.5 Open questions and directions for future work

We leave one obvious open question: Does our algorithm (or some variant) solve γ-SVP for a better approximation factor? It is clear that our current analysis

cannot hope to do better than $\delta \approx \sqrt{n}$, but we see no fundamental reason why the algorithm cannot achieve, say, $\delta = \text{polylog}(n)$ or even $\delta = 1$! (Indeed, we have been trying to prove something like this for roughly five years.)

We think that even a negative answer to this question would also be interesting. In particular, it is not currently clear whether our algorithm is "fundamentally an HSVP algorithm." For example, if one could show that our algorithm fails to output vectors of length $\text{polylog}(n) \cdot \lambda_1(\mathcal{L})$ for some family of input lattices $\mathcal{L}$, then this would be rather surprising. Perhaps such a result could suggest a true algorithmic separation between the two problems.

2 Preliminaries

We write log for the base-two logarithm. We use the notation $a = 1 \pm \delta$ and $a = e^{\pm\delta}$ to denote the statements $1 - \delta \leq a \leq 1 + \delta$ and $e^{-\delta} \leq a \leq e^{\delta}$, respectively.

Definition 2.1. *We say that a distribution $\widehat{D}$ is δ-similar to another distribution D if for all $\mathbf{x}$ in the support of D, we have*

$$\Pr_{\mathbf{X}\sim\widehat{D}}[\mathbf{X} = \mathbf{x}] = e^{\pm\delta} \cdot \Pr_{\mathbf{X}\sim D}[\mathbf{X} = \mathbf{x}] \ .$$

2.1 Probability

The following inequality gives a concentration result for the values of (sub-)martingales that have bounded differences.

Lemma 2.2 ([AS04, Azuma's inequality, Chapter 7])**.** *Let $X_0, X_1, \ldots$ be a set of random variables that form a discrete-time sub-martingale, i.e., for all $n \geq 0$,*

$$\mathbb{E}[X_{n+1} \mid X_1, \ldots, X_n] \geq X_n \ .$$

If for all $n \geq 0$, $|X_n - X_{n-1}| \leq c$, then for all integers N and positive real t,

$$\Pr[X_N - X_0 \leq -t] \leq \exp\left(\frac{-t^2}{2Nc^2}\right) \ .$$

We will need the following corollary of the above inequality.

Corollary 2.3. *Let $\alpha \in (0, 1)$, and let $Y_1, Y_2, Y_3, \ldots$ be random variables in $[0, 1]$ such that for all $n \geq 0$*

$$\mathbb{E}[Y_{n+1} | Y_1, \ldots, Y_n] \geq \alpha \ .$$

Then, for all positive integers N and positive real t,

$$\Pr[\sum_{i=1}^{N} Y_i \leq N\alpha - t] \leq \exp\left(\frac{-t^2}{2N}\right) \ .$$

Proof. Let $X_0 = 0$, and for all $i \geq 1$,

$$X_i := X_{i-1} + Y_i - \alpha = \sum_{j=1}^{i} Y_i - i \cdot \alpha \ .$$

The statement then follows immediately from Lemma 2.2. □

2.2 Lattices

A lattice $\mathcal{L} \subset \mathbb{R}^n$ is the set of integer linear combinations

$$\mathcal{L} := \mathcal{L}(\mathbf{B}) = \{z_1\mathbf{b}_1 + \cdots + z_k\mathbf{b}_k \ : \ z_i \in \mathbb{Z}\}$$

of linearly independent basis vectors $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_k) \in \mathbb{R}^{n \times k}$. We call k the *rank* of the lattice. Given a lattice $\mathcal{L}$, the basis is not unique. For any lattice $\mathcal{L}$, we use $\mathrm{rank}(\mathcal{L})$ to denote its rank. We use $\lambda_1(\mathcal{L})$ to denote the length of the shortest non-zero vector in $\mathcal{L}$, and more generally, for $1 \leq i \leq k$,

$$\lambda_i(\mathcal{L}) := \min\{r \ : \ \dim \mathrm{span}(\{\mathbf{y} \in \mathcal{L} \ : \ \|\mathbf{y}\| \leq r\}) \geq i\} \ .$$

For any lattice $\mathcal{L} \subset \mathbb{R}^n$, its dual lattice $\mathcal{L}^*$ is defined to be the set of vectors in the span of $\mathcal{L}$ that have integer inner products with all vectors in $\mathcal{L}$. More formally:

$$\mathcal{L}^* = \{\mathbf{x} \in \mathrm{span}(\mathcal{L}) : \forall \mathbf{y} \in \mathcal{L}, \langle \mathbf{x}, \mathbf{y} \rangle \in \mathbb{Z}\} \ .$$

We often assume without loss of generality that the lattice is full rank, i.e., that $n = k$, by identifying $\mathrm{span}(\mathcal{L})$ with $\mathbb{R}^k$. However, we do often work with sublattices $\mathcal{L}' \subseteq \mathcal{L}$ with $\mathrm{rank}(\mathcal{L}') < \mathrm{rank}(\mathcal{L})$.

For any sublattice $\mathcal{L}' \subseteq \mathcal{L}$, $\mathcal{L}/\mathcal{L}'$ denotes the set of cosets which are translations of $\mathcal{L}'$ by vectors in $\mathcal{L}$. In particular, any coset can be denoted as $\mathcal{L}' + \mathbf{c}$ for $\mathbf{c} \in \mathcal{L}$. When there is no ambiguity, we drop the $\mathcal{L}'$ and use $\mathbf{c}$ to denote a coset.

2.3 The discrete Gaussian Distribution

For any parameter $s > 0$, we define Gaussian mass function $\rho_s : \mathbb{R}^n \to \mathbb{R}$ to be:

$$\rho_s(\mathbf{x}) = \exp\Big(-\frac{\pi\|\mathbf{x}\|^2}{s^2}\Big) \ ,$$

and for any discrete set $A \subset \mathbb{R}^n$, its Gaussian mass is defined as $\rho_s(A) = \sum_{\mathbf{x} \in A} \rho_s(\mathbf{x})$.

For a lattice $\mathcal{L} \subset \mathbb{R}^n$, shift $\mathbf{t} \in \mathbb{R}^n$, and parameter $s > 0$, we have the following convenient formula for the Gaussian mass of the lattice coset $\mathcal{L} + \mathbf{t}$, which follows from the Poisson Summation Formula

$$\rho_s(\mathcal{L} + \mathbf{t}) = \frac{s^n}{\det(\mathcal{L})} \cdot \sum_{\mathbf{w} \in \mathcal{L}^*} \rho_{1/s}(\mathbf{w}) \cos(2\pi\langle \mathbf{w}, \mathbf{t} \rangle) \ . \tag{3}$$

In particular, for the special case $\mathbf{t} = \mathbf{0}$, we have $\rho_s(\mathcal{L}) = s^n \rho_{1/s}(\mathcal{L}^*)/\det(\mathcal{L})$.

Definition 2.4. *For a lattice $\mathcal{L} \subset \mathbb{R}^n$, $\mathbf{u} \in \mathbb{R}^n$, the discrete Gaussian distribution $\mathcal{D}_{\mathcal{L}+\mathbf{u},s}$ over $\mathcal{L}+\mathbf{u}$ with parameter $s > 0$ is defined as follows. For any $\mathbf{x} \in \mathcal{L}+\mathbf{u}$,*

$$\Pr_{\mathbf{X}\sim\mathcal{D}_{\mathcal{L}+\mathbf{u},s}}[\mathbf{X}=\mathbf{x}] = \frac{\rho_s(\mathbf{x})}{\rho_s(\mathcal{L}+\mathbf{u})} .$$

We will need the following result about the discrete Gaussian distribution.

Lemma 2.5 ([DRS14, Lemma 2.13])**.** *For any lattice $\mathcal{L} \subset \mathbb{R}^n$, $s > 0$, $\mathbf{u} \subset \mathbb{R}^n$, and $t > \frac{1}{\sqrt{2\pi}}$,*

$$\Pr_{\mathbf{X}\sim\mathcal{D}_{\mathcal{L}+\mathbf{u},s}}(\|\mathbf{X}\| > ts\sqrt{n}) < \frac{\rho_s(\mathcal{L})}{\rho_s(\mathcal{L}+\mathbf{u})}\left(\sqrt{2\pi e t^2}\exp(-\pi t^2)\right)^n .$$

2.4 The smoothing parameter

Definition 2.6. *For a lattice $\mathcal{L} \subset \mathbb{R}^n$ and $\varepsilon > 0$, the smoothing parameter $\eta_\varepsilon(\mathcal{L})$ is defined as the unique value that satisfies $\rho_{1/\eta_\varepsilon(\mathcal{L})}(\mathcal{L}^* \setminus \{\mathbf{0}\}) = \varepsilon$.*

We will often use the basic fact that $\eta_\varepsilon(\alpha\mathcal{L}) = \alpha\eta_\varepsilon(\mathcal{L})$ for any $\alpha > 0$ and $\eta_\varepsilon(\mathcal{L}') \geq \eta_\varepsilon(\mathcal{L})$ for any *full-rank* sublattice $\mathcal{L}' \subseteq \mathcal{L}$.

Claim 2.7 ([MR07, Lemma 3.3])**.** *For any $\varepsilon \in (0, 1/2)$, we have*

$$\eta_\varepsilon(\mathbb{Z}) \leq \sqrt{\log(1/\varepsilon)} .$$

We will need the following simple results, which follows immediately from Eq. (3).

Lemma 2.8 ([Reg09, Claim 3.8])**.** *For any lattice $\mathcal{L}$, $s \geq \eta_\varepsilon(\mathcal{L})$, and any vectors $\mathbf{c}_1, \mathbf{c}_2$, we have that*

$$\frac{1-\varepsilon}{1+\varepsilon} \leq \frac{\rho_s(\mathcal{L}+\mathbf{c}_1)}{\rho_s(\mathcal{L}+\mathbf{c}_2)} \leq \frac{1+\varepsilon}{1-\varepsilon} .$$

Thus, for $\varepsilon < 1/3$,

$$e^{-3\varepsilon} \leq \frac{\rho_s(\mathcal{L}+\mathbf{c}_1)}{\rho_s(\mathcal{L}+\mathbf{c}_2)} \leq e^{3\varepsilon} .$$

We prove the following statement.

Theorem 2.9. *For any lattice $\mathcal{L} \subset \mathbb{R}^n$ with rank $k \geq 20$,*

$$\eta_{1/2}(\mathcal{L}) \geq \lambda_k(\mathcal{L})/\sqrt{k} .$$

Proof. If $\mathcal{L}$ is not a full-rank lattice, then we can project to a subspace given by the span of $\mathcal{L}$. So, without loss of generality, we assume that $\mathcal{L}$ is a full-rank lattice, i.e., $k = n$.

Suppose $\lambda_n(\mathcal{L}) > \sqrt{n}\eta_{1/2}(\mathcal{L})$. Then there exists a vector $\mathbf{u} \in \mathbb{R}^n$ such that $\mathrm{dist}(\mathbf{u}, \mathcal{L}) > \frac{1}{2}\sqrt{n}\eta_{1/2}(\mathcal{L})$. Then, using Lemma 2.5 with $t = 1/2$, $s = \eta_{1/2}(\mathcal{L})$, we have

$$\begin{aligned}
1 &= \Pr_{\mathbf{X}\sim\mathcal{D}_{\mathcal{L}+\mathbf{u},\eta_{1/2}(\mathcal{L})}}\left[\|\mathbf{X}\| > st\sqrt{n}\right] \\
&< \frac{\rho_s(\mathcal{L})}{\rho_s(\mathcal{L}+\mathbf{u})}\left(\sqrt{2\pi e t^2}\exp(-\pi t^2)\right)^n \\
&\le \frac{1+1/2}{1-1/2}(\sqrt{\pi e/2}\cdot e^{-\pi/4})^n && \text{using Lemma 2.8} \\
&\le 3\cdot(0.943)^n \\
&< 1 && \text{since } \ k = n \ge 20\ ,
\end{aligned}$$

which is a contradiction. □

Claim 2.10. *For any lattice $\mathcal{L} \subset \mathbb{R}^n$ and any parameters $s \ge s' \ge \eta_{1/2}(\mathcal{L})$,*

$$\frac{\rho_s(\mathcal{L})}{\rho_{s'}(\mathcal{L})} \ge \frac{2s}{3s'}\ .$$

Proof. By the Poisson Summation Formula (Eq. (3)), we have

$$\rho_s(\mathcal{L}) = s^n \cdot \frac{\rho_{1/s}(\mathcal{L}^*)}{\det(\mathcal{L})} \ge s^n/\det(\mathcal{L})\ ,$$

and similarly,

$$\rho_{s'}(\mathcal{L}) = (s')^n \cdot \frac{\rho_{1/s'}(\mathcal{L}^*)}{\det(\mathcal{L})} \le 3(s')^n/(2\det(\mathcal{L}))\ ,$$

since $\rho_{1/s'}(\mathcal{L}^*) \le 3/2$ for $s' \ge \eta_{1/2}(\mathcal{L})$. Combining the two inequalities gives $\rho_s(\mathcal{L}) \ge 2(s/s')^n/3 \ge 2(s/s')/3$, as needed. □

Claim 2.11. *For any lattice $\mathcal{L} \subset \mathbb{R}^n$ and any $s > 0$,*

$$\mathop{\mathbb{E}}_{\mathbf{X}\sim D_{\mathcal{L},s}}[\|\mathbf{X}\|^2] \le \frac{ns^2}{2\pi}\ .$$

Lemma 2.12. *For $s \ge \eta_\varepsilon(\mathcal{L})$, and any real factor $k \ge 1$, $ks \ge \eta_{\varepsilon^{k^2}}(\mathcal{L})$.*

Proof.

$$\begin{aligned}
\sum_{\mathbf{w}\in\mathcal{L}^*\setminus\{0\}} \rho_{1/(ks)}(\mathbf{w}) &= \sum_{\mathbf{w}\in\mathcal{L}^*\setminus\{0\}} e^{-\pi\|\mathbf{w}\|k^2s^2} \\
&= \sum_{\mathbf{w}\in\mathcal{L}^*\setminus\{0\}} \rho_{1/s}(\mathbf{w})^{k^2} \\
&\le \Big(\sum_{\mathbf{w}\in\mathcal{L}^*\setminus\{0\}} \rho_{1/s}(\mathbf{w})\Big)^{k^2} \\
&\le \varepsilon^{k^2}\ .
\end{aligned}$$

□

Corollary 2.13. *For any lattice $\mathcal{L} \subset \mathbb{R}^n$ and $\varepsilon \in (0, 1/2)$, $\eta_\varepsilon(\mathcal{L}) \leq \sqrt{\log(1/\varepsilon)} \cdot \eta_{1/2}(\mathcal{L})$.*

Proof. Let $k = \sqrt{\log(1/\varepsilon)}$ and thus $(\frac{1}{2})^{k^2} = \varepsilon$. By Lemma 2.12, $k\eta_{1/2}(\mathcal{L}) \geq \eta_\varepsilon(\mathcal{L})$. □

We will need the following useful lemma concerning the convolution of two discrete Gaussian distributions. See [GMPW20] for a very general result of this form (and a list of similar results). Our lemma differs from those in [GMPW20] and elsewhere in that we are interested in a stronger notion of statistical closeness: point-wise multiplicative distance, rather than statistical distance. One can check that this stronger variant follows from the proofs in [GMPW20], but we give a separate proof for completeness.

Lemma 2.14. *For any lattice $\mathcal{L} \subset \mathbb{R}^n$, $\varepsilon \in (0, 1/3)$, parameter $s \geq \sqrt{2}\eta_\varepsilon(\mathcal{L})$, and shifts $\mathbf{t}_1, \mathbf{t}_2 \in \mathbb{R}^n$, let $\mathbf{X}_i \sim D_{\mathcal{L}+\mathbf{t}_i, s}$ be independent random variables. Then the distribution of $\mathbf{X}_1 + \mathbf{X}_2$ is 6ε-similar to $D_{\mathcal{L}+\mathbf{t}_1+\mathbf{t}_2, \sqrt{2}s}$.*

Proof. Let $\mathbf{y} \in \mathcal{L} + \mathbf{t}_1 + \mathbf{t}_2$. We have

$$\begin{aligned}
\Pr[\mathbf{X}_1 + \mathbf{X}_2 = \mathbf{y}] &= \frac{1}{\rho_s(\mathcal{L}+\mathbf{t}_1)\rho_s(\mathcal{L}+\mathbf{t}_2)} \sum_{\mathbf{x}\in\mathcal{L}+\mathbf{t}_1} \exp(-\pi(\|\mathbf{x}\|^2 + \|\mathbf{y}-\mathbf{x}\|^2)/s^2) \\
&= \frac{1}{\rho_s(\mathcal{L}+\mathbf{t}_1)\rho_s(\mathcal{L}+\mathbf{t}_2)} \sum_{\mathbf{x}\in\mathcal{L}+\mathbf{t}_1} \exp(-\pi(\|\mathbf{y}\|^2/2 + \|2\mathbf{x}-\mathbf{y}\|^2/2)/s^2) \\
&= \frac{\rho_{\sqrt{2}s}(\mathbf{y})}{\rho_s(\mathcal{L}+\mathbf{t}_1)\rho_s(\mathcal{L}+\mathbf{t}_2)} \rho_{s/\sqrt{2}}(\mathcal{L}+\mathbf{t}_1-\mathbf{y}/2) \\
&= e^{\pm 3\varepsilon} \rho_{\sqrt{2}s}(\mathbf{y}) \cdot \frac{\rho_{s/\sqrt{2}}(\mathcal{L})}{\rho_s(\mathcal{L}+\mathbf{t}_1)\rho_s(\mathcal{L}+\mathbf{t}_2)} ,
\end{aligned}$$

where the last step follows from Lemma 2.8. By applying this for all $\mathbf{y}' \in \mathcal{L} + \mathbf{t}_1 + \mathbf{t}_2$, we see that

$$\Pr[\mathbf{X}_1 + \mathbf{X}_2 = \mathbf{y}] = e^{\pm 3\varepsilon} \cdot \frac{\rho_{\sqrt{2}s}(\mathbf{y})}{\sum_{\mathbf{y}'\in\mathcal{L}+\mathbf{t}_1+\mathbf{t}_2} \chi_{\mathbf{y}'} \rho_{\sqrt{2}s}(\mathbf{y}')}$$

for some $\chi_{\mathbf{y}'} = e^{\pm 3\varepsilon}$. Therefore,

$$\Pr[\mathbf{X}_1 + \mathbf{X}_2 = \mathbf{y}] = e^{\pm 6\varepsilon} \cdot \frac{\rho_{\sqrt{2}s}(\mathbf{y})}{\rho_{\sqrt{2}s}(\mathcal{L}+\mathbf{t}_1+\mathbf{t}_2)} ,$$

as needed. □

2.5 Lattice problems

In this paper, we study the algorithms for the following lattice problems.

Definition 2.15 (r-HSVP)**.** *For an approximation factor $r := r(n) \geq 1$, the r-Hermite Approximate Shortest Vector Problem (r-HSVP) is defined as follows: Given a basis $\mathbf{B}$ for a lattice $\mathcal{L} \subset \mathbb{R}^n$, the goal is to output a vector $\mathbf{x} \in \mathcal{L} \backslash \{\mathbf{0}\}$ with $\|\mathbf{x}\| \leq r \cdot \det(\mathcal{L})^{1/n}$.*

Definition 2.16 (r-SVP)**.** *For an approximation factor $r := r(n) \geq 1$, the r-Shortest Vector Problem (r-SVP) is defined as follows: Given a basis $\mathbf{B}$ for a lattice $\mathcal{L} \subset \mathbb{R}^n$, the goal is to output a vector $\mathbf{x} \in \mathcal{L} \backslash \{\mathbf{0}\}$ with $\|\mathbf{x}\| \leq r \cdot \lambda_1(\mathcal{L})$.*

It will be convenient to define a generalized version of SVP, of which HSVP and SVP are special cases.

Definition 2.17 (η-GSVP)**.** *For a function η mapping lattices to positive real numbers, the η-Generalized Shortest Vector Problem η-GSVP is defined as follows: Given a basis $\mathbf{B}$ for a lattice $\mathcal{L} \subset \mathbb{R}^n$ and a length bound $d \geq \eta(\mathcal{L})$, the goal is to output a vector $\mathbf{x} \in \mathcal{L} \backslash \{\mathbf{0}\}$ with $\|\mathbf{x}\| \leq d$.*

To recover r-SVP or r'-HSVP, we can take $\eta(\mathcal{L}) = r\lambda_1(\mathcal{L})$ or $\eta(\mathcal{L}) = r' \det(\mathcal{L})^{1/n}$ respectively. Below, we will set η to be a new parameter, which in particular will satisfy $\eta(\mathcal{L}) \leq \widetilde{O}(\sqrt{n}) \cdot \min\{\lambda_1(\mathcal{L}), \det(\mathcal{L})^{1/n}\}$.

2.6 Gram-Schmidt orthogonalization

For any given basis $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n) \in \mathbb{R}^{m \times n}$, we define the sequence of projections $\pi_i := \pi_{\{\mathbf{b}_1, \ldots, \mathbf{b}_{i-1}\}^\perp}$ where $\pi_{W^\perp}$ refers to the orthogonal projection onto the subspace orthogonal to W. As in [GN08, ALNS20], we use $\mathbf{B}_{[i,j]}$ to denote the projected block $(\pi_i(\mathbf{b}_i), \pi_i(\mathbf{b}_{i+1}), \ldots, \pi_i(\mathbf{b}_j))$.

The Gram-Schmidt orthogonalization (GSO) $\mathbf{B}^* := (\mathbf{b}_1^*, \ldots, \mathbf{b}_n^*)$ of a basis $\mathbf{B}$ is as follows: for all $i \in [1, n]$, $\mathbf{b}_i^* := \pi_i(\mathbf{b}_i) = \mathbf{b}_i - \sum_{j<i} \mu_{i,j} \mathbf{b}_j^*$, where $\mu_{i,j} = \langle \mathbf{b}_i, \mathbf{b}_j^* \rangle / \|\mathbf{b}_j^*\|^2$.

Theorem 2.18 ([GPV08, Lemma 3.1])**.** *For any lattice $\mathcal{L} \subset \mathbb{R}^n$ with basis $\mathbf{B} := (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ and any $\varepsilon \in (0, 1/2)$,*

$$\eta_\varepsilon(\mathcal{L}) \leq \sqrt{\log(n/\varepsilon)} \cdot \max_i \|\mathbf{b}_i^*\| \,.$$

For $\gamma \geq 1$, a basis is *γ-HKZ-reduced* if for all $i \in \{1, \ldots, n\}$, $\|\mathbf{b}_i^*\| \leq \gamma \cdot \lambda_1(\pi_i(\mathcal{L}))$.

We say that a basis $\mathbf{B}$ is *size-reduced* if it satisfies the following condition: for all $i \neq j$, $|\mu_{i,j}| \leq \frac{1}{2}$. A size-reduced basis $\mathbf{B}$ satisfies that $\|\mathbf{B}\| \leq \sqrt{n} \|\mathbf{B}^*\|$, where $\|\mathbf{B}\|$ is the length of the longest basis vector in $\mathbf{B}$. It is known that we can efficiently transform any basis into a size-reduced basis while maintaining the lattice generated by the basis $\mathcal{L}(\mathbf{B})$ as well as the GSO $\mathbf{B}^*$. We call such operation *size reduction*.

2.7 Some lattice algorithms

Theorem 2.19 ([LLL82]). *Given a basis $\mathbf{B} \in \mathbb{Q}^{n \times n}$, there is an algorithm that computes a vector $\mathbf{x} \in \mathcal{L}(\mathbf{B})$ of length at most $2^{n/2} \cdot \lambda_1(\mathcal{L}(\mathbf{B}))$ in polynomial time.*

We will prove a strictly stronger result than the theorem below in the sequel, but this weaker result will still prove useful.

Theorem 2.20 ([ADRS15, GN08]). *There is a $2^{r+o(r)} \cdot \mathrm{poly}(n)$-time algorithm that takes as input a (basis for a) lattice $\mathcal{L} \subset \mathbb{R}^n$ and $2 \leq r \leq n$ and outputs a γ-HKZ-reduced basis for $\mathcal{L}$, where $\gamma := r^{n/r}$.*

Theorem 2.21 ([BLP+13]). *There is a probabilistic polynomial-time algorithm that takes as input a basis $\mathbf{B}$ for an n-dimensional lattice $\mathcal{L} \subset \mathbb{R}^n$, a parameter $s \geq \|\mathbf{B}^*\|\sqrt{10 \log n}$ and outputs a vector that is distributed as $\mathcal{D}_{\mathcal{L},s}$, where $\|\mathbf{B}^*\|$ is the length of the longest vector in the Gram-Schmidt orthogonalization of $\mathbf{B}$.*

2.8 Lattice basis reduction

LLL reduction. A basis $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ is *ε-LLL-reduced* [LLL82] for $\varepsilon \in [0, 1]$ if it is a size-reduced basis and for $1 \leq i < n$, the projected block $\mathbf{B}_{[i,i+1]}$ satisfies Lovász's condition: $\|\mathbf{b}_i^*\|^2 \leq (1+\varepsilon)\|\mu_{i,i-1}\mathbf{b}_{i-1}^* + \mathbf{b}_i^*\|^2$. For $\varepsilon \geq 1/\mathrm{poly}(n)$, an ε-LLL-reduced basis for any given lattice can be computed efficiently.

SVP reduction and its extensions. Let $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ be a basis of a lattice $\mathcal{L}$ and $\delta \geq 1$ be approximation factors.

We say that $\mathbf{B}$ is *δ-SVP-reduced* if $\|\mathbf{b}_1\| \leq \delta \cdot \lambda_1(\mathcal{L})$. Similarly, we say that $\mathbf{B}$ is *δ-HSVP-reduced* if $\|\mathbf{b}_1\| \leq \delta \cdot \mathrm{vol}(\mathcal{L})^{1/n}$.

$\mathbf{B}$ is *δ-DHSVP-reduced* [GN08, ALNS20] (where D stands for dual) if the reversed dual basis $\mathbf{B}^{-s}$ is δ-HSVP-reduced and it implies that

$$\mathrm{vol}(\mathcal{L})^{1/n} \leq \delta \cdot \|\mathbf{b}_n^*\| \,.$$

Given a δ-(H)SVP oracle on lattices with rank at most n, we can efficiently compute a δ-(H)SVP-reduced basis or a δ-D(H)SVP-reduced basis for any rank n lattice $\mathcal{L} \subseteq \mathbb{Z}^m$. Furthermore, this also applies for a projected block of basis. More specifically, with access to a δ-(H)SVP oracle for lattices with rank at most k, given any basis $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n) \in \mathbb{Z}^{m \times n}$ of $\mathcal{L}$ and an index $i \in [1, n-k+1]$, we can efficiently compute a size-reduced basis

$$\mathbf{C} = (\mathbf{b}_1, \ldots, \mathbf{b}_{i-1}, \mathbf{c}_i, \ldots, \mathbf{c}_{i+k-1}, \mathbf{b}_{i+k}, \ldots, \mathbf{b}_n)$$

such that $\mathbf{C}$ is a basis for $\mathcal{L}$ and the projected block $\mathbf{C}_{[i,i+k-1]}$ is δ-(H)SVP-reduced or δ-D(H)SVP reduced. Moreover, we note the following:

- If $\mathbf{C}_{[i,i+k-1]}$ is δ-(H)SVP-reduced, the procedures in [GN08, MW16] equipped with δ-(H)SVP-oracle ensure that $\|\mathbf{C}^*\| \leq \|\mathbf{B}^*\|$;

– If $\mathbf{C}_{[i,i+k-1]}$ is δ-D(H)SVP-reduced, the inherent LLL reduction implies $\|\mathbf{C}^*\| \leq 2^k\|\mathbf{B}^*\|$. Indeed, the GSO of $\mathbf{C}_{[i,i+k-1]}$ satisfies

$$\|(\mathbf{C}_{[i,i+k-1]})^*\| \leq 2^{k/2}\lambda_k(\mathcal{L}(\mathbf{C}_{[i,i+k-1]}))$$

(by [LLL82, p. 518, Line 27]) and $\lambda_k(\mathcal{L}(\mathbf{C}_{[i,i+k-1]})) \leq \sqrt{k}\|\mathbf{B}^*\|$. Here, $\lambda_k(\cdot)$ denotes the k-th minimum.

Therefore, with size reduction, performing $\mathrm{poly}(n, \log\|\mathbf{B}\|)$ many such operations will increase $\|\mathbf{B}^*\|$ and hence $\|\mathbf{B}\|$ by at most a factor of $2^{\mathrm{poly}(n,\log\|B\|)}$. If the number of operations is bounded by $\mathrm{poly}(n, \log\|\mathbf{B}\|)$, all intermediate steps and the total running time (excluding oracle queries) will be polynomial in the initial input size; Details can be found in e.g., [GN08, LN14]. Hence, we will focus on bounding the number of calls to such block reduction subprocedures when we analyze the running time of basis reduction algorithms.

Twin reduction The following notion of twin reduction and the subsequent fact comes from [GN08, ALNS20].

A basis $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_{d+1})$ is *δ-twin-reduced* if $\mathbf{B}_{[1,d]}$ is *δ-HSVP-reduced* and $\mathbf{B}_{[2,d+1]}$ is *δ-DHSVP-reduced*.

Fact 2.22. *If $\mathbf{B} := (\mathbf{b}_1, \ldots, \mathbf{b}_{d+1}) \in \mathbb{R}^{m\times(d+1)}$ is δ-twin-reduced, then*

$$\|\mathbf{b}_1\| \leq \delta^{2d/(d-1)}\|\mathbf{b}^*_{d+1}\| \,. \tag{4}$$

2.9 The DBKZ algorithm

We augment Micciancio and Walter's elegant DBKZ algorithm [MW16] with a δ_H-HSVP-oracle instead of an SVP-oracle since the SVP-oracle is used as a $\sqrt{\gamma_k}$-HSVP oracle everywhere in their algorithm. See [ALNS20] for a high-level sketch of the proof.

Algorithm 1 The Micciancio-Walter DBKZ algorithm [MW16, Algorithm 1]

Input: A block size $k \geq 2$, number of tours N, a basis $\mathbf{B} = (\mathbf{b}_1, \cdots, \mathbf{b}_n) \in \mathbb{Z}^{m\times n}$, and access to a δ_H-HSVP oracle for lattices with rank k.
Output: A new basis of $\mathcal{L}(\mathbf{B})$.

for $\ell = 1$ **to** N **do**
  for $i = 1$ **to** $n - k$ **do**
    δ_H-HSVP-reduce $\mathbf{B}_{[i,i+k-1]}$.
  end for
  for $j = n - k + 1$ **to** 1 **do**
    δ_H-DHSVP-reduce $\mathbf{B}_{[j,j+k-1]}$
  end for
end for
δ_H-HSVP-reduce $\mathbf{B}_{[1,k]}$.
return $\mathbf{B}$.

Theorem 2.23. *For integers $n > k \geq 2$, an approximation factor $1 \leq \delta_H \leq 2^k$, an input basis $\mathbf{B}_0 \in \mathbb{Z}^{m \times n}$ for a lattice $\mathcal{L} \subseteq \mathbb{Z}^m$, and $N := \lceil (2n^2/(k-1)^2) \cdot \log(n \log(5\|\mathbf{B}_0\|)/\varepsilon) \rceil$ for some $\varepsilon \in [2^{-\mathrm{poly}(n)}, 1]$, Algorithm 1 outputs a basis $\mathbf{B}$ of $\mathcal{L}$ in polynomial time (excluding oracle queries) such that*

$$\|\mathbf{b}_1\| \leq (1+\varepsilon) \cdot (\delta_H)^{\frac{n-1}{(k-1)}} \mathrm{vol}(\mathcal{L})^{1/n} ,$$

by making $N \cdot (2n - 2k + 1) + 1$ calls to the δ_H-HSVP oracle for lattices with rank k.

3 Smooth sublattices and $\overline{\eta}_\varepsilon(\mathcal{L})$

The analysis of our algorithm relies on the existence of a *smooth sublattice* $\mathcal{L}' \subseteq \mathcal{L}$ of our input lattice $\mathcal{L} \subset \mathbb{R}^n$, i.e., a sublattice $\mathcal{L}'$ such that $\eta_\varepsilon(\mathcal{L}')$ is small (relative to, say, $\lambda_1(\mathcal{L})$ or $\det(\mathcal{L})^{1/n}$). To that end, for $\varepsilon > 0$ and a lattice $\mathcal{L} \subset \mathbb{R}^n$, we define

$$\overline{\eta}_\varepsilon(\mathcal{L}) := \min_{\mathcal{L}' \subseteq \mathcal{L}} \eta_\varepsilon(\mathcal{L}') ,$$

where the minimum is taken over all sublattices $\mathcal{L}' \subseteq \mathcal{L}$. (It is not hard to see that the minimum is in fact achieved. Notice that any minimizer $\mathcal{L}'$ must be a primitive sublattice, i.e., $\mathcal{L}' = \mathcal{L} \cap \mathrm{span}(\mathcal{L}')$.)

We will now prove that $\overline{\eta}_\varepsilon(\mathcal{L})$ is bounded both in terms of $\lambda_1(\mathcal{L})$ and $\det(\mathcal{L})$.

Lemma 3.1. *For any lattice $\mathcal{L} \subset \mathbb{R}^n$ and any $\varepsilon \in (0, 1/2)$,*

$$\lambda_1(\mathcal{L})/\sqrt{n} \leq \overline{\eta}_\varepsilon(\mathcal{L}) \leq \sqrt{\log(1/\varepsilon)} \cdot \min\{\lambda_1(\mathcal{L}), 10(\log n + 2) \det(\mathcal{L})^{1/n}\} .$$

The bounds in terms of $\lambda_1(\mathcal{L})$ are more-or-less trivial. The bound $\overline{\eta}_\varepsilon(\mathcal{L}) \lesssim \sqrt{\log(1/\varepsilon) \log n} \det(\mathcal{L})^{1/n}$ follows from the main result in [RS17] (originally conjectured by Dadush [DR16]), which is called a "reverse Minkowski theorem" and which we present below. (In fact, Lemma 3.1 is essentially equivalent to the main result in [RS17].)

Definition 3.2. *A lattice $\mathcal{L} \subset \mathbb{R}^n$ is a **stable** lattice if $\det(\mathcal{L}) = 1$ and $\det(\mathcal{L}') \geq 1$ for all lattices $\mathcal{L}' \subseteq \mathcal{L}$.*

Theorem 3.3 ([RS17])**.** *For any stable lattice $\mathcal{L} \subset \mathbb{R}^n$, $\eta_{1/2}(\mathcal{L}) \leq 10(\log n + 2)$.*

Proof of Lemma 3.1. The lower bound on $\overline{\eta}_\varepsilon(\mathcal{L})$ follows immediately from Theorem 2.9 together with the fact that $\lambda_1(\mathcal{L}) \leq \lambda_1(\mathcal{L}') \leq \lambda_n(\mathcal{L}')$ for any sublattice $\mathcal{L}' \subseteq \mathcal{L}$. The bound $\overline{\eta}_\varepsilon(\mathcal{L}) \leq \sqrt{\log(1/\varepsilon)} \cdot \lambda_1(\mathcal{L})$ is immediate from Claim 2.7 applied to the one-dimensional lattice $\mathbb{Z}\mathbf{v}$ generated by $\mathbf{v} \in \mathcal{L}$ with $\|\mathbf{v}\| = \lambda_1(\mathcal{L})$.

So, we only need to prove that $\overline{\eta}_{1/2}(\mathcal{L}) \leq 10(\log n + 2) \det(\mathcal{L})^{1/n}$. The result for all $\varepsilon \in (0, 1/2)$ then follows from Corollary 2.13.

We prove this by induction on n. The result is trivial for $n = 1$. (Indeed, for $n = 1$ we have $\det(\mathcal{L})^{1/n} = \lambda_1(\mathcal{L})$.) For $n > 1$, we first assume without loss of generality that $\det(\mathcal{L}) = 1$. If $\mathcal{L} \subset \mathbb{R}^n$ is stable, then the result follows

immediately from Theorem 3.3. Otherwise, there exists a sublattice $\mathcal{L}' \subset \mathcal{L}$ such that $\det(\mathcal{L}') < 1$. Notice that $k := \operatorname{rank}(\mathcal{L}') < n$. Therefore, by the induction hypothesis, $\overline{\eta}_{1/2}(\mathcal{L}') \le 10(\log k + 2)\det(\mathcal{L}')^{1/k} < 10(\log n + 2)$. The result then follows from the fact that $\overline{\eta}_\varepsilon(\mathcal{L}) \le \overline{\eta}_\varepsilon(\mathcal{L}')$ for any sublattice $\mathcal{L}' \subseteq \mathcal{L}$. □

3.1 Sampling with parameter $\mathbf{poly}(n) \cdot \overline{\eta}_\varepsilon(\mathcal{L})$

Lemma 3.4. *For any lattice $\mathcal{L} \subset \mathbb{R}^n$, $\gamma \ge 1$, $\varepsilon \in (0, 1/2)$, γ-HKZ-reduced basis $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ of $\mathcal{L}$, $\varepsilon \in (0, 1/2)$, and index $i \in \{2, \ldots, n\}$ such that*

$$\|\mathbf{b}_i^*\| > \gamma\sqrt{n} \cdot \overline{\eta}_\varepsilon(\mathcal{L}) \,,$$

we have $\overline{\eta}_\varepsilon(\mathcal{L}(\mathbf{b}_1, \ldots, \mathbf{b}_{i-1})) = \overline{\eta}_\varepsilon(\mathcal{L})$.

Proof. Suppose that $\mathcal{L}' \subseteq \mathcal{L}$ satisfies $\eta_\varepsilon(\mathcal{L}') = \overline{\eta}_\varepsilon(\mathcal{L}) < \|\mathbf{b}_i^*\|/(\gamma\sqrt{n})$ with $k := \operatorname{rank}(\mathcal{L}')$. We wish to show that $\mathcal{L}' \subseteq \mathcal{L}(\mathbf{b}_1, \ldots, \mathbf{b}_{i-1})$, or equivalently, that $\pi_i(\mathcal{L}') = \{\mathbf{0}\}$. Indeed, by Theorem 2.9, $\lambda_k(\mathcal{L}') \le \sqrt{k} \cdot \eta_\varepsilon(\mathcal{L}') \le \sqrt{n} \cdot \overline{\eta}_\varepsilon(\mathcal{L})$. In particular, there exist $\mathbf{v}_1, \ldots, \mathbf{v}_k \in \mathcal{L}'$ with $\operatorname{span}(\mathbf{v}_1, \ldots, \mathbf{v}_k) = \operatorname{span}(\mathcal{L}')$ and

$$\|\pi_i(\mathbf{v}_j)\| \le \|\mathbf{v}_j\| \le \lambda_k(\mathcal{L}') \le \sqrt{n} \cdot \overline{\eta}_\varepsilon(\mathcal{L}) < \|\mathbf{b}_i^*\|/\gamma$$

for all $j \in \{1, \ldots, k\}$. Therefore, if $\pi_i(\mathbf{v}_j) \ne \mathbf{0}$. Then, $\pi_i(\mathbf{v}_j) \in \pi_i(\mathcal{L})$ is a non-zero vector with norm strictly less than $\|\mathbf{b}_i^*\|/\gamma$, which implies that $\lambda_1(\pi_i(\mathcal{L})) < \|\mathbf{b}_i^*\|/\gamma$, contradicting the assumption that $\mathbf{B}$ is a γ-HKZ basis. Therefore, $\pi_i(\mathbf{v}_j) = \mathbf{0}$ for all j, which implies that $\pi_i(\mathcal{L}') = \{\mathbf{0}\}$, i.e., $\mathcal{L}' \subseteq \mathcal{L}(\mathbf{b}_1, \ldots, \mathbf{b}_{i-1})$, as needed. □

Proposition 3.5. *There is a $(2^{r+o(r)} + M) \cdot \operatorname{poly}(n, \log M)$-time algorithm that takes as input a (basis for a) lattice $\mathcal{L} \subset \mathbb{R}^n$, $2 \le r \le n$, an integer $M \ge 1$, and a parameter*

$$s \ge r^{n/r}\sqrt{n \log n} \cdot \overline{\eta}_\varepsilon(\mathcal{L})$$

for some $\varepsilon \in (0, 1/2)$ and outputs a (basis for a) sublattice $\widehat{\mathcal{L}} \subseteq \mathcal{L}$ with $\overline{\eta}_\varepsilon(\widehat{\mathcal{L}}) = \overline{\eta}_\varepsilon(\mathcal{L})$ and $\mathbf{X}_1, \ldots, \mathbf{X}_M \in \widehat{\mathcal{L}}$ that are sampled independently from $D_{\widehat{\mathcal{L}},s}$.

Proof. The algorithm takes as input a (basis for a) lattice $\mathcal{L} \subset \mathbb{R}^n$, $2 \le r \le n$, $M \ge 1$, and a parameter $s > 0$ and behaves as follows. It first uses the procedure from Theorem 2.20 to compute a γ-HKZ reduced basis $\mathbf{b}_1, \ldots, \mathbf{b}_n$, where $\gamma := r^{n/r}$. Let $i \in \{1, \ldots, n\}$ be maximal such that $\|\mathbf{b}_j^*\| \le s/\sqrt{\log n}$ for all $j \le i$, and let $\widehat{\mathcal{L}} := \mathcal{L}(\mathbf{b}_1, \ldots, \mathbf{b}_i)$. (If no such i exists, the algorithm simply fails.) The algorithm then runs the procedure from Theorem 2.21 repeatedly to sample $\mathbf{X}_1, \ldots, \mathbf{X}_M \sim D_{\widehat{\mathcal{L}},s}$ and outputs $\widehat{\mathcal{L}}$ and $\mathbf{X}_1, \ldots, \mathbf{X}_M$.

The running time of the algorithm is clearly $(2^r + M) \cdot \operatorname{poly}(n, \log M)$. By Theorem 2.21, the $\mathbf{X}_i$ have the correct distribution. Notice that, if the algorithm fails, then

$$\|\mathbf{b}_1\| > s/\sqrt{\log n} \ge \gamma\sqrt{n} \cdot \overline{\eta}_\varepsilon(\mathcal{L}) \,.$$

Recalling that $\|\mathbf{b}_1\| \le \gamma\lambda_1(\mathcal{L})$, it follows that $\sqrt{n}\overline{\eta}_\varepsilon(\mathcal{L}) < \lambda_1(\mathcal{L})$, which contradicts Lemma 3.1. So, the algorithm never fails (provided that the promise on s holds).

It remains to show that $\overline{\eta}_\varepsilon(\mathcal{L}) = \overline{\eta}_\varepsilon(\mathcal{L}(\mathbf{b}_1, \ldots, \mathbf{b}_i))$. If $i = n$, then this is trivial. Otherwise, $i \in \{1, \ldots, n-1\}$, and we have

$$\|\mathbf{b}_{i+1}^*\| > s/\sqrt{\log n} \ge \gamma\sqrt{n} \cdot \overline{\eta}_\varepsilon(\mathcal{L}) \ .$$

The result follows immediately from Lemma 3.4. □

4 An approximation algorithm for HSVP and SVP

In this section, we present our algorithm that solves $\widetilde{O}(\sqrt{n})$-HSVP and $\widetilde{O}(\sqrt{n})$-SVP in $2^{n/2+o(n)}$ time. More precisely, we provide a detailed analysis of a simple "pair-and-sum" algorithm, which will solve $O(\sqrt{n}) \cdot \overline{\eta}_\varepsilon(\mathcal{L})$-GSVP for $\varepsilon = 1/\mathrm{poly}(n)$. This in particular yields an algorithm that simultaneously solves $\widetilde{O}(\sqrt{n})$-SVP and $\widetilde{O}(\sqrt{n})$-HSVP.

4.1 Mixtures of Gaussians

We will be working with random variables $\mathbf{X}$ that are "mixtures" of discrete Gaussians, i.e., random variables that can be written as $D_{\mathcal{L}+\mathbf{C},s}$ for some lattice $\mathcal{L} \subset \mathbb{R}^n$, parameter $s > 0$, and random variable $\mathbf{C} \in \mathbb{R}^n$. In other words, $\mathbf{X}$ can be sampled by first sampling $\mathbf{C} \in \mathbb{R}^n$ from some arbitrary distribution and then sampling $\mathbf{X}$ from $D_{\mathcal{L}+\mathbf{C},s}$. E.g., the discrete Gaussian $D_{\mathcal{L},s}$ itself is such a distribution, as is the discrete Gaussian $D_{\widehat{\mathcal{L}},s}$ for any superlattice $\widehat{\mathcal{L}} \supseteq \mathcal{L}$. Indeed, in our applications, we will always have $\mathbf{C} \in \widehat{\mathcal{L}}$ for some superlattice $\widehat{\mathcal{L}} \supseteq \mathcal{L}$, and we will initialize our algorithm with samples from $D_{\widehat{\mathcal{L}},s}$.

Our formal definition below is a bit technical, since we must consider the joint distribution of many such random variables that are only δ-similar to these distributions and satisfy a certain independence property. In particular, we will work with $\mathbf{X}_1, \ldots, \mathbf{X}_M$ such that each $\mathbf{X}_i$ is δ-similar to $\mathbf{Y}_i \sim D_{\mathcal{L}+\mathbf{C}_i,s}$, where $\mathbf{C}_i$ is an arbitrary random variable (that might depend on the $\mathbf{X}_j$) but once $\mathbf{C}_i$ is fixed, $\mathbf{Y}_i$ is sampled from $D_{\mathcal{L}+\mathbf{C}_i,s}$ independently of everything else. Here and below, we adopt the convention that $\Pr[A \mid B] = 0$ whenever $\Pr[B] = 0$, i.e., all probabilities are zero when conditioned on events with probability zero.

Definition 4.1. *For (discrete) random variables $\mathbf{X}_1, \ldots, \mathbf{X}_m \in \mathbb{R}^n$ and $i \in \{1, \ldots, m\}$, let us define the tuple of random variables*

$$\mathbf{X}_{-i} := (\mathbf{X}_1, \ldots, \mathbf{X}_{i-1}, \mathbf{X}_{i+1}, \ldots, \mathbf{X}_m) \in \mathbb{R}^{(m-1)n} \ .$$

We say that $\mathbf{X}_1, \ldots, \mathbf{X}_m$ are δ-similar to a mixture of independent Gaussians over $\mathcal{L}$ with parameter $s > 0$ if for any $i \in \{1, \ldots, m\}$, $\mathbf{y} \in \mathbb{R}^n$, and $\mathbf{w} \in \mathbb{R}^{(m-1)n}$,

$$\Pr[\mathbf{X}_i = \mathbf{y} \mid \mathbf{X}_{-i} = \mathbf{w}] = e^{\pm\delta} \cdot \frac{\rho_s(\mathbf{y})}{\rho_s(\mathcal{L}+\mathbf{y})} \cdot \Pr[\mathbf{X}_i \in \mathcal{L}+\mathbf{y} \mid \mathbf{X}_{-i} = \mathbf{w}] \ .$$

Additionally we will need the distribution we obtain at every step to be symmetric about the origin as defined below.

Definition 4.2. *We say that a list of (discrete) random variables $\mathbf{X}_1, \ldots, \mathbf{X}_m \in \mathbb{R}^n$ is symmetric if for any $i \in \{1, \ldots, m\}$, any $\mathbf{y} \in \mathbb{R}^n$, and any $\mathbf{w} \in \mathbb{R}^{(m-1)n}$,*

$$\Pr[\mathbf{X}_i = \mathbf{y} \mid \mathbf{X}_{-i} = \mathbf{w}] = \Pr[\mathbf{X}_i = -\mathbf{y} \mid \mathbf{X}_{-i} = \mathbf{w}] \,.$$

We need the following simple lemma that bounds the probability of $\mathbf{X}$ being $\mathbf{0}$, where $\mathbf{X}$ is distributed as a mixture of discrete Gaussians over $\mathcal{L}$.

Lemma 4.3. *For any lattice $\mathcal{L} \subset \mathbb{R}^n$, let $\mathbf{X}_1, \ldots, \mathbf{X}_m \in \mathcal{L}$ be δ-similar to a mixture of independent Gaussians over $\mathcal{L}$ with parameter $s \geq \beta\eta_{1/2}(\mathcal{L})$ for some $\beta > 1$. Then, for any i, and any $\mathbf{w} \in \mathbb{R}^{(m-1)n}$*

$$\Pr[\mathbf{X}_i = \mathbf{0} \mid \mathbf{X}_{-i} = \mathbf{w}] \leq \frac{3e^{\delta}}{2\beta} \,.$$

Proof. Let $s' := \eta_{1/2}(\mathcal{L})$. We have that

$$\Pr[\mathbf{X}_i = \mathbf{0} \mid \mathbf{X}_{-i} = \mathbf{w}] \leq \Pr[\mathbf{X}_i = \mathbf{0} \mid \mathbf{X}_i \in \mathcal{L},\ \mathbf{X}_{-i} = \mathbf{w}] \leq \frac{e^{\delta}}{\rho_s(\mathcal{L})} \leq e^{\delta} \cdot \frac{\rho_{s'}(\mathcal{L})}{\rho_s(\mathcal{L})} \,.$$

The result then follows from Claim 2.10. □

The following corollary shows that a mixture of discrete Gaussians must contain a short non-zero vector in certain cases.

Corollary 4.4. *For any lattices $\mathcal{L}' \subseteq \mathcal{L} \subset \mathbb{R}^n$, parameter $s \geq 10e^{\delta}\eta_{1/2}(\mathcal{L}')$, $m \geq 100$, and random variables $\mathbf{X}_1, \ldots, \mathbf{X}_m$ that are δ-similar to mixtures of independent Gaussians over $\mathcal{L}'$ with parameter s,*

$$\Pr[\exists i \in [1, m] \text{ such that } 0 < \|\mathbf{X}_i\|^2 < 4T] \geq 1/10 \,,$$

where $T := \frac{1}{m}\sum_{i=1}^{m} \mathbb{E}[\|\mathbf{X}_i\|^2]$.

Proof. By Markov's inequality, we have

$$\Pr\left[\sum_{i=1}^{m} \|\mathbf{X}_i\|^2 \geq 2mT\right] \leq \frac{1}{2} \,.$$

Hence, with probability at least $\frac{1}{2}$, we have $\sum_{i=1}^{m} \|\mathbf{X}_i\|^2 < 2mT$.

We next note that many of the $\mathbf{X}_i$ must be non-zero with high probability. Let $Y_1, \ldots, Y_m \in \{0, 1\}$ such that $Y_i = 0$ if and only if $\mathbf{X}_i = \mathbf{0}$. By Lemma 4.3,

$$\mathbb{E}[Y_i \mid Y_1 = y_1, \ldots, Y_{i-1} = y_{i-1}] \geq 4/5$$

for any $y_1, \ldots, y_{i-1} \in \{0, 1\}$. By Corollary 2.3, we have that

$$\Pr[Y_1 + \cdots + Y_m \leq 3m/5] \leq e^{-m/100} \leq 1/e \,.$$

Finally, by union bound, we see that with probability at least $1 - 1/e - 1/2 > 1/10$ the average squared norm will be at most $2T$ and more than half of the $\mathbf{X}_i$ will be non-zero. It follows from another application of Markov's inequality that at least one of the non-zero $\mathbf{X}_i$ must have squared norm less than $4T$. □

4.2 Summing vectors

Our algorithm will start with vectors $\mathbf{X}_1, \ldots, \mathbf{X}_m \in \mathcal{L}_0$, where $\mathcal{L}_0 \subset \mathcal{L}$ is some very dense superlattice of the input lattice $\mathcal{L}$. It then takes sums $\mathbf{Y}_k = \mathbf{X}_i + \mathbf{X}_j$ of pairs of these in such a way that the resulting $\mathbf{Y}_k$ lie in some appropriate sublattice $\mathcal{L}_1 \subset \mathcal{L}_0$, i.e., $\mathbf{Y}_k \in \mathcal{L}_1$. It does this repeatedly, finding vectors in $\mathcal{L}_2, \mathcal{L}_3, \ldots, \mathcal{L}_\ell$ until finally it obtains vectors in $\mathcal{L}_\ell := \mathcal{L}$.

Here, we study a single step of this algorithm, as shown below.

Algorithm 2 One step of the algorithm.

Input: Lattices $\mathcal{L}_0, \mathcal{L}_1 \subset \mathbb{R}^n$ with $2\mathcal{L}_0 \subseteq \mathcal{L}_1 \subseteq \mathcal{L}_0$, and lattice vectors $\mathbf{X}_1, \ldots, \mathbf{X}_m \in \mathcal{L}_0$ with $m \geq 2|\mathcal{L}_0/\mathcal{L}_1|$.

Output: Lattice vectors $\mathbf{Y}_1, \ldots, \mathbf{Y}_M \in \mathcal{L}_1$, with $M := \lceil (m - |\mathcal{L}_0/\mathcal{L}_1|)/2 \rceil$.

Set $\mathsf{USED}_i :=$ **false** for $i = 1, \ldots, m$, $k = 1$, and $i = 1$.
while $k \leq M$ **do**
  if not USED_i **and** $(\exists j \in \{1, \ldots, m\} \setminus \{i\}$ such that $\mathbf{X}_j \equiv \mathbf{X}_i \bmod \mathcal{L}_1$ **and** $\mathsf{USED}_j =$ **false**$)$ **then**
    Let $j \neq i$ be minimal such that $\mathbf{X}_j \equiv \mathbf{X}_i \bmod \mathcal{L}_1$ **and** $\mathsf{USED}_j =$ **false**.
    Set $\mathbf{Y}_k = \mathbf{X}_i + \mathbf{X}_j$.
    Set $\mathsf{USED}_i = \mathsf{USED}_j =$ **true** and increment k.
  end if
  Increment i.
end while
return $\mathbf{Y}_1, \ldots, \mathbf{Y}_M$

Notice that Algorithm 2 can be implemented in time $m \cdot \mathrm{poly}(n, \log m)$. This can be done, e.g., by creating a table of the $\mathbf{X}_i$ sorted according to $\mathbf{X}_i \bmod \mathcal{L}_1$. Then, for each i, such a j can be found (if it exists) by performing binary search on the table. Furthermore, the algorithm is guaranteed to find $M = \lceil (m - |\mathcal{L}_0/\mathcal{L}_1|)/2 \rceil$ output vectors because at most $|\mathcal{L}_0/\mathcal{L}_1|$ of the input vectors can be unpaired.

The key property that we will need from Algorithm 2 is that for any (possibly unknown) sublattice $\mathcal{L}' \subseteq \mathcal{L}_1 \subseteq \mathcal{L}_0$, the algorithm maps mixtures of Gaussians over $\mathcal{L}'$ to mixtures of Gaussians over $\mathcal{L}'$, provided that the parameter s is larger than $\eta_\varepsilon(\mathcal{L}')$ by a factor of $\sqrt{2}$. In other words, as long as there exists some sublattice $\mathcal{L}' \subseteq \mathcal{L}_1$ such that $\eta_\varepsilon(\mathcal{L}') \lesssim s$, then the output of the algorithm will be a mixture of Gaussians. Indeed, this is more-or-less immediate from Lemma 2.14.

Lemma 4.5. *For any lattices $\mathcal{L}_0, \mathcal{L}_1, \mathcal{L}' \subset \mathbb{R}^n$ with $2\mathcal{L}_0 \subseteq \mathcal{L}_1 \subseteq \mathcal{L}_0$ and $\mathcal{L}' \subseteq \mathcal{L}_1$, $\varepsilon \in (0, 1/3)$, $\delta > 0$, and parameter $s \geq \sqrt{2}\eta_\varepsilon(\mathcal{L}')$, if the input vectors $\mathbf{X}_1, \ldots, \mathbf{X}_m \in \mathcal{L}_0$ are sampled from the distribution that is δ-similar to a mixture of independent Gaussians over $\mathcal{L}'$ with parameter s, then the output vectors $\mathbf{Y}_1, \ldots, \mathbf{Y}_M \in \mathcal{L}_1$ are $(2\delta + 3\varepsilon)$-similar to a mixture of independent Gaussians over $\mathcal{L}'$ with parameter $\sqrt{2}s$.*

Proof. For a list of cosets $\mathbf{d} := (\mathbf{c}_1, \ldots, \mathbf{c}_m) \in (\mathcal{L}_0/\mathcal{L}')^m$ such that $\Pr[\mathbf{X}_1 = \mathbf{c}_1 \bmod \mathcal{L}', \ldots, \mathbf{X}_m = \mathbf{c}_m \bmod \mathcal{L}']$ is non-zero, let $\mathbf{Y}_{\mathbf{d},1}, \ldots, \mathbf{Y}_{\mathbf{d},M}$ be the ran-

dom variables obtained by taking $\mathbf{Y}_1, \ldots, \mathbf{Y}_M$ conditioned on $\mathbf{X}_i \equiv \mathbf{c}_i \bmod \mathcal{L}'$ for all i. We similarly define $\mathbf{X}_{\mathbf{d},i}$. Notice that $\mathbf{Y}_1, \ldots, \mathbf{Y}_M$ is a convex combination of random variables of the form $\mathbf{Y}_{\mathbf{d},1}, \ldots, \mathbf{Y}_{\mathbf{d},M}$, and that the property of being close to a mixture of independent Gaussians is preserved by taking convex combinations. Therefore, it suffices to prove the statement for $\mathbf{Y}_{\mathbf{d},1}, \ldots, \mathbf{Y}_{\mathbf{d},M}$ for all fixed $\mathbf{d}$.

To that end, fix $k \in \{1, \ldots, M\}$ and such a $\mathbf{d} \in (\mathcal{L}_0/\mathcal{L}')^m$. Notice that $\mathbf{X}_{\mathbf{d},i} \in \mathcal{L}' + \mathbf{c}_i \subseteq \mathcal{L}_1 + \mathbf{c}_i$. Therefore, there exist fixed i, j such that $\mathbf{Y}_{\mathbf{d},k} = \mathbf{X}_{\mathbf{d},i} + \mathbf{X}_{\mathbf{d},j}$. Furthermore, by assumption, for any $\mathbf{w} \in \mathcal{L}_0^{m-1}$ and $\mathbf{x} \in \mathcal{L}_0$,

$$\Pr[\mathbf{X}_{\mathbf{d},i} = \mathbf{x} \mid \mathbf{X}_{\mathbf{d},-i} = \mathbf{w}] = e^{\pm\delta} \frac{\rho_s(\mathbf{x})}{\rho_s(\mathcal{L}' + \mathbf{c}_i)} ,$$

and likewise for j. It follows from Lemma 2.14 that for any $\mathbf{y} \in \mathcal{L}_1$ and $\mathbf{z} \in \mathcal{L}_1^{M-1}$,

$$\Pr[\mathbf{X}_{\mathbf{d},i} + \mathbf{X}_{\mathbf{d}_j} = \mathbf{y} \mid \mathbf{Y}_{\mathbf{d},-k} = \mathbf{z}] = e^{\pm(2\delta+3\varepsilon)} \frac{\rho_{\sqrt{2}s}(\mathbf{y})}{\rho_{\sqrt{2}s}(\mathcal{L}' + \mathbf{c}_i + \mathbf{c}_j)} ,$$

as needed. □

Lemma 4.6. *For any lattices $\mathcal{L}_0, \mathcal{L}_1 \subset \mathbb{R}^n$ with $2\mathcal{L}_0 \subseteq \mathcal{L}_1 \subseteq \mathcal{L}_0$, if the input vectors $\mathbf{X}_1, \ldots, \mathbf{X}_m \in \mathcal{L}_0$ are sampled from a symmetric distribution, then the distribution of the output vectors $\mathbf{Y}_1, \ldots, \mathbf{Y}_M$ will also be symmetric. Furthermore,*

$$\sum \mathbb{E}[\|\mathbf{Y}_k\|^2] \leq \sum \mathbb{E}[\|\mathbf{X}_i\|^2] .$$

Proof. Let $\mathbf{d} = (\mathbf{c}_1, \ldots, \mathbf{c}_m) \in (\mathcal{L}_0/\mathcal{L}_1)^m$ be a list of cosets such that with non-zero probability we have $\mathbf{X}_1 \in \mathcal{L}_1 + \mathbf{c}_1, \ldots, \mathbf{X}_m \in \mathcal{L}_1 + \mathbf{c}_m$. Let $\mathbf{X}_{\mathbf{d},1}, \ldots, \mathbf{X}_{\mathbf{d},m}$ be the distribution obtained by sampling the $\mathbf{X}_i$ conditioned on this event, and let $\mathbf{Y}_{\mathbf{d},1}, \ldots, \mathbf{Y}_{\mathbf{d},M}$ be the corresponding output.

Notice that the distribution of $\mathbf{X}_{\mathbf{d},1}, \ldots, \mathbf{X}_{\mathbf{d},m}$ is also symmetric, since $\mathcal{L}_1 + \mathbf{c} = -(\mathcal{L}_1 + \mathbf{c})$ for any $\mathbf{c} \in \mathcal{L}_0/\mathcal{L}_1$. (Here, we have used the fact that $2\mathcal{L}_0 \subseteq \mathcal{L}_1 \subseteq \mathcal{L}_0$.)

And, for fixed $\mathbf{d}$ and $k \in \{1, \ldots, M\}$ there exist fixed (distinct) $i, j \in \{1, \ldots, m\}$ such that

$$\mathbf{Y}_{\mathbf{d},k} = \mathbf{X}_{\mathbf{d},i} + \mathbf{X}_{\mathbf{d},j} .$$

But, since the $\mathbf{X}_{\mathbf{d},1}, \ldots, \mathbf{X}_{\mathbf{d},m}$ are distributed symmetrically, we see immediately that for any $\mathbf{y} \in \mathcal{L}_1$ and $\mathbf{w} \in \mathcal{L}_1^{M-1}$,

$$\Pr[\mathbf{Y}_{\mathbf{d},k} = \mathbf{y} \mid \mathbf{Y}_{\mathbf{d},-k} = \mathbf{w}] = \Pr[\mathbf{Y}_{\mathbf{d},k} = -\mathbf{y} \mid \mathbf{Y}_{\mathbf{d},-k} = \mathbf{w}] .$$

In other words, the distribution of $\mathbf{Y}_{\mathbf{d},1}, \ldots, \mathbf{Y}_{\mathbf{d},M}$ is symmetric.

Furthermore, $\mathbb{E}[\|\mathbf{X}_{\mathbf{d},i} + \mathbf{X}_{\mathbf{d},j}\|^2]$ is equal to

$$\mathbb{E}[\|\mathbf{X}_{\mathbf{d},i}\|^2] + \mathbb{E}[\|\mathbf{X}_{\mathbf{d},j}\|^2] + 2\,\mathbb{E}[\langle \mathbf{X}_i, \mathbf{X}_j \rangle] = \mathbb{E}[\|\mathbf{X}_{\mathbf{d},i}\|^2] + \mathbb{E}[\|\mathbf{X}_{\mathbf{d},j}\|^2] ,$$

where in the last step we have used the symmetry of $\mathbf{X}_{\mathbf{d},1}, \ldots, \mathbf{X}_{\mathbf{d},m}$. Since the $\mathbf{Y}_{\mathbf{d},k}$ are sums of disjoint pairs of the $\mathbf{X}_{\mathbf{d},i}$, it follows immediately that

$$\sum_{k=1}^{M} \mathbb{E}[\|\mathbf{Y}_{\mathbf{d},k}\|^2] \leq \sum_{i=1}^{m} \mathbb{E}[\|\mathbf{X}_{\mathbf{d},i}\|^2] \,.$$

The results for $\mathbf{X}_1, \ldots, \mathbf{X}_m, \mathbf{Y}_1, \ldots, \mathbf{Y}_M$ then follow immediately from the fact that this distribution can be written as a convex combination of the vectors $\mathbf{X}_{\mathbf{d},1}, \ldots, \mathbf{X}_{\mathbf{d},m}, \mathbf{Y}_{\mathbf{d},1}, \ldots, \mathbf{Y}_{\mathbf{d},M}$ for different coset lists $\mathbf{d} \in (\mathcal{L}_0/\mathcal{L}_1)^m$, since both symmetry and the inequality on expectations are preserved by convex combinations. □

4.3 A tower of lattices

We will repeatedly apply Algorithm 2 on a "tower" of lattices similar to [ADRS15]. We use (a slight modification of) the definition and construction of the tower of lattices from [ADRS15].

Definition 4.7 ([ADRS15])**.** *For an integer α satisfying $n/2 \leq \alpha \leq n$, we say that $(\mathcal{L}_0, \ldots, \mathcal{L}_\ell)$ is a tower of lattices in $\mathbb{R}^n$ of index 2^α if for all i we have $2\mathcal{L}_{i-1} \subseteq \mathcal{L}_i \subset \mathcal{L}_{i-1}, \mathcal{L}_i/2 \subseteq \mathcal{L}_{i-2}$, $|\mathcal{L}_{i-1}/\mathcal{L}_i| = 2^\alpha$, and $2^{\lceil i\alpha/n \rceil}\mathcal{L}_0 \subseteq \mathcal{L}_i \subseteq 2^{\lfloor i\alpha/n \rfloor}\mathcal{L}_0$ for all i.*

Theorem 4.8 ([ADRS15])**.** *There is a polynomial-time algorithm that takes as input integers $\ell \geq 1$ and $n/2 \leq \alpha \leq n$ as well as a lattice $\mathcal{L} \subseteq \mathbb{R}^n$ and outputs a tower of lattice $(\mathcal{L}_0, \ldots, \mathcal{L}_\ell)$ with $\mathcal{L}_\ell = \mathcal{L}$.*

Proof. We give the construction below. The desired properties are immediate from the construction. Let $\mathbf{b}_1, \ldots, \mathbf{b}_n$ be a basis of $\mathcal{L}$. The tower is then defined by "cyclically halving α coordinates", namely,

$$\begin{aligned}
\mathcal{L}_\ell &= \mathcal{L}(\mathbf{b}_1, \ldots, \mathbf{b}_n), \\
\mathcal{L}_{\ell-1} &= \mathcal{L}(\mathbf{b}_1/2, \ldots, \mathbf{b}_\alpha/2, \mathbf{b}_{\alpha+1}, \ldots \mathbf{b}_n), \\
\mathcal{L}_{\ell-2} &= \mathcal{L}(\mathbf{b}_1/4, \ldots, \mathbf{b}_{2\alpha-n}/4, \mathbf{b}_{2\alpha-n+1}/2, \ldots \mathbf{b}_n/2),
\end{aligned}$$

etc. The required properties can be easily verified. □

The following proposition shows that starting with discrete Gaussian samples from $\mathcal{L}_0$ and then repeatedly applying Algorithm 2 gives us a list of vectors in $\mathcal{L}_\ell$ that is close to a mixture of Gaussians, provided that there exists an appropriate "smooth sublattice" $\mathcal{L}' \subseteq \mathcal{L}_0$.

Proposition 4.9. *There is an algorithm that runs in $m \cdot \mathrm{poly}(n, \ell, \log m)$ time; takes as input a tower of lattices $(\mathcal{L}_0, \ldots, \mathcal{L}_\ell)$ in $\mathbb{R}^n$ of index 2^α, and vectors $\mathbf{X}_1, \ldots, \mathbf{X}_m \in \mathcal{L}_0$ with $m := 2^{\ell+\alpha+1}$; and outputs $\mathbf{Y}_1, \ldots, \mathbf{Y}_M \in \mathcal{L}_\ell$ with $M := 2^\alpha$ with the following properties. If the input vectors $\mathbf{X}_1, \ldots, \mathbf{X}_m$ are symmetric and 0-similar to a mixture of Gaussians over $\mathcal{L}' \subseteq \mathcal{L}_0$ with parameter $s >$*

$10 \cdot 2^{(\alpha/n-1/2)\ell}\eta_\varepsilon(\mathcal{L}')$ *for some (possibly unknown) sublattice* $\mathcal{L}' \subseteq \mathcal{L}_0$ *and* $\varepsilon \in (0, 1/3)$*; then the output distribution is* $(10^\ell\varepsilon)$*-similar to a mixture of independent Gaussians over* $2^{\lceil \ell\alpha/n \rceil}\mathcal{L}' \subseteq \mathcal{L}_\ell$ *with parameter* $2^{\ell/2}s$*, and*

$$\sum_{k=1}^{M} \mathbb{E}[\|\mathbf{Y}_k\|^2] \le \sum_{i=1}^{m} \mathbb{E}[\|\mathbf{X}_i\|^2] \,.$$

Proof. The algorithm simply applies Algorithm 2 repeatedly, first using the input vectors in $\mathcal{L}_0$ to obtain vectors in $\mathcal{L}_1$, then using these to obtain vectors in $\mathcal{L}_2$, etc., until eventually it obtains vectors $\mathbf{Y}_1, \ldots, \mathbf{Y}_M \in \mathcal{L}_\ell$. The running time is clearly $m \cdot \mathrm{poly}(n, \ell, \log m)$, as claimed.

By Lemma 4.6 and a simple induction argument, we see that every call to Algorithm 2 results in a symmetric distribution, and the sum of the expected squared norms is non-increasing after each step. In particular,

$$\sum_{k=1}^{M} \mathbb{E}[\|\mathbf{Y}_k\|^2] \le \sum_{i=1}^{m} \mathbb{E}[\|\mathbf{X}_i\|^2] \,,$$

as needed.

We suppose for induction that the distribution of the output of the ith call to Algorithm 2 is $10^i\varepsilon$-similar to a mixture of independent Gaussians over $2^{\lceil i\alpha/n \rceil}\mathcal{L}' \subseteq 2^{\lceil i\alpha/n \rceil}\mathcal{L}_0 \subseteq \mathcal{L}_i$ with parameter $2^{i/2}s$ (which is true by assumption for $i = 0$). Then, this distribution is also $10^i\varepsilon$-similar to a mixture of independent Gaussians over $2^{\lceil (i+1)\alpha/n \rceil}\mathcal{L}' \subseteq 2^{\lceil i\alpha/n \rceil}\mathcal{L}'$ (since a mixture of Gaussians over a lattice is also a mixture of Gaussians over any sublattice). Furthermore, $\eta_\varepsilon(2^{\lceil (i+1)\alpha/n \rceil}\mathcal{L}') = 2^{\lceil (i+1)\alpha/n \rceil}\eta_\varepsilon(\mathcal{L}') < 2^{i/2}s/\sqrt{2}$. Therefore, we may apply Lemma 4.5 to conclude that the distribution of the output of the $(i+1)$st call to Algorithm 2 is $10^{i+1}\varepsilon$-similar to a mixture of independent Gaussians over $2^{\lceil (i+1)\alpha/n \rceil}\mathcal{L}' \subseteq \mathcal{L}_{i+1}$ with parameter $2^{(i+1)/2}s$. In particular, the final output vectors are $10^\ell\varepsilon$-similar to a mixture of independent Gaussians over $2^{\lceil \ell\alpha/n \rceil}\mathcal{L}'$, as needed. □

4.4 The algorithm

Theorem 4.10. *For any* $\varepsilon = \varepsilon(n) \in (0, n^{-200})$*, there is a* $2^{n/2+O(n\log(n)/\log(1/\varepsilon))+o(n)}$*-time algorithm that solves* $(100\sqrt{n}\overline{\eta}_\varepsilon)$*-GSVP. In particular, if* $\varepsilon = n^{-\omega(1)}$*, then the running time is* $2^{n/2+o(n)}$*.*

Proof. The algorithm takes as input a (basis for a) lattice $\mathcal{L} \subset \mathbb{R}^n$ with $n \ge 50$ and behaves as follows. Without loss of generality, we may assume that $\varepsilon > 2^{-n}$ and that the algorithm has access to a parameter $s > 0$ with $50\overline{\eta}_\varepsilon(\mathcal{L}) \le s \le 100\overline{\eta}_\varepsilon(\mathcal{L})$. Let $\ell := \lfloor \log(1/\varepsilon)/\log(10) \rfloor - 1$ and $\alpha := \lceil n/2 + 100n\log n/\log(1/\varepsilon) \rceil$.

The algorithm first runs the procedure from Theorem 4.8 on input ℓ, α, and $\mathcal{L}$, receiving as output a tower of lattices $(\mathcal{L}_0, \ldots, \mathcal{L}_\ell)$ with $\mathcal{L}_\ell = \mathcal{L}$. The algorithm then runs the procedure from Proposition 3.5 on input $\mathcal{L}_0$, $r := n/5$, $m := 2^{\ell+\alpha+1}$, and parameter $s' := 2^{-\ell/2}s$, receiving as output a sublattice

$\widehat{\mathcal{L}} \subseteq \mathcal{L}_0$, and vectors $\mathbf{X}_1, \ldots, \mathbf{X}_m \in \widehat{\mathcal{L}} \subseteq \mathcal{L}_0$. Finally, the algorithm runs the procedure from Proposition 4.9 on input $(\mathcal{L}_0, \ldots, \mathcal{L}_\ell)$ and $\mathbf{X}_1, \ldots, \mathbf{X}_m$, receiving as output $\mathbf{Y}_1, \ldots, \mathbf{Y}_M \in \mathcal{L}_\ell = \mathcal{L}$. It then simply outputs the shortest non-zero vector amongst the $\mathbf{Y}_i \in \mathcal{L}$. (If all of the $\mathbf{Y}_i$ are zero, the algorithm fails.)

The running time of the algorithm is clearly $(m + 2^{r+o(r)}) \cdot \mathrm{poly}(n, \ell, \log m) = 2^{n/2+O(n \log n/\log(1/\varepsilon))+o(n)}$. We first show that the promise $s' \geq r^{n/r}\sqrt{n \log n} \cdot \overline{\eta}_\varepsilon(\mathcal{L}_0)$ needed to apply Proposition 3.5 is satisfied. Indeed, by the definition of a tower of lattices, we have $\mathcal{L} \subseteq 2^{\lfloor \ell\alpha/n \rfloor}\mathcal{L}_0$, so that

$$s' \geq 50 \cdot 2^{-\ell/2} \cdot \overline{\eta}_\varepsilon(\mathcal{L}) \geq 50 \cdot 2^{\lfloor \ell\alpha/n \rfloor - \ell/2} \cdot \overline{\eta}_\varepsilon(\mathcal{L}_0) \geq r^{n/r}\sqrt{n \log n} \cdot \overline{\eta}_\varepsilon(\mathcal{L}_0) \ ,$$

as needed. Therefore, the procedure from Proposition 3.5 succeeds, i.e. we have $\overline{\eta}_\varepsilon(\widehat{\mathcal{L}}) = \overline{\eta}_\varepsilon(\mathcal{L}_0)$ and that the $\mathbf{X}_i$ are distributed as independent samples from $D_{\widehat{\mathcal{L}},s'}$.

In particular, let $\mathcal{L}' \subseteq \widehat{\mathcal{L}} \subseteq \mathcal{L}_0$ such that $\eta_\varepsilon(\mathcal{L}') = \overline{\eta}_\varepsilon(\widehat{\mathcal{L}}) = \overline{\eta}_\varepsilon(\mathcal{L}_0)$. Then, the distribution of $\mathbf{X}_1, \ldots, \mathbf{X}_m$ is symmetric and 0-similar to a mixture of Gaussians over $\mathcal{L}'$ with parameter $s' > 10 \cdot 2^{(\alpha/n-1/2)\ell}\eta_\varepsilon(\mathcal{L}')$. We may therefore apply Proposition 4.9 and see that the $\mathbf{Y}_1, \ldots, \mathbf{Y}_M \in \mathcal{L}$ are δ-similar to a mixture of independent Gaussians over $2^{\lceil \ell\alpha/n \rceil}\mathcal{L}'$ with parameter s and $\delta := 10^\ell\varepsilon \leq 1/10$. Furthermore,

$$\sum_{k=1}^{M} \mathbb{E}[\|\mathbf{Y}_k\|^2] \leq \sum_{i=1}^{m} \mathbb{E}[\|\mathbf{X}_i\|^2] \leq \frac{nm(s')^2}{2\pi} = 2^{-\ell} \cdot \frac{nms^2}{2\pi} \ ,$$

where the last inequality is Claim 2.11.

Finally, we notice that

$$\begin{aligned} s &\geq 50\overline{\eta}_\varepsilon(\mathcal{L}) \geq 50 \cdot 2^{\lfloor \ell\alpha/n \rfloor}\overline{\eta}_\varepsilon(\mathcal{L}_0) = 50\eta_\varepsilon(2^{\lfloor \ell\alpha/n \rfloor}\mathcal{L}') \geq 25\eta_\varepsilon(2^{\lceil \ell\alpha/n \rceil}\mathcal{L}') \\ &\geq 10e^\delta\eta_{1/2}((2^{\lceil \ell\alpha/n \rceil}\mathcal{L}') \ . \end{aligned}$$

Therefore, we may apply Corollary 4.4 to $\mathbf{Y}_1, \ldots, \mathbf{Y}_M$ to conclude that with probability at least $1/10$, there exists $k \in \{1, \ldots, M\}$ such that

$$0 < \|\mathbf{Y}_k\|^2 < \frac{4}{M} \cdot \sum_{i=1}^{M} \mathbb{E}[\|\mathbf{Y}_i\|^2] \leq 2^{-\ell} \cdot \frac{nms^2}{2\pi M} \leq ns^2 \leq 100^2 n\overline{\eta}_\varepsilon(\mathcal{L})^2 \ .$$

In other words, $\mathbf{Y}_k \in \mathcal{L}$ is a valid solution to $(100\sqrt{n}\overline{\eta}_\varepsilon)$-GSVP, as needed. □

Corollary 4.11. *There is a $2^{n/2+o(n)}$-time algorithm that solves γ-SVP for any $\gamma = \gamma(n) > \omega(\sqrt{n \log n})$.*

Proof. Theorem 4.10 gives an algorithm with the desired running time that finds a non-zero lattice vector with norm bounded by $100\sqrt{n}\overline{\eta}_\varepsilon(\mathcal{L})$ for

$$\varepsilon := 2^{-\gamma^2/(100^2 n)} < n^{-\omega(1)} \ .$$

The result follows from Lemma 3.1, which in particular tells us that

$$\overline{\eta}_\varepsilon(\mathcal{L}) \leq \sqrt{\log(1/\varepsilon)}\lambda_1(\mathcal{L}) \leq \gamma/(100\sqrt{n}) \cdot \lambda_1(\mathcal{L}) \ ,$$

as needed. □

Corollary 4.12. *There is a $2^{n/2+o(n)}$-time algorithm that solves γ-HSVP for any $\gamma = \gamma(n) > \omega(\sqrt{n \log^3 n})$.*

Proof. Theorem 4.10 gives an algorithm with the desired running time that finds a non-zero lattice vector with norm bounded by $100\sqrt{n}\overline{\eta}_\varepsilon(\mathcal{L})$ for

$$\varepsilon := 2^{-\gamma^2/(10^{10} n \log^2 n)} < n^{-\omega(1)} .$$

The result follows from Lemma 3.1, which in particular tells us that

$$\overline{\eta}_\varepsilon(\mathcal{L}) \leq 10\sqrt{\log(1/\varepsilon)}(\log n + 2)\det(\mathcal{L})^{1/n} \leq \gamma/(100\sqrt{n}) \cdot \det(\mathcal{L})^{1/n} ,$$

as needed (where we have assumed that n is sufficiently large). □

5 Approximate SVP via Basis Reduction

Basis reduction algorithms solve δ-(H)SVP in dimension n by making polynomially many calls to a δ'-SVP algorithm on lattices in dimension $k < n$. We will show in this section how to modify the basis reduction algorithm from [GN08, ALNS20] to prove Theorem 1.2.

5.1 Slide-reduced bases

Here, we introduce our notion of a reduced basis. This differs from prior work in that we allow the length ℓ of the last block to be not equal to k, and we use HSVP reduction where other works use SVP reduction. E.g., taking $\ell = k$ and replacing (D)HSVP reduction with (D)SVP reduction in Item 2 recovers the definition from [ALNS20]. (Taking $\ell = k$ and $q = 0$ and replacing all (D)HSVP reduction with (D)SVP reduction recovers the original definition in [GN08].)

Definition 5.1 (Slide reduction)**.** *Let n, k, p, q, ℓ be integers such that $n = pk + q + \ell$ with $p \geq 1, k, \ell \geq 2$ and $0 \leq q \leq k - 1$. Let $\delta_H \geq 1$ and $\delta_S \geq 1$. A basis $\mathbf{B} \in \mathbb{R}^{m \times n}$ is $(\delta_H, k, \delta_S, \ell)$-slide-reduced if it is size-reduced and satisfies the following four sets of constraints.*

1. *The block $\mathbf{B}_{[1,k+q+1]}$ is η-twin-reduced for $\eta := \delta_H^{\frac{k+q-1}{k-1}}$.*
2. *For all $i \in [1, p-1]$, the block $\mathbf{B}_{[ik+q+1,(i+1)k+q+1]}$ is δ_H-twin-reduced.*
3. *The block $\mathbf{B}_{[pk+q+1,n]}$ is δ_S-SVP-reduced.*

Theorem 5.2. *For any $\delta_H, \delta_S \geq 1, k \geq 2, \ell \geq 2$, if $\mathbf{B} \in \mathbb{R}^{n \times n}$ is a $(\delta_H, k, \delta_S, \ell)$-slide-reduced basis of a lattice $\mathcal{L}$ with $\lambda_1(\mathcal{L}(\mathbf{B}_{[1,n-\ell]})) > \lambda_1(\mathcal{L})$ then*

$$\|\mathbf{b}_1\| \leq \delta_S(\delta_H^2)^{\frac{n-\ell}{k-1}}\lambda_1(\mathcal{L}) .$$

Proof. By Fact 2.22, $\|\mathbf{b}_1\| \leq \eta^{\frac{2(k+q)}{k+q-1}}\|\mathbf{b}^*_{k+q+1}\| = \delta_H^{\frac{2(k+q)}{k-1}}\|\mathbf{b}^*_{k+q+1}\|$. Also, for all $i \in [1, p-1]$, $\|\mathbf{b}^*_{ik+q+1}\| \leq \delta_H^{\frac{2k}{k-1}}\|\mathbf{b}^*_{(i+1)k+q+1}\|$. All together we have:

$$\|\mathbf{b}_1\| \leq (\delta_H^2)^{\frac{k+q+(p-1)k}{k-1}}\|\mathbf{b}^*_{pk+q+1}\| = (\delta_H^2)^{\frac{n-\ell}{k-1}}\|\mathbf{b}^*_{pk+q+1}\|$$

Lastly, since $\lambda_1(\mathcal{L}(\mathbf{B}_{[1,n-\ell]})) > \lambda_1(\mathcal{L})$, $\|\mathbf{b}^*_{pk+q+1}\| \leq \delta_S\lambda_1(\mathcal{L}(\mathbf{B}_{[pk+q+1,n]})) \leq \delta_S\lambda_1(\mathcal{L})$. The result does follow. □

5.2 The slide reduction algorithm

We show our algorithm for generating a slide-reduced basis. We stress that this is essentially the same algorithm as in [ALNS20] (which itself is a generalization of the algorithm in [GN08]) with a slight modification that allows the last block to have arbitrary length ℓ. Our proof for bounding the running time of the algorithm is therefore essentially identical to the proof in [GN08, ALNS20].

Algorithm 3 Our slide-reduction algorithm

Input: Block size $k \geq 2$, slack $\varepsilon > 0$, approximation factor $\delta_H, \delta_S \geq 1$, basis $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n) \in \mathbb{Z}^{m \times n}$ of a lattice $\mathcal{L}$ of rank $n = pk + q + \ell$ for $0 \leq q \leq k - 1$, and access to a rank k δ_H-HSVP oracle and a rank ℓ δ_S-SVP oracle.

Output: A $((1+\varepsilon)\delta_H, k, \delta_S, \ell)$-slide-reduced basis of $\mathcal{L}(\mathbf{B})$.

while $\mathrm{vol}(\mathbf{B}_{[1,ik+q]})^2$ is modified by the loop for some $i \in [1, p]$ **do**
  $(1+\varepsilon)\eta$-HSVP-reduce $\mathbf{B}_{[1,k+q]}$ using Alg. 1 for $\eta := (\delta_H)^{\frac{k+q-1}{k-1}}$.
  for $i = 1$ **to** $p - 1$ **do**
    δ_H-HSVP-reduce $\mathbf{B}_{[ik+q+1,(i+1)k+q]}$.
  end for
  δ_S-SVP-reduce $\mathbf{B}_{[pk+q+1,n]}$.
  if $\mathbf{B}_{[2,k+q+1]}$ is not $(1+\varepsilon)\eta$-DHSVP-reduced **then**
    $(1+\varepsilon)^{1/2}\eta$-DHSVP-reduce $\mathbf{B}_{[2,k+q+1]}$ using Alg. 1.
  end if
  for $i = 1$ **to** $p - 1$ **do**
    Find a new basis $\mathbf{C} := (\mathbf{b}_1, \ldots, \mathbf{b}_{ik+q+1}, \mathbf{c}_{ik+q+2}, \ldots, \mathbf{c}_{(i+1)k+q+1}, \mathbf{b}_{ik+q+2}, \ldots, \mathbf{b}_n)$ of $\mathcal{L}$ by δ_H-DHSVP-reducing $\mathbf{B}_{[ik+q+2,(i+1)k+q+1]}$.
    if $(1+\varepsilon)\|\mathbf{b}^*_{(i+1)k+q+1}\| < \|\mathbf{c}^*_{(i+1)k+q+1}\|$ **then**
      $\mathbf{B} \leftarrow \mathbf{C}$.
    end if
  end for
end while
return $\mathbf{B}$.

Theorem 5.3. *For $\varepsilon \in [1/\mathrm{poly}(n), 1]$, Algorithm 3 runs in polynomial time (excluding oracle calls), makes polynomially many calls to its δ_H-HSVP oracle and δ_S-SVP oracle, and outputs a $((1+\varepsilon)\delta_H, k, \delta_S, \ell)$-slide-reduced basis of the input lattice $\mathcal{L}$.*

Proof. First, notice that if Algorithm 3 ever terminates, the output must be $((1+\varepsilon)\delta_H, k, \delta_S, \ell)$-slide-reduced basis. It remains to show that the algorithm terminates in polynomially many steps (excluding oracle calls).

Let $\mathbf{B}_0 \in \mathbb{Z}^{m \times n}$ be the input basis and let $\mathbf{B} \in \mathbb{Z}^{m \times n}$ denote the current basis during the execution of Algorithm 3. Following the analysis of basis reduction algorithms in [LLL82, GN08, LN14, ALNS20], we consider an integral potential

$$P(\mathbf{B}) := \prod_{i=1}^{p} \mathrm{vol}(\mathbf{B}_{[1,ik+q]})^2 \in \mathbb{Z}^+.$$

At the beginning of the algorithm, the potential satisfies $\log P(\mathbf{B}_0) \leq 2n^2 \cdot \log \|\mathbf{B}_0\|$. For each of the primal steps (i.e., Steps 2, 4 and 6), the lattice $\mathcal{L}(\mathbf{B}_{[1,ik+q]})$ for any $i \geq 1$ is unchanged. Hence $P(\mathbf{B})$ does not change. On the other hand, the dual steps (i.e., Steps 8 and 13) either leave $\mathrm{vol}(\mathbf{B}_{[1,ik+q]})$ unchanged for all i or decrease $P(\mathbf{B})$ by a multiplicative factor of at least $(1+\varepsilon)$.

Therefore, there are at most $\log P(\mathbf{B}_0)/\log(1+\varepsilon)$ updates on $P(\mathbf{B})$ by Algorithm 3. This directly implies that the algorithm makes at most $4pn^2 \log \|\mathbf{B}_0\|/\log(1+\varepsilon)$ calls to the HSVP oracle, the SVP oracle, and Algorithm 1.

We then conclude that Algorithm 3's running time is bounded by some polynomial in the size of input (excluding the running time of oracle calls).

□

Corollary 5.4. *For any constant $c \geq 1$, there is a randomized algorithm that solves $(\mathrm{polylog}(n)n^c)$-SVP that runs in $2^{k/2+o(k)}$ time for $k := \frac{n-c}{c+5/(8.02)}$.*

Proof. Let $\ell = \frac{0.5k}{0.802}$ and run Algorithm 3, using the $O(\mathrm{polylog}(n)\sqrt{n})$-HSVP algorithm from Corollary 4.12 and the $O(1)$-SVP algorithm from [LWXZ11] as oracles. We receive a $((1+\varepsilon)\mathrm{polylog}(k)\sqrt{k}, k, O(1), \ell)$-slide-reduced basis $\mathbf{B}$ for any input lattice $\mathcal{L}$. Now consider two cases:

CASE 1: $\lambda_1(\mathcal{L}(\mathbf{B}_{[1,n-\ell]})) > \lambda_1(\mathcal{L})$: By Theorem 5.2, we conclude that

$$\|\mathbf{b}_1\| \leq \delta_S(\delta_H^2)^{\frac{n-\ell}{k-1}}\lambda_1(\mathcal{L}) \leq O(\mathrm{polylog}(k)^c n^c)\lambda_1(\mathcal{L}) \ ,$$

as desired.

CASE 2: $\lambda_1(\mathcal{L}(\mathbf{B}_{[1,n-\ell]})) = \lambda_1(\mathcal{L})$: Then we repeat the algorithm on the lattice $\mathcal{L}(\mathbf{B}_{[1,n-\ell]})$ with lower dimension. This can happen at most n/ℓ times, introducing at most a polynomial factor in the running time.

For the running time, the algorithm from Corollary 4.12 runs in time $2^{0.5k+o(k)}$. The algorithm from [LWXZ11] runs in time $2^{0.802\ell+o(\ell)}$, which is the same as $2^{0.5k+o(k)}$, by our choice of ℓ. This completes the proof. □

References

[ADRS15] Divesh Aggarwal, Daniel Dadush, Oded Regev, and Noah Stephens-Davidowitz. Solving the Shortest Vector Problem in 2^n time via Discrete Gaussian Sampling. In *STOC*, 2015. http://arxiv.org/abs/1412.7994. 1, 2, 3, 4, 5, 6, 7, 15, 23

[AKS01] Miklós Ajtai, Ravi Kumar, and D. Sivakumar. A sieve algorithm for the Shortest Lattice Vector Problem. In *STOC*, 2001. 2

[ALNS20] Divesh Aggarwal, Jianwei Li, Phong Q. Nguyen, and Noah Stephens-Davidowitz. Slide reduction, revisited–Filling the gaps in SVP approximation. In *CRYPTO*, 2020. 2, 4, 5, 14, 15, 16, 26, 27

[AS04] Noga Alon and Joel H Spencer. *The probabilistic method*. John Wiley & Sons, 2004. 9

[AS18] Divesh Aggarwal and Noah Stephens-Davidowitz. Just take the average! An embarrassingly simple 2^n-time algorithm for SVP (and CVP). In *SOSA*, 2018. http://arxiv.org/abs/1709.01535. 2, 3, 6

[AUV19] Divesh Aggarwal, Bogdan Ursu, and Serge Vaudenay. Faster sieving algorithm for approximate SVP with constant approximation factors. https://eprint.iacr.org/2019/1028, 2019. 2, 5

[BDGL16] Anja Becker, Léo Ducas, Nicolas Gama, and Thijs Laarhoven. New directions in nearest neighbor searching with applications to lattice sieving. In *SODA*, 2016. 2

[BGJ14] Anja Becker, Nicolas Gama, and Antoine Joux. A sieve algorithm based on overlattices. *LMS Journal of Computation and Mathematics*, 17(A):49–70, 2014. 6

[BLP+13] Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. Classical hardness of Learning with Errors. In *STOC*, 2013. 6, 15

[DR16] Daniel Dadush and Oded Regev. Towards strong reverse Minkowski-type inequalities for lattices. In *FOCS*, 2016. http://arxiv.org/abs/1606.06913. 1, 8, 17

[DRS14] Daniel Dadush, Oded Regev, and Noah Stephens-Davidowitz. On the Closest Vector Problem with a distance guarantee. In *CCC*, 2014. 11

[Duc18] Léo Ducas. Shortest vector from lattice sieving: A few dimensions for free. In *Eurocrypt*, 2018. 2

[GINX16] Nicolas Gama, Malika Izabachène, Phong Q. Nguyen, and Xiang Xie. Structural lattice reduction: Generalized worst-case to average-case reductions and homomorphic cryptosystems. In *Eurocrypt*, 2016. 6

[GMPW20] Nicholas Genise, Daniele Micciancio, Chris Peikert, and Michael Walter. Improved discrete Gaussian and subgaussian analysis for lattice cryptography. In *PKC*, 2020. https://eprint.iacr.org/2020/337. 13

[GN08] Nicolas Gama and Phong Q. Nguyen. Finding short lattice vectors within Mordell's inequality. In *STOC*, 2008. 1, 2, 4, 5, 14, 15, 16, 26, 27

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC*, 2008. https://eprint.iacr.org/2007/432. 6, 14

[Kan83] Ravi Kannan. Improved algorithms for integer programming and related lattice problems. In *STOC*, 1983. 2

[Laa15] Thijs Laarhoven. Sieving for shortest vectors in lattices using angular locality-sensitive hashing. In *CRYPTO*, 2015. 2

[LLL82] Arjen K. Lenstra, Hendrik W. Lenstra, Jr., and László Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261(4), 1982. 2, 15, 16, 27

[LN14] Jianwei Li and Phong Q. Nguyen. Approximating the densest sublattice from Rankin's inequality. *LMS J. of Computation and Mathematics*, 17(A), 2014. 16, 27

[Lov86] László Lovász. *An algorithmic theory of numbers, graphs and convexity*. Society for Industrial and Applied Mathematics, 1986. 4

[LWXZ11] Mingjie Liu, Xiaoyun Wang, Guangwu Xu, and Xuexin Zheng. Shortest lattice vectors in the presence of gaps. http://eprint.iacr.org/2011/139, 2011. 1, 2, 5, 28

[MP13] Daniele Micciancio and Chris Peikert. Hardness of SIS and LWE with small parameters. In *CRYPTO*, 2013. 7, 8

[MR07] Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on Gaussian measures. *SIAM Journal on Computing*, 37(1):267–302, 2007. 7, 11

[MV13] Daniele Micciancio and Panagiotis Voulgaris. A deterministic single exponential time algorithm for most lattice problems based on Voronoi cell computations. *SIAM J. on Computing*, 42(3), 2013. 2

[MW16] Daniele Micciancio and Michael Walter. Practical, predictable lattice basis reduction. In *Eurocrypt*, 2016. http://eprint.iacr.org/2015/1123. 2, 4, 15, 16

[NIS18] Computer Security Division NIST. Post-quantum cryptography. https://csrc.nist.gov/Projects/Post-Quantum-Cryptography, 2018. 2

[NV08] Phong Q. Nguyen and Thomas Vidick. Sieve algorithms for the Shortest Vector Problem are practical. *J. Mathematical Cryptology*, 2(2), 2008. 2

[Pei16] Chris Peikert. A decade of lattice cryptography. *Foundations and Trends in Theoretical Computer Science*, 10(4), 2016. 2

[PS09] Xavier Pujol and Damien Stehlé. Solving the Shortest Lattice Vector Problem in time $2^{2.465n}$, 2009. http://eprint.iacr.org/2009/605. 2

[Reg09] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM (JACM)*, 56(6):34, 2009. 11

[RS17] Oded Regev and Noah Stephens-Davidowitz. A reverse Minkowski theorem. In *STOC*, 2017. 1, 8, 17

[Sch87] Claus-Peter Schnorr. A hierarchy of polynomial time lattice basis reduction algorithms. *Theor. Comput. Sci.*, 53(23), 1987. 2

[SE94] Claus-Peter Schnorr and M. Euchner. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. *Mathematical Programming*, 66, 1994. 2

[Ste17] Noah Stephens-Davidowitz. *On the Gaussian measure over lattices*. Phd thesis, New York University, 2017. 3

[WLW15] Wei Wei, Mingjie Liu, and Xiaoyun Wang. Finding shortest lattice vectors in the presence of gaps. In *CT-RSA*, 2015. 2, 5