2015 IEEE 56th Annual Symposium on Foundations of Computer Science

Solving the Closest Vector Problem in 2ˆn Time— The Discrete Gaussian Strikes Again!

Divesh Aggarwal*, Daniel Dadush†, and Noah Stephens-Davidowitz‡
*Department of Computer Science, EPFL
Email: Divesh.Aggarwal@epfl.ch
†Centrum Wiskunde & Informatica, Amsterdam
Email: dadush@cwi.nl
‡Courant Institute of Mathematical Sciences, New York University
Email: noahsd@cs.nyu.edu

Abstract

We give a $2^{n+o(n)}$-time and space randomized algorithm for solving the *exact* Closest Vector Problem (CVP) on n-dimensional Euclidean lattices. This improves on the previous fastest algorithm, the deterministic $\widetilde{O}(4^n)$-time and $\widetilde{O}(2^n)$-space algorithm of Micciancio and Voulgaris [1].

We achieve our main result in three steps. First, we show how to modify the sampling algorithm from [2] to solve the problem of discrete Gaussian sampling over *lattice shifts*, $\mathcal{L} - \mathbf{t}$, with very low parameters. While the actual algorithm is a natural generalization of [2], the analysis uses substantial new ideas. This yields a $2^{n+o(n)}$-time algorithm for approximate CVP with the very good approximation factor $\gamma = 1 + 2^{-o(n/\log n)}$. Second, we show that the approximate closest vectors to a target vector $\mathbf{t}$ can be grouped into "lower-dimensional clusters," and we use this to obtain a recursive reduction from exact CVP to a variant of approximate CVP that "behaves well with these clusters." Third, we show that our discrete Gaussian sampling algorithm can be used to solve this variant of approximate CVP.

The analysis depends crucially on some new properties of the discrete Gaussian distribution and approximate closest vectors, which might be of independent interest.

Index Terms

Discrete Gaussian; Closest Vector Problem; Lattice Problems.

I. Introduction

A lattice $\mathcal{L}$ is the set of all integer combinations of linearly independent vectors $\mathbf{b}_1, \ldots, \mathbf{b}_n \in \mathbb{R}^n$. The matrix $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ is called a basis of $\mathcal{L}$, and we write $\mathcal{L}(\mathbf{B})$ for the lattice generated by $\mathbf{B}$.

The two most important computational problems on lattices are the Shortest Vector Problem (SVP) and the Closest Vector Problem (CVP). Given a basis for a lattice $\mathcal{L} \subseteq \mathbb{R}^n$, SVP asks us to compute a non-zero vector in $\mathcal{L}$ of minimal length, and CVP asks us to compute a lattice vector nearest in Euclidean distance to a target vector $\mathbf{t}$.

Starting with the seminal work of [3], algorithms for solving these problems either exactly or approximately have been studied intensely. Such algorithms have found applications in factoring polynomials over rationals [3], integer programming [4], [5], [6], cryptanalysis [7], [8], [9], checking the solvability by radicals [10], and solving low-density subset-sum problems [11]. More recently, many powerful cryptographic primitives have been constructed whose security is based on the *worst-case* hardness of these or related lattice problems [12], [13], [14], [15], [16], [17], [18].

In their exact forms, both problems are known to be NP-complete, and they are even hard to approximate to within a factor of $n^{O(1/\log\log n)}$ under reasonable complexity assumptions [19], [20], [21], [22], [23], [24], [25], [26]. CVP is thought to be the "harder" of the two problems, as there is a simple reduction from SVP to CVP that preserves the dimension n of the lattice [27], even in the approximate case, while there is no known reduction in the other direction that preserves the dimension. Indeed, CVP is in some sense nearly "complete for lattice problems," as there are known dimension-preserving reductions from

0272-5428/15 $31.00 © 2015 IEEE
DOI 10.1109/FOCS.2015.41

nearly all important lattice problems to CVP, such as the Shortest Independent Vector Problem, Subspace Avoidance Problem, Generalized Closest Vector Problem, and the Successive Minima Problem [29], [30], [1]. (The Lattice Isomorphism Problem is an important exception.) None of these problems has a known dimension-preserving reduction to SVP.

Exact algorithms for CVP and SVP have a rich history. Kannan initiated their study with an enumeration-based $n^{O(n)}$-time algorithm for CVP [5], and many others improved upon his technique to improve the running time [31], [32], [33]. Since these algorithms solve CVP, they also imply solutions for SVP and all of the problems listed above. (Notably, these algorithms use only polynomial space.)

For over a decade, these $n^{O(n)}$-time algorithms remained the state of the art until, in a major breakthrough, Ajtai, Kumar, and Sivakumar (AKS) published the first $2^{O(n)}$-time algorithm for SVP [34]. The AKS algorithm is based on "randomized sieving," in which many randomly generated lattice vectors are iteratively combined to create successively shorter lattice vectors. The work of AKS led to two major questions: First, can CVP be solved in $2^{O(n)}$ time? And second, what is the best achievable constant in the exponent? Much work went into solving both of these problems using AKS's sieving technique [34], [35], [36], [37], [30], [38], [39], [40], culminating in a $\widetilde{O}(2^{2.456n})$-time algorithm for SVP and a $2^{O(n)}(1+1/\varepsilon)^{O(n)}$-time algorithm for $(1+\varepsilon)$-approximate CVP.

But, exact CVP is a much subtler problem than approximate CVP or exact SVP. In particular, for any approximation factor $\gamma > 1$, a target vector $\mathbf{t}$ can have arbitrarily many γ-approximate closest vectors in the lattice $\mathcal{L}$. For example, $\mathcal{L}$ might contain many vectors whose length is arbitrarily shorter than the distance between $\mathbf{t}$ and the lattice, so that any closest lattice vector is "surrounded by" many γ-approximate closest vectors. Randomized sieving algorithms for CVP effectively sample from a distribution that assigns weight to each lattice vector $\mathbf{y}$ according to some smooth function of $\|\mathbf{y}-\mathbf{t}\|$. Such algorithms face a fundamental barrier in solving exact CVP: they can "barely distinguish between" γ-approximate closest vectors and exact closest vectors for very small γ. (This problem does not arise when solving SVP because upper bounds on the lattice kissing number show that there *cannot* be arbitrarily many γ-approximate shortest lattice vectors. Indeed, such upper bounds play a crucial role in the analysis of sieving algorithms for exact SVP.)

So, the important question of whether CVP could be solved exactly in singly exponential time remained open until the landmark algorithm of Micciancio and Voulgaris [1] (MV), which built upon the approach of Sommer, Feder, and Shalvi [41]. MV showed a *deterministic* $\widetilde{O}(4^n)$-time and $\widetilde{O}(2^n)$-space algorithm for exact CVP. The MV algorithm uses the *Voronoi cell* of the lattice—the centrally symmetric polytope corresponding to the points closer to the origin than to any other lattice point. Until very recently, this algorithm had the best known asymptotic running time for *both* SVP and CVP. Prior to this work, this was the only known algorithm to solve CVP exactly in $2^{O(n)}$ time.

Very recently, Aggarwal, Dadush, Regev, and Stephens-Davidowitz (ADRS) gave a $2^{n+o(n)}$-time and space algorithm for SVP [2]. They accomplished this by giving an algorithm that solves the Discrete Gaussian Sampling problem (DGS) over a lattice $\mathcal{L}$. (As this is the starting point for our work, we describe their techniques in some detail below.) They also showed how to use their techniques to approximate CVP to within a factor of 1.97 in time $2^{n+o(n)}$, but like AKS a decade earlier, they left open a natural question: is there a corresponding algorithm for *exact* CVP (or even $(1+o(1))$-approximate CVP)?

A. Main contribution.

Our main result is a $2^{n+o(n)}$-time and space algorithm that solves CVP exactly via discrete Gaussian sampling. We achieve this in three steps. First, we show how to modify the ADRS sampling algorithm to solve DGS over *lattice shifts*, $\mathcal{L}-\mathbf{t}$. While the actual algorithm is a natural generalization of ADRS, the analysis uses substantial new ideas. This result alone immediately gives a $2^{n+o(n)}$-time algorithm to approximate CVP to within any approximation factor $\gamma = 1+2^{-o(n/\log n)}$. Second, we show that the approximate closest vectors to a target can be grouped into "lower-dimensional clusters." We use this to show a reduction from *exact* CVP to a variant of approximate CVP. Third, we show that our sampling algorithm actually solves this variant of approximate CVP, yielding a $2^{n+o(n)}$-time algorithm for *exact* CVP.

We find this result to be quite surprising as, in spite of much research in this area, all previous "truly randomized" algorithms only gave approximate solutions to CVP. Indeed, this barrier seemed inherent, as we described above. Our solution depends crucially on the large number of outputs from our sampling algorithm and new properties of the discrete Gaussian.

B. Our techniques

a) The ADRS algorithm for centered DGS and our generalization: The centered discrete Gaussian distribution over a lattice $\mathcal{L}$ with parameter $s > 0$, denoted $D_{\mathcal{L},s}$, is the probability distribution obtained by assigning to each vector $\mathbf{y} \in \mathcal{L}$ a probability proportional to its Gaussian mass, $\rho_s(\mathcal{L}) := e^{-\pi\|\mathbf{y}\|^2/s^2}$. As the parameter s becomes smaller, $D_{\mathcal{L},s}$ becomes more concentrated on the shorter vectors in the lattice. So, for a properly chosen parameter, a sample from $D_{\mathcal{L},s}$ is guaranteed to be a shortest lattice vector with not-too-small probability.

ADRS's primary contribution was an algorithm that solves DGS in the centered case, i.e., an algorithm that samples from $D_{\mathcal{L},s}$ for any s. To achieve this, they show how to build a discrete Gaussian "combiner," which takes samples from $D_{\mathcal{L},s}$ and converts them to samples from $D_{\mathcal{L},s/\sqrt{2}}$. The combiner is based on the simple but powerful observation that the average of two vectors sampled from $D_{\mathcal{L},s}$ is distributed exactly as $D_{\mathcal{L},s/\sqrt{2}}$, *provided that we condition on the result being in the lattice* [2, Lemma 3.4]. Note that the average of two lattice vectors is in the lattice if and only if they lie in the same *coset* of $2\mathcal{L}$. The ADRS algorithm therefore starts with many samples from $D_{\mathcal{L},s}$ for some very high s (which can be computed efficiently [42], [43], [17]) and repeatedly takes the average of carefully chosen pairs of vectors that lie in the same coset of $2\mathcal{L}$ to obtain samples from the discrete Gaussian with a much lower parameter.

The ADRS algorithm chooses which vectors to combine via rejection sampling applied to the cosets of $2\mathcal{L}$, and a key part of the analysis shows that this rejection sampling does not "throw out" too many vectors. In particular, ADRS show that, if a single run of the combiner starts with M samples from $D_{\mathcal{L},s}$, then the output will be $\beta(s)M$ samples from $D_{\mathcal{L},s/\sqrt{2}}$, where the "loss factor" $\beta(s)$ is equal to the ratio of the *collision probability* of $D_{\mathcal{L},s}$ mod $2\mathcal{L}$ divided by the maximal weight of a single coset (with some smaller factors that we ignore here for simplicity). It is not hard to check that for any probability distribution over 2^n elements, this loss factor is lower bounded by $2^{-n/2}$. This observation does not suffice, however, since the combiner must be run many times to solve SVP. Surprisingly ADRS show that the *total* loss factor $\beta(s)\beta(s/\sqrt{2})\cdots\beta(s/2^{-\ell/2})$ accumulated after running the combiner ℓ times is bounded by $2^{-n/2-o(n)}$. So, (ignoring small factors) their sampler returns at least $2^{-n/2}\cdot M$ samples from $D_{\mathcal{L},s/2^{-\ell/2}}$. The ADRS combiner requires $M \geq 2^n$ vectors "just to get started," so they obtain a $2^{n+o(n)}$-time algorithm for centered DGS that yields $2^{n/2}$ samples.

In this work, we show that some of the above analysis carries over easily to the more general case of shifted discrete Gaussians, $D_{\mathcal{L}-\mathbf{t},s}$ for $\mathbf{t} \in \mathbb{R}^n$—the distribution that assigns Gaussian weight $\rho_s(\mathbf{w})$ to each $\mathbf{w} \in \mathcal{L} - \mathbf{t}$. As in the centered case, the average of two vectors sampled from $D_{\mathcal{L}-\mathbf{t},s}$ is distributed exactly as $D_{\mathcal{L}-\mathbf{t},s/\sqrt{2}}$, *provided that we condition on the two vectors landing in the same coset of* $2\mathcal{L}$. (See Lemma 17 and Proposition 18.) We can therefore use essentially the same combiner as ADRS to obtain discrete Gaussian samples from the shifted discrete Gaussian with low parameters.

The primary technical challenge in this part of our work is to bound the accumulated loss factor $\beta(s)\beta(s/\sqrt{2})\cdots\beta(s/2^{-\ell/2})$. Using new techniques (based on the work of [44]), we show how to bound the accumulated loss factor in the shifted case by (ignoring small factors)

$$2^{-n} \cdot \frac{\rho_s(\mathcal{L}-\mathbf{t})}{\max_{\mathbf{c}\in\mathcal{L}/(2\mathcal{L})}\rho_s(\mathbf{c}-\mathbf{t})} \geq 2^{-n}\,. \tag{1}$$

So, we only need to start out with 2^n vectors to guarantee that our sampler will return at least one vector. (Like the ADRS algorithm, our algorithm requires at least 2^n vectors "just to get started.")

This is already sufficient to obtain a $2^{n+o(n)}$-time solution to approximate CVP for any approximation factor $\gamma = 1 + 2^{o(-n/\log n)}$. (See Corollary 23.) Below, we show that the loss factor in (1) is essentially exactly what we need to construct our *exact* CVP algorithm. In particular, we note that if we start with

$T \cdot 2^n$ vectors, then the number of output samples is

$$T \cdot \frac{\rho_s(\mathcal{L} - \mathbf{t})}{\max_{\mathbf{c} \in \mathcal{L}/(2\mathcal{L})} \rho_s(\mathbf{c} - \mathbf{t})} = \frac{T}{\max_{\mathbf{c} \in \mathcal{L}/(2\mathcal{L})} \Pr[D_{\mathcal{L}-\mathbf{t},s} \in \mathbf{c} - \mathbf{t}]} \,. \tag{2}$$

I.e., we essentially obtain just enough samples to "see each coset whose mass is within a factor T of the maximum."

b) A reduction from exact CVP to a variant of approximate CVP: In order to solve *exact* CVP, we first observe that the approximate closest vectors have some structure. Note that two *exact* closest lattice vectors to $\mathbf{t}$ cannot be in the same coset of $2\mathcal{L}$. If they were, then their average would be a closer lattice vector to $\mathbf{t}$, contradicting the assumption that the original vectors were as close as possible. A similar argument shows that the γ-approximate closest vectors to $\mathbf{t}$ can be grouped into 2^n "clusters" according to their coset mod $2\mathcal{L}$, where vectors in the same cluster must lie in a ball whose radius depends on the approximation factor γ. Indeed, if γ is low enough, then the clusters will be contained in *shifts* of a sublattice $\mathcal{L}'$ with dimension strictly less than n. If we could find a cluster that contains a closest lattice vector to $\mathbf{t}$, then we could solve CVP recursively "inside the corresponding shift of $\mathcal{L}'$," and we would have an exact CVP algorithm.

This observation (together with the fact that our sampling algorithm outputs enough vectors to "find every coset with relatively high Gaussian mass") suggests a recursive algorithm that repeatedly finds a cluster with a closest vector. In particular, we say that an algorithm solves γ-approximate neighbor CVP (γ-nCVP) if it outputs many solutions to γ-approximate CVP and at least one of them lies in the same coset mod $2\mathcal{L}$ of an *exact* closest vector $\mathbf{y}$. (Note that we do not require the algorithm to actually output $\mathbf{y}$; we simply ask that it finds a γ-approximate closest vector in the same coset as $\mathbf{y}$. See Definition 26.) We think of this point as a "neighbor" to $\mathbf{y}$ because, as we described above, it is necessarily very close to $\mathbf{y}$. Our basic idea is then to reduce exact CVP to γ-nCVP by the following recursive procedure: (1) solve γ-nCVP on the input lattice $\mathcal{L}$ and target $\mathbf{t}$; (2) for each coset mod $2\mathcal{L}$ contained in the output, solve CVP recursively over the shifted sublattice defined by the cluster of γ-approximate closest points in this coset; and then (3) output the closest resulting point to the target $\mathbf{t}$.

Correctness of this algorithm is actually quite straightforward. (See Lemma 27.) However, bounding the number of recursive calls is more difficult. We accomplish this via a technical lemma, which shows that we can always choose the parameters such that either (1) the number of clusters is at most 2^{n-d}, where d is the rank of the sublattice $\mathcal{L}'$; or (2) there are "slightly more" than 2^{n-d} clusters, but the dimension d of $\mathcal{L}'$ is "significantly less than" n. (See Lemma 29.) This will allow us to show that the total number of calls made on sublattices of rank d after a full run of the algorithm is at most $2^{n-d+o(n)}$. (See Theorem 30.) In particular, this shows that, in order to solve *exact* CVP in time $2^{n+o(n)}$, it suffices to find an algorithm that solves γ-nCVP for small γ that itself runs in time $2^{d+o(d)}$ on lattices of rank d.

c) Solving neighbor CVP: Our final task is to solve γ-nCVP for sufficiently small γ in $2^{n+o(n)}$ time. In other words, we must find an algorithm that outputs a list of γ-approximate closest vectors to the target $\mathbf{t}$, at least one of which is in the same coset mod $2\mathcal{L}$ of some exact closest vector $\mathbf{y}$. As we noted above, our discrete Gaussian sampler can be used to obtain approximate closest vectors with extremely good approximation factors. It therefore suffices to show that at least one of these will be in the same coset as an exact closest vector.

This is why the number of output samples that we computed in (2) is so remarkably convenient. If a coset's Gaussian mass is within a factor T of the mass of the maximum coset and we run our sampler, say, $T \cdot \mathrm{poly}(n)$ times, then with high probability one of our output vectors will land in this coset! In particular, if we can find a bound $T \leq 2^{o(n)}$ on the ratio between the maximal mass of a coset and a coset with a closest vector, then we can simply run our sampler $T \cdot \mathrm{poly}(n)$ times to obtain a $2^{n+o(n)}$-time solution to γ-nCVP, as needed. Intuitively, such a bound seems reasonable, as the closest vector itself has higher mass than any other point.

Unfortunately, we cannot have such a bound for arbitrary s. There exist "pathological" lattices $\mathcal{L}$ and targets $\mathbf{t}$ such that for some parameter s, the coset of the closest vector to $\mathbf{t}$ has relatively low mass, while some other coset contains many points whose combined mass is quite high, even though it does not

contain an exact closest vector. However, we can show that this cannot happen for "too many" different parameters s. Specifically, we show how to pick a list of parameters $s_1 \geq \cdots \geq s_\ell$ such that, for at least one of these parameters, the bound $T \leq 2^{o(n)}$ that we required above will hold. This suffices for our purposes. The proof of this statement is quite technical and relies heavily on the new inequality we prove in Section III. (See Corollary 33.)

C. Related work

Our exact CVP algorithm uses many ideas from many different types of lattice algorithms, including sieving, basis reduction, and discrete Gaussian sampling. Our algorithm combines these ideas in a way that (almost magically, and in ways that we do not fully understand) avoids the major pitfalls of each. We summarize the relationship of our algorithm to some prior work below.

First, our algorithm finds an approximate Hermite-Korkine-Zolatoreff (HKZ) basis and essentially "guesses" the last $n-k$ coefficients of a closest vector with respect to this basis. HKZ bases are extremely well-studied by the basis reduction community [5], [31], [45], [32], [33], and this idea is used in essentially all enumeration algorithms for CVP. However, there are examples where the standard basis enumeration techniques require $n^{\Omega(n)}$ time to solve CVP. (See, e.g., [46].) The main reason for this is that such techniques work recursively on *projections* of the base lattice, and the projected lattice often contains many points close to the projected target that do not "lift" to points close to the target in the full lattice. Using our techniques, we never need to project, and we are therefore able to ignore these useless points while still guaranteeing that we will find a point whose last $n-k$ coefficients with respect to the basis are equal to those of the closest vector.

Many other authors have noted that the approximate closest lattice vectors form clusters, mostly in the context of AKS-like sieving algorithms. For example, the $(1+\varepsilon)$-approximate closest vectors to $\mathbf{t}$ can be grouped into $2^{O(n)}(1+1/\varepsilon)^n$ clusters of diameter $\varepsilon \cdot \mathrm{dist}(\mathbf{t}, \mathcal{L})$ (see, e.g., [37], [47]). While the clustering bound that we obtain is both stronger and simpler to prove (using an elementary parity argument), we are unaware of prior work mentioning this particular bound. This is likely because sieving algorithms are typically concerned with constant-factor approximations, whereas our sampler allows us to work with "unconscionably" good approximation factors $\gamma = 1 + 2^{-o(n/\log n)}$. Our clustering bound seems to be both less natural and less useful for the constant-factor approximations achieved by $2^{O(n)}$-time sieving algorithms.

[48] improve on the MV algorithm by showing that, once the Voronoi cell of $\mathcal{L}$ has been computed, CVP on $\mathcal{L}$ can be solved in $\widetilde{O}(2^n)$ expected time. Indeed, before we found this algorithm, we hoped to solve CVP quickly by using the ADRS sampler to compute the Voronoi cell in $2^{n+o(n)}$ time. (This corresponds to computing the shortest vectors in every coset of $\mathcal{L}/(2\mathcal{L})$.) Even with our current techniques, we do not know how to achieve this, and we leave this as an open problem.

Finally, after this work was published, [49] showed a dimension-preserving reduction from DGS to CVP, answering a question posed in an earlier version of this paper. Together with our work, this reduction immediately implies a $2^{n+o(n)}$-time algorithm for DGS with *any parameter* s. (Our algorithm works for any parameter $s \geq \mathrm{dist}(\mathbf{t}, \mathcal{L}) \cdot 2^{o(n/\log n)}$, but not arbitrarily small s.) This also provides some (arguably weak) evidence that our technique of using DGS for solving CVP is "correct," in the sense that any faster algorithm for CVP necessarily yields a faster algorithm for DGS.

D. Open problems and directions for future work

Of course, the most natural and important open problem is whether a faster algorithm for CVP is possible. (Even an algorithm with the same running time as ours that is simpler or deterministic would be very interesting.) There seem to be fundamental barriers to significantly improving our method, as both our sampler and our reduction to exact CVP require enumeration over the 2^n cosets of $2\mathcal{L}$. And, Micciancio and Voulgaris note that their techniques also seem incapable of yielding an algorithm that runs in less than 2^n time (for similar reasons) [1]. Indeed, our techniques and those of MV seem to inherently solve the harder (though likely not very important) problem of finding *all* closest vectors simultaneously. Since there can be 2^n such vectors, this problem trivially cannot be solved in better than

2^n time in the worst case. So, if an algorithm with a better running time is to be found, it would likely require substantial new ideas.

Given these barriers, we also ask whether we can find a comparable lower bound. In particular, Micciancio and Voulgaris note that the standard NP-hardness proof for CVP actually shows that, assuming the Exponential Time Hypothesis, there is some constant $c > 0$ such that no 2^{cn}-time algorithm solves CVP [1]. Recent unpublished work by Samuel Yeom shows that we can take $c = 10^{-4}$ under plausible complexity assumptions [50]. Obviously, this gap is quite wide, and we ask whether we can make significant progress towards closing it.

In this work, we show how to use a technique that seems "inherently approximate" to solve *exact* CVP. I.e., our algorithm is randomized and, during any given recursive call, each γ-approximate closest vector has nearly the same likelihood of appearing as an exact closest vector for sufficiently small γ. Indeed, prior to this work, the only known algorithm that solved exact CVP in $2^{O(n)}$ time was the deterministic MV algorithm, while the "AKS-like" randomized sieving algorithms for CVP achieve only constant approximation factors. It would be very interesting to find exact variants of the sieving algorithms. The primary hurdle towards adapting our method to such algorithms seems to be the very good approximation factor that we require—our ideas seem to require an approximation factor of at most $\gamma = 1 + 1/\mathrm{poly}(n)$, while $2^{O(n)}$-time sieving algorithms only achieve constant approximation factors. But, it is plausible that our techniques could be adapted to work in this setting, potentially yielding an "AKS-like" algorithm for exact CVP. Even if such an algorithm were not provably faster than ours, it might be more efficient in practice, as sieving algorithms tend to outperform their provable running times (while our algorithm quite clearly runs in time at least 2^n).

A long-standing open problem is to find an algorithm that solves CVP in $2^{O(n)}$ time but *polynomial* space. Currently, the only known algorithms that run in polynomial space are the enumeration-based method of Kannan and its variants, which run in $n^{O(n)}$ time. Indeed, even for SVP, there is no known polynomial-space algorithm that runs in $2^{O(n)}$ time. This is part of the reason why $n^{O(n)}$-time enumeration-based methods are often used in practice to solve large instances of CVP and SVP, in spite of their much worse asymptotic running time.

The authors are particularly interested in finding a better explanation for why "everything seems to work out" so remarkably well in the analysis of our algorithm. It seems almost magical that we end up with exactly as many samples as we need for our CVP to DGS reduction to go through. We do not have a good intuitive understanding of why our sampler returns the number of samples that it does, but it seems largely unrelated to the reason that our CVP algorithm needs as many samples as it does. The fact that these two numbers are the same is remarkable, and we would love a clear explanation. A better understanding of this would be interesting in its own right, and it could lead to an improved algorithm.

II. Preliminaries

Let $\mathbb{N} = \{0, 1, 2, \ldots\}$. Except where we specify otherwise, we use C, C_1, and C_2 to denote universal positive constants, which might differ from one occurrence to the next (even in the same sequence of (in)equalities). We use bold letters $\mathbf{x}$ for vectors and denote a vector's coordinates with indices x_i. Throughout the paper, n will always be the dimension of the ambient space $\mathbb{R}^n$.

A. Lattices

A rank d lattice $\mathcal{L} \subset \mathbb{R}^n$ is the set of all integer linear combinations of d linearly independent vectors $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_d)$. $\mathbf{B}$ is called a basis of the lattice and is not unique. Formally, a lattice is represented by a basis $\mathbf{B}$ for computational purposes, though for simplicity we often do not make this explicit. If $n = d$, we say that the lattice has full rank. We often implicitly assume that the lattice is full rank, as otherwise we can simply work over the subspace spanned by the lattice.

Given a basis, $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$, we write $\mathcal{L}(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ to denote the lattice with basis $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$. The length of a shortest non-zero vector in the lattice is written $\lambda_1(\mathcal{L})$. For a vector $\mathbf{t} \in \mathbb{R}^n$, we write $\mathrm{dist}(\mathbf{t}, \mathcal{L})$ to denote the distance between $\mathbf{t}$ and the lattice, $\min_{\mathbf{y} \in \mathcal{L}}(\|\mathbf{y} - \mathbf{t}\|)$. We call any $\mathbf{y} \in \mathcal{L}$ minimizing $\|\mathbf{y} - \mathbf{t}\|$ a closest vector to $\mathbf{t}$. The covering radius is $\mu(\mathcal{L}) := \max_{\mathbf{t}} \mathrm{dist}(\mathbf{t}, \mathcal{L})$.

Definition 1. *For a lattice $\mathcal{L}$, the ith successive minimum of $\mathcal{L}$ is*

$$\lambda_i(\mathcal{L}) = \min\{r : \dim(\mathrm{span}(\mathcal{L} \cap B(\mathbf{0}, r))) \geq i\} .$$

Intuitively, the ith successive minimum of $\mathcal{L}$ is the smallest value r such that there are i linearly independent vectors in $\mathcal{L}$ of length at most r. We will need the following two facts.

Theorem 2 ([51, Theorem 2.1])**.** *For any lattice $\mathcal{L} \subset \mathbb{R}^n$ and $s > 0$,*

$$|\{\mathbf{y} \in \mathcal{L} : \|\mathbf{y}\| \leq s\lambda_1(\mathcal{L})\}| \leq 2\lceil 2s \rceil^n - 1.$$

Lemma 3. *For any lattice $\mathcal{L} \subset \mathbb{R}^n$ with basis $(\mathbf{b}_1, \ldots, \mathbf{b}_n)$,*

$$\lambda_n(\mathcal{L})^2 \leq \mu(\mathcal{L})^2 \leq \frac{1}{4} \cdot \sum_{i=1}^{n} \|\widetilde{\mathbf{b}}_i\|^2 .$$

B. The discrete Gaussian distribution

For any $s > 0$, we define the function $\rho_s : \mathbb{R}^n \to \mathbb{R}$ as $\rho_s(\mathbf{t}) := \exp(-\pi\|\mathbf{t}\|^2/s^2)$. When $s = 1$, we simply write $\rho(\mathbf{t})$. For a discrete set $A \subset \mathbb{R}^n$ we define $\rho_s(A) := \sum_{\mathbf{x} \in A} \rho_s(\mathbf{x})$.

Definition 4. *For a lattice $\mathcal{L} \subset \mathbb{R}^n$, a shift $\mathbf{t} \in \mathbb{R}^n$, and parameter $s > 0$, let $D_{\mathcal{L}-\mathbf{t},s}$ be the probability distribution over $\mathcal{L} - \mathbf{t}$ such that the probability of drawing $\mathbf{x} \in \mathcal{L} - \mathbf{t}$ is proportional to $\rho_s(\mathbf{x})$. We call this the discrete Gaussian distribution over $\mathcal{L} - \mathbf{t}$ with parameter s.*

We make frequent use of the discrete Gaussian over the cosets of a sublattice. If $\mathcal{L}' \subseteq \mathcal{L}$ is a sublattice of $\mathcal{L}$, then the set of cosets, $\mathcal{L}/\mathcal{L}'$ is the set of translations of $\mathcal{L}'$ by lattice vectors, $\mathbf{c} = \mathcal{L}' + \mathbf{y}$ for some $\mathbf{y} \in \mathcal{L}$. (Note that $\mathbf{c}$ is a *set*, not a vector.) Banaszczyk proved the following three bounds [52].

Lemma 5 ([52, Lemma 1.4])**.** *For any lattice $\mathcal{L} \subset \mathbb{R}^n$ and $s > 1$,*

$$\rho_s(\mathcal{L}) \leq s^n \rho(\mathcal{L}) .$$

Lemma 6. *For any lattice $\mathcal{L} \subset \mathbb{R}^n$, $s > 0$, $\mathbf{t} \in \mathbb{R}^n$*

$$\rho_s(\mathbf{t}) \leq \frac{\rho_s(\mathcal{L} - \mathbf{t})}{\rho_s(\mathcal{L})} \leq 1 .$$

Lemma 7 ([53, Lemma 2.13])**.** *For any lattice $\mathcal{L} \subset \mathbb{R}^n$, $s > 0$, $\mathbf{t} \in \mathbb{R}^n$, and $r \geq 1/\sqrt{2\pi}$,*

$$\Pr_{\mathbf{X} \sim D_{\mathcal{L}-\mathbf{t},s}}[\|\mathbf{X}\| \geq rs\sqrt{n}] < \frac{\rho_s(\mathcal{L})}{\rho_s(\mathcal{L} - \mathbf{t})} \left(\sqrt{2\pi e r^2} \exp(-\pi r^2)\right)^n .$$

From these, we derive the following corollary. (We include a proof in the full version.)

Corollary 8. *For any lattice $\mathcal{L} \subset \mathbb{R}^n$, $s > 0$, and $\mathbf{t} \in \mathbb{R}^n$, let $\alpha := \mathrm{dist}(\mathbf{t}, \mathcal{L})/(\sqrt{n}s)$. Then, for any $r \geq 1/\sqrt{2\pi}$,*

$$\Pr_{\mathbf{X} \sim D_{\mathcal{L}-\mathbf{t},s}}[\|\mathbf{X}\| \geq rs\sqrt{n}] < e^{\pi n \alpha^2} \left(\sqrt{2\pi e r^2} \exp(-\pi r^2)\right)^n . \tag{3}$$

Furthermore, if $\alpha \leq 2^n$, we have that

$$\Pr[\|\mathbf{X}\|^2 \geq \mathrm{dist}(\mathbf{t}, \mathcal{L})^2 + 2(sn)^2] \leq e^{-3n^2}.$$

C. The Gram-Schmidt orthogonalization and γ-HKZ bases

Given a basis, $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$, we define its Gram-Schmidt orthogonalization $(\widetilde{\mathbf{b}}_1, \ldots, \widetilde{\mathbf{b}}_n)$ by

$$\widetilde{\mathbf{b}}_i = \pi_{\{b_1,\ldots,b_{i-1}\}^\perp}(\mathbf{b}_i) \ ,$$

and the corresponding Gram-Schmidt coefficients $\mu_{i,j}$ by

$$\mu_{i,j} = \frac{\langle \mathbf{b}_i, \widetilde{\mathbf{b}}_j \rangle}{\|\widetilde{\mathbf{b}}_j\|^2} \ .$$

Here, π_A is the orthogonal projection on the subspace A and $\{\mathbf{b}_1, \ldots, \mathbf{b}_{i-1}\}^\perp$ denotes the subspace orthogonal to $\mathbf{b}_1, \ldots, \mathbf{b}_{i-1}$.

Definition 9. *A basis $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ of $\mathcal{L}$ is a γ-approximate Hermite-Korkin-Zolotarev (γ-HKZ) basis if*

1) $\|\mathbf{b}_1\| \leq \gamma \cdot \lambda_1(\mathcal{L})$;
2) *the Gram-Schmidt coefficients of* $\mathbf{B}$ *satisfy* $|\mu_{i,j}| \leq \frac{1}{2}$ *for all* $j < i$; *and*
3) $\pi_{\{\mathbf{b}_1\}^\perp}(\mathbf{b}_2), \ldots, \pi_{\{\mathbf{b}_1\}^\perp}(\mathbf{b}_n)$ *is a γ-HKZ basis of* $\pi_{\{\mathbf{b}_1\}^\perp}(\mathcal{L})$.

We use γ-HKZ bases in the sequel to find "sublattices that contain all short vectors." In particular, note that if $(\mathbf{b}_1, \ldots, \mathbf{b}_n)$ is a γ-HKZ basis for $\mathcal{L}$, then for any index k, $\mathcal{L}(\mathbf{b}_1, \ldots, \mathbf{b}_{k-1})$ contains all lattice vectors $\mathbf{y} \in \mathcal{L}$ with $\|\mathbf{y}\| < \|\widetilde{\mathbf{b}}_k\|/\gamma$. When $\gamma = 1$, we omit it.

D. Lattice problems

Definition 10. *For $\gamma = \gamma(n) \geq 1$ (the approximation factor), the search problem γ-CVP (Closest Vector Problem) is defined as follows: The input is a basis $\mathbf{B}$ for a lattice $\mathcal{L} \subset \mathbb{R}^n$ and a target vector $\mathbf{t} \in \mathbb{R}^n$. The goal is to output a vector $\mathbf{y} \in \mathcal{L}$ with $\|\mathbf{y} - \mathbf{t}\| \leq \gamma \cdot \mathrm{dist}(\mathbf{t}, \mathcal{L})$.*

When $\gamma = 1$, we omit it and call the problem *exact* CVP or simply CVP.

Definition 11. *For $\varepsilon \geq 0$ (the error), σ (the minimal parameter) a function that maps shifted lattices to non-negative real numbers, and m (the desired number of output vectors) a function that maps shifted lattices and positive real numbers to natural numbers, ε-DGS$^m_\sigma$ (the Discrete Gaussian Sampling problem) is defined as follows: The input is a basis $\mathbf{B}$ for a lattice $\mathcal{L} \subset \mathbb{R}^n$, a shift $\mathbf{t} \in \mathbb{R}^n$, and a parameter $s > \sigma(\mathcal{L} - \mathbf{t})$. The goal is to output a sequence of $\hat{m} \geq m(\mathcal{L} - \mathbf{t}, s)$ vectors whose joint distribution is ε-close to $D^{\hat{m}}_{\mathcal{L}-\mathbf{t},s}$.*

We stress that ε bounds the statistical distance between the *joint* distribution of the output vectors and $\hat{m}$ independent samples from $D_{\mathcal{L}-\mathbf{t},s}$.

E. Some known algorithms

The following theorem was proven by Ajtai, Kumar, and Sivakumar [34], building on work of Schnorr [54].

Theorem 12. *There is an algorithm that takes as input a lattice $\mathcal{L} \subset \mathbb{R}^n$, target $\mathbf{t} \in \mathbb{R}^n$, and parameter $u \geq 2$ and outputs a γ-HKZ basis of $\mathcal{L}$ and a γ'-approximate closest vector to $\mathbf{t}$ in time $2^{O(u)} \cdot \mathrm{poly}(n)$, where $\gamma := u^{n/u}$ and $\gamma' := \sqrt{n}u^{n/u}$.*

The next theorem was proven by [27].

Theorem 13. *For any $\gamma = \gamma(n) \geq 1$, there is an efficient dimension-preserving reduction from the problem of computing a γ-HKZ basis to γ-CVP.*

III. Some inequalities concerning Gaussians on shifted lattices

We first prove an inequality (Corollary 16) concerning the Gaussian measure over shifted lattices. We will use this inequality to show that our sampler outputs sufficiently many samples; and to show that our recursive CVP algorithm will "find a cluster with a closest point" with high probability. The inequality is similar in flavor to the main inequality in [44], and it (or the more general form given in Lemma 15) may have additional applications. The proof uses the following identity from [44].

Lemma 14 ([44, Eq. (3)]). *For any lattice $\mathcal{L} \subset \mathbb{R}^n$, any two vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$, and $s > 0$, we have*

$$\rho_s(\mathcal{L} - \mathbf{x})\rho_s(\mathcal{L} - \mathbf{y}) = \sum_{\mathbf{c} \in \mathcal{L}/(2\mathcal{L})} \rho_{\sqrt{2}s}(\mathbf{c} - \mathbf{x} - \mathbf{y})\rho_{\sqrt{2}s}(\mathbf{c} - \mathbf{x} + \mathbf{y}) \ .$$

Our inequality then follows easily.

Lemma 15. *For any lattice $\mathcal{L} \subset \mathbb{R}^n$, any two vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$, and $s > 0$, we have*

$$\rho_s(\mathcal{L} - \mathbf{x})\rho_s(\mathcal{L} - \mathbf{y}) \leq \max_{\mathbf{c} \in \mathcal{L}/(2\mathcal{L})} \rho_{\sqrt{2}s}(\mathbf{c} - \mathbf{x} - \mathbf{y}) \cdot \rho_{\sqrt{2}s}(\mathcal{L} - \mathbf{x} + \mathbf{y}) \ .$$

Proof: Using Lemma 14, we get the following.

$$\begin{aligned}
\rho_s(\mathcal{L} - \mathbf{x})\rho_s(\mathcal{L} - \mathbf{y}) &= \sum_{\mathbf{c} \in \mathcal{L}/(2\mathcal{L})} \rho_{\sqrt{2}s}(\mathbf{c} - \mathbf{x} - \mathbf{y})\rho_{\sqrt{2}s}(\mathbf{c} - \mathbf{x} + \mathbf{y}) \\
&\leq \max_{\mathbf{c} \in \mathcal{L}/(2\mathcal{L})} \rho_{\sqrt{2}s}(\mathbf{c} - \mathbf{x} - \mathbf{y}) \cdot \sum_{\mathbf{d} \in \mathcal{L}/(2\mathcal{L})} \rho_{\sqrt{2}s}(\mathbf{d} - \mathbf{x} + \mathbf{y}) \\
&= \max_{\mathbf{c} \in \mathcal{L}/(2\mathcal{L})} \rho_{\sqrt{2}s}(\mathbf{c} - \mathbf{x} - \mathbf{y}) \cdot \rho_{\sqrt{2}s}(\mathcal{L} - \mathbf{x} + \mathbf{y}) \ .
\end{aligned}$$

■

Setting $\mathbf{x} = \mathbf{y} = \mathbf{w} + \mathbf{t}$ for any $\mathbf{w} \in \mathcal{L}$ and switching $2\mathcal{L}$ with $\mathcal{L}$ gives the following inequality.

Corollary 16. *For any lattice $\mathcal{L} \subset \mathbb{R}^n$, $\mathbf{t} \in \mathbb{R}^n$, and $s > 0$, we have*

$$\max_{\mathbf{c} \in \mathcal{L}/(2\mathcal{L})} \rho_s(\mathbf{c} - \mathbf{t})^2 \leq \max_{\mathbf{c} \in \mathcal{L}/(2\mathcal{L})} \rho_{s/\sqrt{2}}(\mathbf{c} - \mathbf{t}) \cdot \rho_{s/\sqrt{2}}(\mathcal{L}) \ .$$

IV. Sampling from the discrete Gaussian

A. Combining discrete Gaussian samples

The following lemma and proposition are the shifted analogues of [2, Lemma 3.4] and [2, Proposition 3.5] respectively. Their proofs are included in the full version for completeness.

Lemma 17. *Let $\mathcal{L} \subset \mathbb{R}^n$, $s > 0$ and $\mathbf{t} \in \mathbb{R}^n$. Then for all $\mathbf{y} \in \mathcal{L} - \mathbf{t}$,*

$$\Pr_{(\mathbf{X}_1, \mathbf{X}_2) \sim D^2_{\mathcal{L}-\mathbf{t}, s}} [(\mathbf{X}_1 + \mathbf{X}_2)/2 = \mathbf{y} \mid \mathbf{X}_1 + \mathbf{X}_2 \in 2\mathcal{L} - 2\mathbf{t}] = \Pr_{\mathbf{X} \sim D_{\mathcal{L}-\mathbf{t}, s/\sqrt{2}}} [\mathbf{X} = \mathbf{y}] \ . \tag{4}$$

Proposition 18. *There is an algorithm that takes as input a lattice $\mathcal{L} \subset \mathbb{R}^n$, $\mathbf{t} \in \mathbb{R}^n$, $\kappa \geq 2$ (the confidence parameter), and a sequence of vectors from $\mathcal{L} - \mathbf{t}$, and outputs a sequence of vectors from $\mathcal{L} - \mathbf{t}$ such that, if the input consists of*

$$M \geq 10\kappa^2 \cdot \frac{\rho_s(\mathcal{L} - \mathbf{t})}{\max_{\mathbf{c} \in \mathcal{L}/(2\mathcal{L})} \rho_s(\mathbf{c} - \mathbf{t})}$$

independent samples from $D_{\mathcal{L}-\mathbf{t}, s}$ for some $s > 0$, then the output is within statistical distance $M \exp(C_1 n - C_2 \kappa)$ of m independent samples from $D_{\mathcal{L}-\mathbf{t}, s/\sqrt{2}}$ where m is a random variable with

$$m \geq M \cdot \frac{1}{32\kappa} \cdot \frac{\rho_{s/\sqrt{2}}(\mathcal{L}) \cdot \rho_{s/\sqrt{2}}(\mathcal{L} - \mathbf{t})}{\rho_s(\mathcal{L} - \mathbf{t}) \max_{\mathbf{c} \in \mathcal{L}/(2\mathcal{L})} \rho_s(\mathbf{c} - \mathbf{t})} \ .$$

The running time of the algorithm is at most $M \cdot \mathrm{poly}(n, \log \kappa)$.

We will show in Theorem 19 that by calling the algorithm from Proposition 18 repeatedly, we obtain a general discrete Gaussian combiner.

Theorem 19. *There is an algorithm that takes as input a lattice $\mathcal{L} \subset \mathbb{R}^n$, $\ell \in \mathbb{N}$ (the step parameter), $\kappa \geq 2$ (the confidence parameter), $\mathbf{t} \in \mathbb{R}^n$, and $M = (32\kappa)^{\ell+1} \cdot 2^n$ vectors in $\mathcal{L}$ such that, if the input vectors are distributed*

as $D_{\mathcal{L}-\mathbf{t},s}$ for some $s > 0$, then the output is a list of vectors whose distribution is within statistical distance $\ell M \exp(C_1 n - C_2 \kappa)$ of at least

$$m = \frac{\rho_{2^{-\ell/2}s}(\mathcal{L} - \mathbf{t})}{\max_{\mathbf{c} \in \mathcal{L}/(2\mathcal{L})} \rho_{2^{-\ell/2}s}(\mathbf{c} - \mathbf{t})}$$

independent samples from $D_{\mathcal{L}-\mathbf{t},2^{-\ell/2}s}$. The algorithm runs in time $\ell M \cdot \mathrm{poly}(n, \log \kappa)$.

Proof: Let $\mathcal{X}_0 = (\mathbf{X}_1, \ldots, \mathbf{X}_M)$ be the sequence of input vectors. For $i = 0, \ldots, \ell - 1$, the algorithm calls the procedure from Proposition 18 with input $\mathcal{L}$, κ, and $\mathcal{X}_i$, receiving an output sequence $\mathcal{X}_{i+1}$ of length M_{i+1}. Finally, the algorithm outputs the sequence $\mathcal{X}_\ell$.

The running time is clear. Fix $\mathcal{L}$, s, $\mathbf{t}$ and ℓ. Define $\theta(i) := \rho_{2^{-i/2}s}(\mathcal{L})$, $\phi(i) := \max_{\mathbf{c} \in \mathcal{L}/(2\mathcal{L})} \rho_{2^{-i/2}s}(\mathbf{c} - \mathbf{t})$, and $\psi(i) := \rho_{2^{-i/2}s}(\mathcal{L} - \mathbf{t})$.

We wish to prove by induction that $\mathcal{X}_i$ is within statistical distance $iM \exp(C_1 n - C_2 \kappa)$ of $D^{M_i}_{\mathcal{L}-\mathbf{t},2^{-i/2}s}$ with

$$M_i \geq (32\kappa)^{\ell - i + 1} \cdot \frac{\psi(i)}{\phi(i)} \,, \tag{5}$$

for all $i \geq 1$. This implies that $M_\ell \geq m$ as needed.

Let

$$L(i) := \frac{\theta(i+1)\psi(i+1)}{\psi(i)\phi(i)} \,,$$

be the "loss factor" resulting from the $(i+1)$st run of the combiner, ignoring the factor of 32κ. By Corollary 16, we have

$$L(i) \geq \frac{\psi(i+1)}{\phi(i+1)} \cdot \frac{\phi(i)}{\psi(i)} \,. \tag{6}$$

By Proposition 18, up to statistical distance $M \exp(C_1 n - C_2 \kappa)$, we have that $\mathcal{X}_1$ has the right distribution with

$$\begin{aligned} M_1 &\geq \frac{1}{32\kappa} \cdot M_0 \cdot L(0) \\ &\geq (32\kappa)^\ell \cdot 2^n \cdot \frac{\psi(1)}{\phi(1)} \cdot \frac{\phi(0)}{\psi(0)} \,, \end{aligned}$$

where we used Eq. (6) with $i = 0$. By noting that $\psi(0) \leq 2^n \phi(0)$, we see that (5) holds when $i = 1$.

Suppose that $\mathcal{X}_i$ has the correct distribution and (5) holds for some i with $0 \leq i < \ell$. In particular, we have that M_i is at least $10\kappa^2 \psi(i)/\phi(i)$. This is precisely the condition necessary to apply Proposition 18. So, we can apply the proposition and the induction hypothesis and obtain that (up to statistical distance at most $(i+1)M \exp(C_1 n - C_2 \kappa)$), $\mathcal{X}_{i+1}$ has the correct distribution with

$$M_{i+1} \geq \frac{1}{32\kappa} \cdot M_i \cdot L(i) \geq (32\kappa)^{\ell - i} \cdot \frac{\psi(i)}{\phi(i)} \cdot \frac{\phi(i)}{\psi(i)} \cdot \frac{\psi(i+1)}{\phi(i+1)} = (32\kappa)^{\ell - i} \cdot \frac{\psi(i+1)}{\phi(i+1)} \,,$$

where in the second inequality we used the induction hypothesis and Eq. (6). ∎

B. Initializing the sampler

The following two standard results are proven in the full version.

Proposition 20. *There is an algorithm that takes as input a lattice $\mathcal{L} \subset \mathbb{R}^n$, shift $\mathbf{t} \in \mathbb{R}^n$, $r > 0$, and parameter $u \geq 2$, such that if*

$$r \geq u^{n/u}(1 + \sqrt{n} u^{n/u}) \cdot \mathrm{dist}(\mathbf{t}, \mathcal{L}) \,,$$

then the output of the algorithm is $\mathbf{y} \in \mathcal{L}$ and a basis $\mathbf{B}'$ of a (possibly trivial) sublattice $\mathcal{L}' \subseteq \mathcal{L}$ such that all vectors from $\mathcal{L} - \mathbf{t}$ of length at most $r/u^{n/u} - \mathrm{dist}(\mathbf{t}, \mathcal{L})$ are also contained in $\mathcal{L}' - \mathbf{y} - \mathbf{t}$, and $\|\widetilde{\mathbf{B}'}\| \leq r$. The algorithm runs in time $\mathrm{poly}(n) \cdot 2^{O(u)}$.

Corollary 21. *There is an algorithm that takes as input a lattice $\mathcal{L} \subset \mathbb{R}^n$ with $n \geq 2$, shift $\mathbf{t} \in \mathbb{R}^n$, $M \in \mathbb{N}$ (the desired number of output vectors), and parameters $u \geq 2$ and $\hat{s} > 0$ and outputs $\mathbf{y} \in \mathcal{L}$, a (possibly trivial) sublattice $\mathcal{L}' \subseteq \mathcal{L}$, and M vectors from $\mathcal{L}' - \mathbf{y} - \mathbf{t}$ such that if*

$$\hat{s} \geq C\sqrt{n \log n} \cdot u^{2n/u} \cdot \operatorname{dist}(\mathbf{t}, \mathcal{L}) ,$$

then the output vectors are distributed as M independent samples from $D_{\mathcal{L}'-\mathbf{y}-\mathbf{t},\hat{s}}$, and $\mathcal{L}' - \mathbf{y} - \mathbf{t}$ contains all vectors in $\mathcal{L} - \mathbf{t}$ of length at most $C\hat{s}/(u^{n/u}\sqrt{\log n})$. The algorithm runs in time $\operatorname{poly}(n) \cdot 2^{O(u)} + \operatorname{poly}(n) \cdot M$.

C. The sampler

We are now ready to present our discrete Gaussian sampler.

Theorem 22. *For any efficiently computable function $f(n) \geq n^{\omega(1)}$, let σ be the function defined by $\sigma(\mathcal{L} - \mathbf{t}) := \operatorname{dist}(\mathbf{t}, \mathcal{L})/f(n)$ for any lattice $\mathcal{L} \subset \mathbb{R}^n$ and $\mathbf{t} \in \mathbb{R}^n$. Let*

$$m(\mathcal{L} - \mathbf{t}, s) := \frac{\rho_s(\mathcal{L} - \mathbf{t})}{\max_{\mathbf{c} \in \mathcal{L}/(2\mathcal{L})} \rho_s(\mathbf{c} - \mathbf{t})} .$$

Then, there is an algorithm that solves ε-DGS$_\sigma^m$ with $\varepsilon(n) := 2^{-Cn^2}$ in time $2^{n+O(\log n \log f(n))}$.

Proof: We assume without loss of generality that $f(n) \geq 2n > 10$. The algorithm behaves as follows on input a lattice $\mathcal{L} \subset \mathbb{R}^n$, a shift $\mathbf{t}$, and a parameter $s > \sigma(\mathcal{L} - \mathbf{t})$. First, it runs the procedure from Corollary 21 with input $\mathcal{L}$, $\mathbf{t}$, $M := (Cn^2)^{\ell+2} \cdot 2^n$ with $\ell := C\lceil \log f(n) \rceil$, $u := Cn \log n / \log f(n) + 2$, and

$$\hat{s} := 2^\ell s > C\sqrt{n \log n} \cdot u^{2n/u} \cdot \operatorname{dist}(\mathbf{t}, \mathcal{L}) .$$

(Note that $u^{n/u} \leq f(n)^C$.) It receives as output $\mathcal{L}' \subset \mathbb{R}^n$, $\mathbf{y} \in \mathcal{L}$, and $(\mathbf{X}_1, \ldots, \mathbf{X}_M) \in \mathcal{L}' - \mathbf{y} - \mathbf{t}$. It then runs the procedure from Theorem 19 *twice*, first with input $\mathcal{L}'$, ℓ, $\kappa := Cn^2$, $\mathbf{t}$, and the first half of the vectors, $(\mathbf{X}_1, \ldots, \mathbf{X}_{M/2})$; and next with input $\mathcal{L}'$, ℓ, κ, $\mathbf{t}$, and the second half of the vectors, $(\mathbf{X}_{M/2+1}, \ldots, \mathbf{X}_M)$. Finally, it outputs the resulting vectors.

The running time follows from the respective running times of the two subprocedures. In particular, the procedure from Corollary 21 runs in time $\operatorname{poly}(n) \cdot (2^{O(u)} + M) = n^{O(n/\log f(n))} + 2^{n+O(\log n \log f(n))} = 2^{n+O(\log n \log f(n))}$, and the procedure from Theorem 19 runs in time $\ell M \cdot \operatorname{poly}(n, \log \kappa) = 2^{n+O(\log n \log f(n))}$.

By Corollary 21, the $\mathbf{X}_i$ are M independent samples from $D_{\mathcal{L}'-\mathbf{y}-\mathbf{t},\hat{s}}$ and $\mathcal{L}' - \mathbf{y} - \mathbf{t}$ contains all vectors in $\mathcal{L} - \mathbf{t}$ of length at most $C\hat{s}/(u^{n/u}\sqrt{\log n})$. By Theorem 19, the output contains at least $2m(\mathcal{L}' - \mathbf{t}, s)$ vectors whose distribution is within statistical distance 2^{-Cn^2} of independent samples from $D_{\mathcal{L}'-\mathbf{y}-\mathbf{t},s}$.

We now show that $D_{\mathcal{L}'-\mathbf{y}-\mathbf{t},s}$ is statistically close to $D_{\mathcal{L}-\mathbf{t},s}$. Let $d := \operatorname{dist}(\mathbf{t}, \mathcal{L})$ and

$$r := \frac{C2^\ell}{u^{n/u}\sqrt{n \log n}} \geq f(n)^C \geq \frac{1}{\sqrt{2\pi}} .$$

The statistical distance is exactly

$$\begin{aligned}
\Pr_{\mathbf{w} \sim D_{\mathcal{L}-\mathbf{t},s}}\left[\mathbf{w} \notin \mathcal{L}' - \mathbf{y} - \mathbf{t}\right] &< \Pr_{\mathbf{w} \sim D_{\mathcal{L}-\mathbf{t},s}}\left[\|\mathbf{w}\| > C\hat{s}/(u^{n/u}\sqrt{\log n})\right] \\
&= \Pr_{\mathbf{w} \sim D_{\mathcal{L}-\mathbf{t},s}}\left[\|\mathbf{w}\| > rs\sqrt{n}\right] \\
&< e^{\pi d^2/s^2} e^{-f(n)^C} \\
&< 2^{-Cn^2} ,
\end{aligned}$$

where we have used Corollary 8. It follows that the output has the correct size and distribution. In particular, it follows from applying union bound over the output samples that the distribution of the output is within statistical distance ε of independent samples from $D_{\mathcal{L}-\mathbf{t},s}$, and an easy calculation shows that $2m(\mathcal{L}' - \mathbf{t}, s) > m(\mathcal{L} - \mathbf{t}, s)$. ∎

From Theorem 22 and Corollary 8, we immediately get a weaker version of our main result, a $2^{n+o(n)}$-time algorithm for γ-CVP for any $\gamma = 1 + 2^{-o(n/\log n)}$.

Corollary 23. *For any efficiently computable function $f(n) \geq n^{\omega(1)}$, there is an algorithm solving $(1+1/f(n))$-CVP (with high probability) in time $2^{n+O(\log n \log f(n))}$. In particular, if $f(n) = 2^{o(n/\log n)}$, the algorithm runs in time $2^{n+o(n)}$.*

V. Reduction from exact CVP to a variant of approximate CVP

A. Clusters of approximate closest lattice vectors

We now show that the approximate closest lattice vectors to $\mathbf{t}$ form "clusters." This will motivate an algorithm that solves *exact* CVP by working recursively "inside the clusters."

Lemma 24. *For any lattice $\mathcal{L} \subset \mathbb{R}^n$, $\mathbf{t} \in \mathbb{R}^n$, $r_1, r_2 > 0$, and $\mathbf{w}_1, \mathbf{w}_2 \in \mathcal{L} - \mathbf{t}$ with $\mathbf{w}_1 \equiv \mathbf{w}_2 \pmod{2\mathcal{L}}$, if the $\mathbf{w}_i$ satisfy $\|\mathbf{w}_i\|^2 < \operatorname{dist}(\mathbf{t}, \mathcal{L})^2 + r_i^2$, then $\|\mathbf{w}_1 - \mathbf{w}_2\|^2 < 2(r_1^2 + r_2^2)$.*

Proof: Since $\mathbf{w}_1 \equiv \mathbf{w}_2 \pmod{2\mathcal{L}}$, we have that $(\mathbf{w}_1 + \mathbf{w}_2)/2 \in \mathcal{L} - \mathbf{t}$. Therefore, we have that

$$\begin{aligned}\|\mathbf{w}_1 - \mathbf{w}_2\|^2 &= 2\|\mathbf{w}_1\|^2 + 2\|\mathbf{w}_2\|^2 - 4\|(\mathbf{w}_1 + \mathbf{w}_2)/2\|^2 \\ &< 2(\operatorname{dist}(\mathbf{t}, \mathcal{L})^2 + r_1^2) + 2(\operatorname{dist}(\mathbf{t}, \mathcal{L})^2 + r_2^2) - 4\operatorname{dist}(\mathbf{t}, \mathcal{L})^2 \\ &= 2(r_1^2 + r_2^2)\ .\end{aligned}$$

■

In particular, Lemma 24 shows that there are at most 2^n clusters of approximate closest points. We now derive an immediate corollary, which shows that, if the points are very close to $\mathbf{t}$, then each cluster lies in a shift of a lower-dimensional lattice $\mathcal{L}'$, defined in terms of a γ-HKZ basis.

Corollary 25. *For any $\mathcal{L} \subset \mathbb{R}^n$ with γ-HKZ basis $(\mathbf{b}_1, \ldots, \mathbf{b}_n)$ for some $\gamma \geq 1$, $\mathbf{t} \in \mathbb{R}^n$, and $k \in [n]$, let $\mathcal{L}' := \mathcal{L}(\mathbf{b}_1, \ldots, \mathbf{b}_{k-1})$. If $\mathbf{w}_1, \mathbf{w}_2 \in \mathcal{L} - \mathbf{t}$ with $\mathbf{w}_1 \equiv \mathbf{w}_2 \pmod{2\mathcal{L}}$ satisfy $\|\mathbf{w}_i\|^2 < \operatorname{dist}(\mathbf{t}, \mathcal{L})^2 + \|\widetilde{\mathbf{b}}_k\|^2/\gamma^2$, then $\mathbf{w}_1 \in \mathcal{L}' + \mathbf{w}_2$.*

Proof: Let $2\mathbf{v} = \mathbf{w}_1 - \mathbf{w}_2 \neq \mathbf{0}$. Note that $\mathbf{v} \in \mathcal{L}$ by hypothesis, and by Lemma 24, we have that $\|\mathbf{v}\| < \|\widetilde{\mathbf{b}}_k\|/\gamma$. Since $\lambda_1(\pi_{\mathcal{L}'^{\perp}}(\mathcal{L})) \geq \|\widetilde{\mathbf{b}}_k\|/\gamma$, it follows that $\mathbf{v} \in \mathcal{L}'$, as needed. ■

B. Recursion inside the clusters

Corollary 25 suggests a strategy for solving *exact* CVP recursively. The idea is to find a γ-HKZ basis and a list of approximate closest vectors $\mathbf{y}_1, \ldots, \mathbf{y}_{\hat{p}}$ such that at least one of them is in the same coset mod $2\mathcal{L}$ of an exact closest vector. By calling the algorithm recursively "inside the clusters" defined by the $\mathbf{y}_i$ and the basis, we will eventually find an exact closest vector. This motivates the following definition.

Definition 26. *For $\gamma = \gamma(n) \geq 1$ (the approximation factor) and $p = p(n) \geq 1$ (a bound on the output size), the search problem γ-nCVP^p (neighbor Closest Vector Problem) is defined as follows: The input is a basis $\mathbf{B}$ for a lattice $\mathcal{L} \subset \mathbb{R}^n$ and a target vector $\mathbf{t} \in \mathbb{R}^n$. The goal is to output lattice vectors $\mathbf{y}_1, \ldots, \mathbf{y}_{\hat{p}} \in \mathcal{L}$ with $\hat{p} \leq p(n)$ with $\|\mathbf{y}_i - \mathbf{t}\| \leq \gamma \cdot \operatorname{dist}(\mathbf{t}, \mathcal{L})$ such that there exists an index j and $\mathbf{y}' \in \mathcal{L}$ with $\|\mathbf{y}' - \mathbf{t}\| = \operatorname{dist}(\mathbf{t}, \mathcal{L})$ and $\mathbf{y}_j \equiv \mathbf{y}' \pmod{2\mathcal{L}}$.*

In other words, all of the $\mathbf{y}_i$ should be γ-approximate closest vectors, and at least one of them must be in the same coset as an exact closest vector. We think of this point as a "neighbor" of the closest point, since Lemma 24 tells us that the two points must be close to each other. Indeed, Theorem 22 shows that our DGS algorithm outputs a number of vectors that is roughly proportional to "the number of cosets with relatively high Gaussian mass." If we assume that a coset with an exact closest vector has relatively high Gaussian mass, then our DGS algorithm should yield a solution to nCVP. (In Section VI, we make this precise.)

We now analyze the reduction described above in the special case when $p = 1$. This will imply correctness in the general case, when p is arbitrarily large. But, we will need to do some more work in order to bound the running time in the general case.

Lemma 27. *There is an efficient reduction from exact CVP to γ-nCVP^1 for any efficiently computable $1 \leq \gamma(n) < 1 + 1/n$.*

Proof: On input $\mathcal{L} \subset \mathbb{R}^n$ and $\mathbf{t} \in \mathbb{R}^n$, the reduction behaves as follows. First, if $n = 1$, it solves the one-dimensional CVP instance in the straightforward way. Otherwise, it uses Theorem 13 and its nCVP oracle to compute a γ-HKZ basis $(\mathbf{b}_1, \ldots, \mathbf{b}_n)$ for $\mathcal{L}$. It then calls its nCVP oracle on input $\mathcal{L}$ and $\mathbf{t}$ and receives as output $\mathbf{y} \in \mathcal{L}$. Let $(\widetilde{\mathbf{b}}_1, \ldots, \widetilde{\mathbf{b}}_n)$ be the Gram-Schmidt orthogonalization of the $\mathbf{b}_i$, and choose any index k such that $\|\widetilde{\mathbf{b}}_k\|^2 > (\gamma(n)^2 - 1) \cdot \|\mathbf{y} - \mathbf{t}\|^2$. Let $\mathcal{L}' := \mathcal{L}(\mathbf{b}_1, \ldots, \mathbf{b}_{k-1})$. The reduction then calls itself recursively on input $\mathcal{L}'$ and $\mathbf{t} - \mathbf{y}$, receiving as output $\mathbf{x} \in \mathcal{L}'$. Finally, it returns $\mathbf{y} + \mathbf{x}$.

We first argue that the index k always exists. In particular, there must be some index j such that

$$\begin{aligned}
\|\widetilde{\mathbf{b}}_j\|^2 &\geq \frac{1}{n} \cdot \sum_i \|\widetilde{\mathbf{b}}_i\|^2 \\
&\geq \frac{4}{n} \cdot \operatorname{dist}(\mathbf{t}, \mathcal{L})^2 && \text{(Lemma 3)} \\
&\geq \frac{4}{\gamma(n)^2 n} \cdot \|\mathbf{y} - \mathbf{t}\|^2 \\
&> (\gamma(n)^2 - 1) \cdot \|\mathbf{y} - \mathbf{t}\|^2 .
\end{aligned}$$

Therefore, k exists, and since $\dim(\mathcal{L}') < \dim(\mathcal{L})$, the reduction terminates in at most n recursive calls. It follows that it runs in polynomial time.

If $n = 1$, then correctness is clear. We assume for induction that the reduction is correct when the dimension of the lattice is less than n. By the induction hypothesis, $\mathbf{x}$ is a closest vector to $\mathbf{t} - \mathbf{y}$ in $\mathcal{L}'$, and it follows that $\mathbf{y} + \mathbf{x}$ is a closest vector to $\mathbf{t}$ in $\mathcal{L}' + \mathbf{y}$. By the definition of nCVP, there is a vector $\mathbf{y}' \in 2\mathcal{L} + \mathbf{y}$ that is a closest vector to $\mathbf{t}$ in $\mathcal{L}$. Since

$$\|\mathbf{y} - \mathbf{t}\|^2 < \|\mathbf{y} - \mathbf{t}\|^2/\gamma(n)^2 + \|\widetilde{\mathbf{b}}_k\|^2/\gamma(n)^2 \leq \operatorname{dist}(\mathbf{t}, \mathcal{L})^2 + \|\widetilde{\mathbf{b}}_k\|^2/\gamma(n)^2 ,$$

we may apply Corollary 25 to see that $\mathbf{y}' \in \mathcal{L}' + \mathbf{y}$. In particular, a closest vector to $\mathbf{t}$ in $\mathcal{L}' + \mathbf{y}$ must also be a closest vector to $\mathbf{t}$ in the full lattice $\mathcal{L}$, and the result follows. ∎

C. Bounding the number of recursive calls

The next two technical lemmas will allow us to bound the number of recursive calls made by the reduction described in Section V-B by bounding the number of shifts of $\mathcal{L}'$ that contain approximate closest vectors.

Lemma 28. *For any $\mathcal{L} \subset \mathbb{R}^n$ with γ-HKZ basis $(\mathbf{b}_1, \ldots, \mathbf{b}_n)$ for some $\gamma \geq 1$, $\mathbf{t} \in \mathbb{R}^n$, and $k \in [n]$, let $\mathcal{L}' := \mathcal{L}(\mathbf{b}_1, \ldots, \mathbf{b}_{k-1})$. If $r > 0$, $s > 0$, and $k \leq \ell \leq n + 1$ satisfy*

$$r^2 + \frac{(k-1)}{2} \cdot \sum_{i=1}^{k-1} \|\widetilde{\mathbf{b}}_i\|^2 \leq \frac{1}{\gamma} \cdot \begin{cases} s^2 \|\widetilde{\mathbf{b}}_k\|^2 & : \ell = n+1 \\ \min\{s^2 \|\widetilde{\mathbf{b}}_k\|^2, \|\widetilde{\mathbf{b}}_\ell\|^2\} & : \text{otherwise} \end{cases} , \tag{7}$$

then we have that

$$\left|\{\mathbf{c} \in \mathcal{L}/\mathcal{L}' \ : \ \operatorname{dist}(\mathbf{t}, \mathbf{c})^2 < \operatorname{dist}(\mathbf{t}, \mathcal{L})^2 + r^2\}\right| \leq 2^{n-k+1}(2\lceil 2s \rceil^{\ell-k} - 1) .$$

Proof: For each $\mathbf{d} \in \mathcal{L}/(2\mathcal{L} + \mathcal{L}')$, let

$$S_{\mathbf{d}} := \{\mathbf{c} \in \mathcal{L}/\mathcal{L}' \ : \ \mathbf{c} \subset \mathbf{d} \text{ and } \operatorname{dist}(\mathbf{t}, \mathbf{c})^2 < \operatorname{dist}(\mathbf{t}, \mathcal{L})^2 + r^2\}$$

be the set of shifts of $\mathcal{L}'$ that are subsets of $\mathbf{d}$ and contain an approximate closest vector. Since $\mathcal{L}/\mathcal{L}'$ is a refinement of $\mathcal{L}/(2\mathcal{L} + \mathcal{L}')$ and $|\mathcal{L}/(2\mathcal{L} + \mathcal{L}')| = 2^{n-k+1}$, it suffices to show that $|S_{\mathbf{d}}| \leq (2\lceil 2s \rceil^{l-k} - 1)$ for all $\mathbf{d} \in \mathcal{L}/(2\mathcal{L} + \mathcal{L}')$.

Fix $\mathbf{d}$. Let $\mathbf{w}_1, \mathbf{w}_2 \in \mathbf{d} - \mathbf{t}$. Suppose $\|\mathbf{w}_i\|^2 < \operatorname{dist}(\mathbf{t}, \mathcal{L})^2 + r^2$. A simple computation shows that there exist $a_1, \ldots, a_{k-1} \in \{-1, 0, 1\}$ such that $\mathbf{w}_1 - \sum_{i=1}^{k-1} a_i \mathbf{b}_i \equiv \mathbf{w}_2 \pmod{2\mathcal{L}}$ and

$$\left\| \mathbf{w}_1 - \sum_{i=1}^{k-1} a_i \mathbf{b}_i \right\|^2 \leq \|\mathbf{w}_1\|^2 + \sum_{i=1}^{k-1} \|\mathbf{b}_i\|^2 < \operatorname{dist}(\mathbf{t}, \mathcal{L})^2 + r^2 + \sum_{i=1}^{k-1} \|\mathbf{b}_i\|^2 .$$

Since the $\mathbf{b}_i$ are γ-HKZ, we have that $\|\mathbf{b}_i\|^2 \le \|\widetilde{\mathbf{b}}_i\|^2 + \frac{1}{4}\sum_{i=j}^{i-1}\|\widetilde{\mathbf{b}}_j\|^2$. Therefore,

$$\left\|\mathbf{w}_1 - \sum_{i=1}^{k-1} a_i\mathbf{b}_i\right\|^2 < \operatorname{dist}(\mathbf{t},\mathcal{L})^2 + r^2 + (k-1)\sum_{i=1}^{k-1}\|\widetilde{\mathbf{b}}_i\|^2 .$$

Let $2\mathbf{v} := \mathbf{w}_1 - \sum_{i=1}^{k-1} a_i\mathbf{b}_i - \mathbf{w}_2 \in 2\mathcal{L}$. Since $\mathbf{w}_1 - \sum_{i=1}^{k-1} a_i\mathbf{b}_i \equiv \mathbf{w}_2 \pmod{2\mathcal{L}}$, we may apply Lemma 24 to obtain

$$\|\mathbf{v}\|^2 < r^2 + \frac{k-1}{2}\cdot\sum_{i=1}^{k-1}\|\widetilde{\mathbf{b}}_i\|^2 .$$

Let $\pi_k := \pi_{\{\mathbf{b}_1,\ldots,\mathbf{b}_{k-1}\}^\perp}$ and $\mathcal{M} := \pi_k(\mathcal{L}(\mathbf{b}_k,\ldots,\mathbf{b}_{\ell-1}))$. From the above, we have

$$\|\pi_k(\mathbf{v})\|^2 \le \|\mathbf{v}\|^2 < r^2 + \frac{k-1}{2}\cdot\sum_{i=1}^{k-1}\|\widetilde{\mathbf{b}}_i\|^2 .$$

Recalling the constraint on ℓ imposed by Eq. (7), this implies that $\pi_k(\mathbf{v}) \in \mathcal{M}$. Furthermore, note that $\mathbf{w}_1 \in \mathcal{L}' + \mathbf{w}_2$ if and only if $\pi_k(\mathbf{w}_1 - \mathbf{w}_2) = \pi_k(\mathbf{v}) = \mathbf{0}$. Therefore,

$$|S_{\mathbf{d}}| \le \left|\left\{\mathbf{y}\in\mathcal{M} \;:\; \|\mathbf{y}\| < r^2 + \frac{k-1}{2}\cdot\sum_{i=1}^{k-1}\|\widetilde{\mathbf{b}}_i\|^2\right\}\right| .$$

Finally, note that $\lambda_1(\mathcal{M}) \ge \|\widetilde{\mathbf{b}}_k\|/\gamma$. By Eq. (7) the length bound in the above equation is at most $s\lambda_1(\mathcal{M})$. The result then follows from applying Theorem 2 and noting that $\dim\mathcal{M} = \ell - k$. ∎

This next lemma shows that we can choose an index k such that either $\dim\mathcal{L}'$ is fairly small or relatively few shifts of $\mathcal{L}'$ contain approximate closest vectors.

Lemma 29. *For any lattice $\mathcal{L}\subset\mathbb{R}^n$ with γ-HKZ basis $(\mathbf{b}_1,\ldots,\mathbf{b}_n)$ for some $n\ge 2$ and $1\le\gamma\le 1+\frac{1}{10n^2}$, any efficiently computable function $f:\mathbb{Z}^+\mapsto\mathbb{Z}^+$, and*

$$r := n^{-2f(n)}\max_{i\in[n]}\|\widetilde{\mathbf{b}}_i\| ,$$

there exists $k\in[n]$ such that if $\mathcal{L}' := \mathcal{L}(\mathbf{b}_1,\ldots,\mathbf{b}_{k-1})$, then $\|\widetilde{\mathbf{b}}_k\| \ge \gamma\cdot\frac{\mu(\mathcal{L})}{n^{2f(n)}}$ and

$$\left|\{\mathbf{c}\in\mathcal{L}/\mathcal{L}' \;:\; \operatorname{dist}(\mathbf{t},\mathbf{c})^2 < \operatorname{dist}(\mathbf{t},\mathcal{L})^2 + r^2\}\right| \le \begin{cases} 2^{n-k+1} & : \text{ if } n - f(n) < k \le n \\ 2^{n-k+2}n^{n/f(n)} & : \text{ otherwise} \end{cases}$$

Furthermore, the index k can be computed efficiently from the $\mathbf{b}_i$.

Proof: Let $R := \max_{i\in[n]}\|\widetilde{\mathbf{b}}_i\| = n^{2f(n)}r$. Define $m_j\in[n]$ for $0\le j<2f(n)$ to be the smallest index i such that $\|\widetilde{\mathbf{b}}_i\| \ge \gamma\cdot\frac{R}{n^j}$. Then, by definition, we have that $m_0\ge m_1\ge\cdots\ge m_{2f(n)-1}$. Furthermore,

$$\begin{aligned} r^2 + \frac{m_j-1}{2}\cdot\sum_{i=1}^{m_j-1}\|\widetilde{\mathbf{b}}_i\|^2 &< R^2\cdot\left(\frac{1}{n^{4f(n)}} + \gamma^2\cdot\frac{(m_j-1)^2}{n^{2j}}\right) \\ &\le \frac{R^2}{n^{2j}}\cdot\left(\frac{1}{n^{4f(n)-2j}} + \gamma^2\cdot(n-1)^2\right) \\ &< \frac{R^2}{n^{2j-2}} . \end{aligned} \tag{8}$$

First, consider the case when there exists $j\le f(n)$ such that $m_j = m_{j-1}$. In this case, we claim that the required index is $k = m_j$. To see this, simply note that $\|\widetilde{\mathbf{b}}_k\| \ge \gamma\cdot\frac{R}{n^{j-1}}$ by definition. Then, by Eq. (8), the conditions for Lemma 28 are satisfied with $\ell = k$ and $s = n$. Applying Lemma 3 gives $\|\widetilde{\mathbf{b}}_k\| > \gamma\frac{\mu(\mathcal{L})}{n^{2f(n)}}$, as needed.

So, it is suffices to assume that $m_0 > m_1 > \cdots > m_{f(n)}$. In this case, clearly $m_{f(n)} \leq n - f(n)$. Now, by the pigeonhole principle, there exists $j \in \{f(n), f(n)+1, \ldots, 2f(n)-1\}$ such that $m_{j-1} - m_j < \frac{n}{f(n)}$. Then, let $k = m_j$, and $\ell = m_{j-1}$. Noting the fact that $\|\widetilde{\mathbf{b}}_k\| \geq \frac{R}{n^j}$ and $\|\widetilde{\mathbf{b}}_\ell\| \geq \frac{R}{n^{j-1}}$, the bound on the number of shifts follows from Lemma 28 and Eq. (8). The result again follows from applying Lemma 3. ∎

D. The reduction

With this, we can present our more general reduction. We note in passing that, if the nCVP oracle happens to output a vector in each coset that contains an exact closest vector, then (a minor modification of) our reduction actually finds all closest vectors.

Theorem 30. *For any constant $\delta \in [0,1)$, there is a reduction from exact CVP to γ-nCVP^p where $\gamma(n) := 1 + 1/(10n^{2n^\delta})$ such that the maximal number of oracle calls that the reduction makes on lattices of dimension d when the input lattice has dimension n is*

$$g(n,d) \leq \min\left\{2^{n-d+O(n^{2-2\delta}\log n)},\ \mathrm{poly}(n)\prod_{i=d+1}^{n} p(i)\right\};$$

and the running time of the reduction is $\mathrm{poly}(n)\cdot\sum_d p(d)g(n,d)$.

Proof: The reduction behaves quite similarly to the simple procedure from Lemma 27. The only difference is that this new reduction makes recursive calls on many shifts of $\mathcal{L}'$ corresponding to the many outputs of its γ-nCVP^p oracle. In particular, on input $\mathcal{L} \subset \mathbb{R}^n$ and $\mathbf{t} \in \mathbb{R}^n$, the reduction behaves as follows. First, if $n = 1$, it solves the one-dimensional CVP instance in the obvious way. Otherwise, it uses Theorem 13 and its oracle to compute a γ-HKZ basis $(\mathbf{b}_1, \ldots, \mathbf{b}_n)$ for $\mathcal{L}$. It then calls its oracle on input $\mathcal{L}$ and $\mathbf{t}$ and receives as output $\mathbf{y}_1, \ldots, \mathbf{y}_{\hat{p}} \in \mathcal{L}$.

It then computes the index k as in Lemma 29 with $f(n) := n^\delta$. Let $\mathcal{L}' := \mathcal{L}(\mathbf{b}_1, \ldots, \mathbf{b}_{k-1})$. The reduction groups the $\mathbf{y}_i$ according to their coset mod $\mathcal{L}'$. For each such coset $\mathbf{c}$, it picks an arbitrary representative $\mathbf{y}_\mathbf{c} \in \mathbf{c}$ and calls itself recursively on input $\mathcal{L}'$ and $\mathbf{t} - \mathbf{y}_\mathbf{c}$, receiving as output $\mathbf{x}_\mathbf{c}$. Finally, it outputs the closest $\mathbf{x}_\mathbf{c} + \mathbf{y}_\mathbf{c}$ to $\mathbf{t}$.

Correctness follows immediately from the proof of Lemma 27. In particular, consider a sequence of recursive calls such that the corresponding sequence of $\mathbf{y}_\mathbf{c}$ is a sequence of valid solutions to γ-nCVP^1 and note that the reduction behaves identically to the procedure from Lemma 27 along this sequence.

The statement about the running time is clear. We now analyze the number of recursive calls. Consider a single thread with $\dim \mathcal{L} = n$ and $\dim \mathcal{L}' = \hat{n}$. The total number of recursive calls made by this thread is

$$\begin{aligned} L(n,\hat{n}) &:= \left|\{\mathbf{c} \in \mathcal{L}/\mathcal{L}' \ :\ \exists\ i \text{ with } \mathbf{y}_i \in \mathbf{c}\}\right| \\ &\leq \min\left\{p(n)\ ,\ \left|\{\mathbf{c} \in \mathcal{L}/\mathcal{L}' \ :\ \mathrm{dist}(\mathbf{t},\mathbf{c})^2 < \mathrm{dist}(\mathbf{t},\mathcal{L})^2 + n^{-2f(n)}\}\right|\right\}. \end{aligned} \tag{9}$$

Note that $g(n,d)$ satisfies the recurrence relation

$$g(n,d) \leq \max_{d \leq \hat{n} < n} L(n,\hat{n})g(\hat{n},d)\ , \tag{10}$$

with base case $g(d,d) = \mathrm{poly}(n)$. The bound $g(n,d) \leq \mathrm{poly}(n)\prod_{i=d+1}^{n} p(i)$ follows immediately from the fact that $L(n,\hat{n}) \leq p(n)$.

We wish to prove by induction that for any d and n, we have $g(n,d) \leq 2^{n-d+C^* n^{2-2\delta}\log n}$ for some constant C^*. For $n = 1$ or $d = n$, this is trivial. Suppose that the induction hypothesis holds for dimensions less than n. By Eq. (10), it suffices to prove that $L(n,\hat{n})g(\hat{n},d) \leq 2^{n-d+C^* n^{2-2\delta}\log n}$ for all $\hat{n} < n$. Plugging Lemma 29 into Eq. (9), we have

$$L(n,\hat{n}) \leq \begin{cases} 2^{n-\hat{n}} & :\ \text{if}\ \ n - f(n) \leq \hat{n} < n \\ 2^{n-\hat{n}+1}n^{n/f(n)} & :\ \text{otherwise} \end{cases}.$$

If $\hat{n} \geq n - f(n)$, then by this bound and the induction hypothesis,

$$L(n,\hat{n})g(\hat{n},d) \leq 2^{n-\hat{n}} \cdot g(\hat{n},d) \leq 2^{n-d+C^* n^{2-2\delta}\log n} ,$$

as needed. Otherwise, $\hat{n} < n - f(n)$, and we have

$$\begin{aligned} L(n,\hat{n})g(\hat{n},d) &\leq 2^{n-d+1+C^*\hat{n}^{2-2\delta}\log n} n^{n/f(n)} \\ &\leq 2^{n-d+1+C^*(n-f(n))^{2-2\delta}\log n + n\log_2 n/f(n)} \\ &\leq 2^{n-d+1+C^* n^{2-2\delta} - C^*(2-2\delta)n^{1-2\delta}f(n)\log n + n\log_2 n/f(n)} \\ &\leq 2^{n-d+C^* n^{2-2\delta}} , \end{aligned}$$

as needed. ∎

VI. Finishing the proof

A. The mass of cosets with closest vectors

We now show that our DGS algorithm yields a solution to γ-nCVP^m. (See Definition 26.) Since the number of samples returned by our algorithm is essentially the number that we need to "see each coset with relatively high Gaussian mass," it will suffice to show that cosets of $2\mathcal{L} - \mathbf{t}$ that contain a shortest vector have high mass. Instead, we are only able to prove the slightly weaker (but still sufficient) fact that for a suitable list of parameters $s_1, \ldots, s_\ell$, each such coset has high mass with respect to the discrete Gaussian with at least one of these parameters. (See Corollary 33.)

Lemma 31. *Let $\mathcal{L} \subset \mathbb{R}^n$ be a lattice and $\mathbf{t} \in \mathbb{R}^n$ with $\mathbf{y} \in \mathcal{L}$ a closest vector to $\mathbf{t}$ in $\mathcal{L}$. Then, for any $s > 0$,*

$$1 \leq \frac{\max_{\mathbf{c}\in\mathcal{L}/(2\mathcal{L})} \rho_s(\mathbf{c}-\mathbf{t})}{\rho_s(\mathbf{y}-\mathbf{t})\cdot\rho_s(2\mathcal{L})} \leq \frac{\prod_{j=1}^{\infty}\rho_{2^{-j/2}s}(\mathcal{L})^{1/2^j}}{\rho_s(2\mathcal{L})} \leq 2^{n/4} .$$

Proof: The first inequality trivially follows from Lemma 6. Let $\theta(i) := \rho_{2^{-i/2}s}(\mathcal{L})$ and $\phi(i) := \max_{\mathbf{c}\in\mathcal{L}/(2\mathcal{L})} \rho_{2^{-i/2}s}(\mathbf{c}-\mathbf{t})$. By Corollary 16, we have

$$\phi(i) \leq \phi(i+1)^{1/2}\theta(i+1)^{1/2} .$$

Applying this inequality k times, we have

$$\phi(0) \leq \phi(k)^{1/2^k} \cdot \prod_{j=1}^{k}\theta(j)^{1/2^j} .$$

We take the limit as $k \to \infty$. Since $\mathbf{y} \in \mathcal{L}$ is a closest vector to $\mathbf{t}$, we have

$$\lim_{k\to\infty} \phi(k)^{1/2^k} = \rho_s(\mathbf{y}-\mathbf{t}) .$$

The second inequality is then immediate. For the third inequality, note that for all $i \geq 2$, $\theta(i) \leq \theta(2) = \rho_s(2\mathcal{L})$, and by Lemma 5, $\theta(1) \leq 2^{n/2}\theta(2)$. Therefore,

$$\prod_{j=1}^{\infty}\theta(j)^{1/2^j} \leq 2^{n/4} \cdot \prod_{j=1}^{\infty}\theta(2)^{1/2^j} = 2^{n/4}\cdot\theta(2) .$$

∎

We will need the following technical lemma to obtain a stronger version of Lemma 31.

Lemma 32. *For any lattice $\mathcal{L} \subset \mathbb{R}^n$, $s > 0$, and integer $\ell > 0$, there exists an integer $1 \leq i \leq \ell$ such that*

$$\frac{\prod_{j=1}^{\infty}\rho_{2^{-(i+j)/2}s}(\mathcal{L})^{1/2^j}}{\rho_{2^{-i/2}s}(2\mathcal{L})} \leq 2^{\frac{3n}{4\ell}} .$$

Proof: For $i \geq 0$, let $\theta(i) := \rho_{2^{-i/2}s}(\mathcal{L})$ as in the previous proof. Let

$$S_i := \frac{\prod_{j=1}^{\infty} \theta(i+j)^{1/2^j}}{\theta(i+2)} ,$$

and

$$R_i := \frac{\theta(i+1)}{\theta(i+2)} .$$

We need to show that there exists an integer $1 \leq i \leq \ell$ such that $S_i \leq 2^{3n/4\ell}$.

By Lemma 31, we have that for all i, $1 \leq S_i \leq 2^{n/4}$, and by Lemma 5, we have that, $1 \leq R_i \leq 2^{n/2}$. Note that

$$\frac{S_i^2}{S_{i+1}} = \frac{\theta(i+1) \cdot \theta(i+3)}{\theta(i+2)^2} = \frac{R_i}{R_{i+1}} .$$

Therefore,

$$2^{n/2} \geq \frac{R_0}{R_{\ell+1}} = \prod_{i=0}^{\ell} \frac{R_i}{R_{i+1}} = \prod_{i=0}^{\ell} \frac{S_i^2}{S_{i+1}} = \frac{S_0^2}{S_{\ell+1}} \prod_{i=1}^{\ell} S_i \geq \frac{1}{2^{n/4}} \prod_{i=1}^{\ell} S_i ,$$

where the first inequality uses $R_0 \leq 2^{n/2}$ and $R_{\ell+1} \geq 1$, and the last inequality uses $S_0 \geq 1$ and $S_{\ell+1} \leq 2^{n/4}$. The result then follows. ∎

Finally, we have the following corollary, which follows immediately from Lemmas 31 and 32, and Lemma 6. The corollary shows that, if $\mathbf{c} \in \mathcal{L}/(2\mathcal{L})$ contains a closest vector to $\mathbf{t}$ and we sample from $D_{\mathcal{L}-\mathbf{t},s}$ for many different values of s, then $\mathbf{c} - \mathbf{t}$ will have significantly higher weight than Lemma 31 guarantees for at least one parameter s.

Corollary 33. *For any lattice $\mathcal{L} \subset \mathbb{R}^n$ and $\mathbf{t} \in \mathbb{R}^n$, let $\mathbf{y} \in \mathcal{L}$ a closest vector to $\mathbf{t}$ in $\mathcal{L}$. Then, for any $s > 0$ and integer $\ell > 0$, there exists an integer $1 \leq i \leq \ell$ such that*

$$1 \leq \frac{\max_{\mathbf{c} \in \mathcal{L}/(2\mathcal{L})} \rho_{2^{-i/2}s}(\mathbf{c} - \mathbf{t})}{\rho_{2^{-i/2}s}(2\mathcal{L} + \mathbf{y} - \mathbf{t})} \leq \frac{\max_{\mathbf{c} \in \mathcal{L}/(2\mathcal{L})} \rho_{2^{-i/2}s}(\mathbf{c} - \mathbf{t})}{\rho_{2^{-i/2}s}(\mathbf{y} - \mathbf{t}) \cdot \rho_{2^{-i/2}s}(2\mathcal{L})} \leq 2^{\frac{3n}{4\ell}} .$$

B. The nCVP algorithm

With Corollary 33, it is almost immediate that the algorithm from Theorem 22 yields a solution to nCVP. Below, we make this formal.

Theorem 34. *For any efficiently computable function $f(n) \geq n^{\omega(1)}$, there is an algorithm that solves γ-nCVP^p with probability at least $1 - 2^{-Cn^2}$ in time $2^{n+O(\log n \log f(n) + n/\log f(n))}$, where $\gamma(n) := 1 + 1/f(n)$ and $p(n) := 2^{n+O(n/\log f(n))}$.*

Proof: On input a lattice $\mathcal{L} \subset \mathbb{R}^n$ and shift $\mathbf{t} \in \mathbb{R}^n$, the algorithm first calls the procedure from Corollary 23 to compute $\tilde{d}$ with $\mathrm{dist}(\mathbf{t}, \mathcal{L})/2 \leq \tilde{d} \leq \mathrm{dist}(\mathbf{t}, \mathcal{L})$. Let $s := \tilde{d}/(n^3 f(n))$. For $i = 0, \ldots, \ell := \lceil \log 10 f(n) \rceil$, the algorithm runs the procedure from Theorem 22 $n^2 \cdot \lceil 2^{n/\ell} \rceil$ times with input $\mathcal{L}$, $\mathbf{t}$, and $s_i := 2^{-i/2}s$, receiving as output a total of $\hat{m}_i \geq n^2 2^{n/\ell} \cdot m(\mathcal{L} - \mathbf{t}, s_i)$ vectors $(\mathbf{X}_{i,1}, \ldots, \mathbf{X}_{i,\hat{m}_i}) \in \mathcal{L} - \mathbf{t}$. (We may assume that $\hat{m}_i \leq n^2 2^n \cdot \lceil 2^{n/\ell} \rceil$, since we can trivially truncate the output of each run at $2^n \geq m(\mathcal{L} - \mathbf{t}, s_i)$ vectors.) For each i, j, let $\mathbf{y}_{i,j} := \mathbf{X}_{i,j} + \mathbf{t} \in \mathcal{L}$. Finally, the algorithm outputs the $\mathbf{y}_{i,j}$.

The running time is dominated by the running time of the $\ell n^2 2^{n/\ell}$ applications of Theorem 22. So, the algorithm runs in time $\ell n^2 2^{n+O(\log n \log f(n)) + n/\ell} = 2^{n+O(\log n \log f(n) + n/\log f(n))}$. The value for $p(n)$ follows from the assumed bound on $\hat{m}_i$.

To prove correctness, first note that by Theorem 22, up to statistical distance 2^{-Cn^2}, we may assume that the $\mathbf{X}_{i,j}$ are distributed exactly as independent discrete Gaussians $D_{\mathcal{L}-\mathbf{t},s_i}$. Then, by Corollary 8, all of the output vectors are γ-approximate closest vectors except with probability at most 2^{-Cn^2}.

So, it suffices to show that with high probability there is some i, j such that $\mathbf{y}_{i,j}$ is in the same coset mod $2\mathcal{L}$ as a closest vector $\bar{\mathbf{y}} \in \mathcal{L}$ to $\mathbf{t}$. Fix i as in Corollary 33. Then, for any j,

$$\begin{aligned}
\Pr[\mathbf{y}_{i,j} \equiv \bar{\mathbf{y}} \pmod{2\mathcal{L}}] &= \frac{\rho_{s_i}(2\mathcal{L} + \bar{\mathbf{y}} - \mathbf{t})}{\rho_{s_i}(\mathcal{L} - \mathbf{t})} \\
&= \frac{1}{m(\mathcal{L} - \mathbf{t}, s_i)} \cdot \frac{\rho_{s_i}(2\mathcal{L} + \bar{\mathbf{y}} - \mathbf{t})}{\max_{\mathbf{c} \in \mathcal{L}/(2\mathcal{L})} \rho_{s_i}(\mathbf{c} - \mathbf{t})} \\
&\geq \frac{n^2 2^{n/\ell}}{\hat{m}_i} \cdot 2^{-\frac{3n}{4\ell}} && \text{(Corollary 33)} \\
&> \frac{2n^2}{\hat{m}_i}.
\end{aligned}$$

The result follows by recalling that the $\mathbf{y}_{i,j}$ are independent. ∎

We obtain our main result as a corollary. We note in passing that a simple union bound shows that the algorithm from Theorem 34 actually finds a vector in *each* coset that contains a closest vector. Together with the remark above Theorem 30, this shows that we can actually find *all* closest vectors in time $2^{n+o(n)}$.

Corollary 35. *There is an algorithm that solves* exact *CVP (with high probability) in time* $2^{n+O(n^{2/3}\log^2 n)}$.

Proof: Combine the algorithm from Theorem 34 with $f(n) := 2^{2n^{2/3}\log n}$ with the reduction from Theorem 30 with $\delta := 2/3$. (By applying a union bound over all oracle calls in the reduction, we see that the error is not an issue.) ∎

ACKNOWLEDGMENTS

We thank Oded Regev and Alexander Golovnev for many helpful conversations. We also thank the anonymous FOCS reviewers for their valuable comments. The work of the third author was partially supported by by the National Science Foundation under Grant No. CCF-1320188. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

[1] D. Micciancio and P. Voulgaris, "A deterministic single exponential time algorithm for most lattice problems based on Voronoi cell computations," *SIAM Journal on Computing*, vol. 42, no. 3, pp. 1364–1391, 2013.
[2] D. Aggarwal, D. Dadush, O. Regev, and N. Stephens-Davidowitz, "Solving the Shortest Vector Problem in 2^n time via discrete Gaussian sampling," in *STOC*, 2015, full version available at http://arxiv.org/abs/1412.7994.
[3] A. K. Lenstra, H. W. Lenstra, Jr., and L. Lovász, "Factoring polynomials with rational coefficients," *Math. Ann.*, vol. 261, no. 4, pp. 515–534, 1982. [Online]. Available: http://dx.doi.org/10.1007/BF01457454
[4] H. W. Lenstra Jr, "Integer programming with a fixed number of variables," *Mathematics of operations research*, vol. 8, no. 4, pp. 538–548, 1983.
[5] R. Kannan, "Minkowski's convex body theorem and integer programming," *Mathematics of Operations Research*, vol. 12, no. 3, pp. pp. 415–440, 1987. [Online]. Available: http://www.jstor.org/stable/3689974
[6] D. Dadush, C. Peikert, and S. Vempala, "Enumerative lattice algorithms in any norm via M-ellipsoid coverings," in *Foundations of Computer Science (FOCS), 2011 IEEE 52nd Annual Symposium on*. IEEE, 2011, pp. 580–589.
[7] A. M. Odlyzko, "The rise and fall of knapsack cryptosystems," *Cryptology and computational number theory*, vol. 42, pp. 75–88, 1990.
[8] A. Joux and J. Stern, "Lattice reduction: A toolbox for the cryptanalyst," *Journal of Cryptology*, vol. 11, no. 3, pp. 161–185, 1998.
[9] P. Q. Nguyen and J. Stern, "The two faces of lattices in cryptology," in *Cryptography and lattices*. Springer, 2001, pp. 146–180.
[10] S. Landau and G. L. Miller, "Solvability by radicals is in polynomial time," in *Proceedings of the fifteenth annual ACM symposium on Theory of computing*. ACM, 1983, pp. 140–151.
[11] M. J. Coster, A. Joux, B. A. LaMacchia, A. M. Odlyzko, C.-P. Schnorr, and J. Stern, "Improved low-density subset sum algorithms," *computational complexity*, vol. 2, no. 2, pp. 111–128, 1992.
[12] M. Ajtai, "Generating hard instances of lattice problems," in *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*. ACM, 1996, pp. 99–108.
[13] D. Micciancio and O. Regev, "Worst-case to average-case reductions based on Gaussian measures," *SIAM Journal on Computing*, vol. 37, no. 1, pp. 267–302, 2007.
[14] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *STOC'09—Proceedings of the 2009 ACM International Symposium on Theory of Computing*. ACM, New York, 2009, pp. 169–178.
[15] O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," *Journal of the ACM*, vol. 56, no. 6, pp. Art. 34, 40, 2009. [Online]. Available: http://dx.doi.org/10.1145/1568318.1568324

[16] Z. Brakerski and V. Vaikuntanathan, "Efficient fully homomorphic encryption from (standard) LWE," in *FOCS*. IEEE, 2011, pp. 97–106. [Online]. Available: http://dx.doi.org/10.1109/FOCS.2011.12

[17] Z. Brakerski, A. Langlois, C. Peikert, O. Regev, and D. Stehlé, "Classical hardness of learning with errors," in *STOC*, 2013, pp. 575–584. [Online]. Available: http://doi.acm.org/10.1145/2488608.2488680

[18] Z. Brakerski and V. Vaikuntanathan, "Lattice-based FHE as secure as PKE," in *ITCS*, 2014, pp. 1–12. [Online]. Available: http://doi.acm.org/10.1145/2554797.2554799

[19] S. Arora, L. Babai, J. Stern, and Z. Sweedyk, "The hardness of approximate optima in lattices, codes, and systems of linear equations," in *Foundations of Computer Science, 1993. Proceedings., 34th Annual Symposium on*. IEEE, 1993, pp. 724–733.

[20] M. Ajtai, "The shortest vector problem in ℓ_2 is NP-hard for randomized reductions," in *Proceedings of the thirtieth annual ACM symposium on Theory of computing*. ACM, 1998, pp. 10–19.

[21] J.-Y. Cai and A. Nerurkar, "Approximating the SVP to within a factor $(1+1/\dim^{\varepsilon})$ is NP-hard under randomized conditions," in *Computational Complexity, 1998. Proceedings. Thirteenth Annual IEEE Conference on*. IEEE, 1998, pp. 46–55.

[22] J. Blömer and J.-P. Seifert, "On the complexity of computing short linearly independent vectors and short bases in a lattice," in *Proceedings of the thirty-first annual ACM symposium on Theory of computing*. ACM, 1999, pp. 711–720.

[23] I. Dinur, G. Kindler, R. Raz, and S. Safra, "Approximating CVP to within almost-polynomial factors is NP-hard," *Combinatorica*, vol. 23, no. 2, pp. 205–243, 2003.

[24] D. Micciancio, "The shortest vector problem is NP-hard to approximate to within some constant," *SIAM Journal on Computing*, vol. 30, no. 6, pp. 2008–2035, Mar. 2001, preliminary version in FOCS 1998.

[25] S. Khot, "Hardness of approximating the shortest vector problem in lattices," *Journal of the ACM*, vol. 52, no. 5, pp. 789–808, Sep. 2005, preliminary version in FOCS'04.

[26] I. Haviv and O. Regev, "Tensor-based hardness of the shortest vector problem to within almost polynomial factors," *Theory of Computing*, vol. 8, no. 23, pp. 513–531, 2012, preliminary version in STOC'07.

[27] O. Goldreich, D. Micciancio, S. Safra, and J.-P. Seifert, "Approximating shortest lattice vectors is not harder than approximating closest lattice vectors," *Information Processing Letters*, vol. 71, no. 2, pp. 55 – 61, 1999.

[28] C. Dubey and T. Holenstein, "Approximating the closest vector problem using an approximate shortest vector oracle," in *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*. Springer, 2011, pp. 184–193.

[29] D. Micciancio, "Efficient reductions among lattice problems," in *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms*. ACM, New York, 2008, pp. 84–93.

[30] J. Blömer and S. Naewe, "Sampling methods for shortest vectors, closest vectors and successive minima," *Theoret. Comput. Sci.*, vol. 410, no. 18, pp. 1648–1665, 2009. [Online]. Available: http://dx.doi.org/10.1016/j.tcs.2008.12.045

[31] B. Helfrich, "Algorithms to construct Minkowski reduced and Hermite reduced lattice bases," *Theoret. Comput. Sci.*, vol. 41, no. 2-3, pp. 125–139 (1986), 1985. [Online]. Available: http://dx.doi.org/10.1016/0304-3975(85)90067-2

[32] G. Hanrot and D. Stehlé, "Improved analysis of Kannan's shortest lattice vector algorithm (extended abstract)," in *Advances in cryptology—CRYPTO 2007*, ser. Lecture Notes in Comput. Sci. Springer, Berlin, 2007, vol. 4622, pp. 170–186. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-74143-5_10

[33] D. Micciancio and M. Walter, "Fast lattice point enumeration with minimal overhead," in *SODA*, 2015.

[34] M. Ajtai, R. Kumar, and D. Sivakumar, "A sieve algorithm for the shortest lattice vector problem," in *STOC*, 2001, pp. 601–610. [Online]. Available: http://doi.acm.org/10.1145/380752.380857

[35] ——, "Sampling short lattice vectors and the closest lattice vector problem," in *CCC*, 2002, pp. 41–45.

[36] P. Q. Nguyen and T. Vidick, "Sieve algorithms for the shortest vector problem are practical," *J. Math. Cryptol.*, vol. 2, no. 2, pp. 181–207, 2008. [Online]. Available: http://dx.doi.org/10.1515/JMC.2008.009

[37] V. Arvind and P. S. Joglekar, "Some sieving algorithms for lattice problems." in *FSTTCS*, 2008, pp. 25–36.

[38] X. Pujol and D. Stehlé, "Solving the shortest lattice vector problem in time $2^{2.465n}$," *IACR Cryptology ePrint Archive*, vol. 2009, p. 605, 2009.

[39] D. Micciancio and P. Voulgaris, "Faster exponential time algorithms for the shortest vector problem," in *SODA*, 2010, pp. 1468–1480.

[40] G. Hanrot, X. Pujol, and D. Stehlé, "Algorithms for the shortest and closest lattice vector problems," in *Coding and Cryptology*. Springer, 2011, pp. 159–190.

[41] N. Sommer, M. Feder, and O. Shalvi, "Finding the closest lattice point by iterative slicing," *SIAM J. Discrete Math.*, vol. 23, no. 2, pp. 715–731, 2009. [Online]. Available: http://dx.doi.org/10.1137/060676362

[42] P. Klein, "Finding the closest lattice vector when it's unusually close," in *Proceedings of the eleventh annual ACM-SIAM Symposium on Discrete Algorithms*, 2000, pp. 937–941.

[43] C. Gentry, C. Peikert, and V. Vaikuntanathan, "Trapdoors for hard lattices and new cryptographic constructions," in *STOC*, 2008, pp. 197–206.

[44] O. Regev and N. Stephens-Davidowitz, "An inequality for Gaussians on lattices," http://arxiv.org/abs/1502.04796, 2015.

[45] J. C. Lagarias, H. W. L. Jr., and C.-P. Schnorr, "Korkin-Zolotarev bases and successive minima of a lattice and its reciprocal lattice." *Combinatorica*, vol. 10, no. 4, pp. 333–348, 1990.

[46] A. Becker, N. Gama, and A. Joux, "A sieve algorithm based on overlattices," *LMS Journal of Computation and Mathematics*, vol. 17, no. A, pp. 49–70, 2014.

[47] D. Dadush and G. Kun, "Lattice sparsification and the approximate closest vector problem," in *SODA*, 2013.

[48] N. Bonifas and D. Dadush, "Short paths on the Voronoi graph and the closest vector problem with preprocessing," in *SODA*, 2015.

[49] N. Stephens-Davidowitz, "Discrete Gaussian sampling reduces to CVP and SVP," http://arxiv.org/abs/1506.07490, 2015.

[50] V. Vaikuntanathan, Private communication, 2015.

[51] U. Betke, M. Henk, and J. Wills, "Successive-minima-type inequalities," *Discrete & Computational Geometry*, vol. 9, no. 1, pp. 165–175, 1993. [Online]. Available: http://dx.doi.org/10.1007/BF02189316

[52] W. Banaszczyk, "New bounds in some transference theorems in the geometry of numbers," *Mathematische Annalen*, vol. 296, no. 4, pp. 625–635, 1993. [Online]. Available: http://dx.doi.org/10.1007/BF01445125

[53] D. Dadush, O. Regev, and N. Stephens-Davidowitz, "On the Closest Vector Problem with a distance guarantee," in *IEEE 29th Conference on Computational Complexity*, 2014, pp. 98–109, full version available at http://arxiv.org/abs/1409.8063. [Online]. Available: http://dx.doi.org/10.1109/CCC.2014.18

[54] C. Schnorr, "A hierarchy of polynomial time lattice basis reduction algorithms," *Theoretical Computer Science*, vol. 53, pp. 201 – 224, 1987. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0304397587900648