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

Divesh Aggarwal* Daniel Dadush† Noah Stephens-Davidowitz‡§
Divesh.Aggarwal@epfl.ch dadush@cwi.nl 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 [MV13].

We achieve our main result in two steps. First, we show how to modify the sampling algorithm from [ADRS15] 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 [ADRS15], 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 near-closest vectors to a target vector $\mathbf{t}$ can be grouped into "lower-dimensional clusters," and we use this to obtain a recursive algorithm based on our sampler that solves *exact* CVP in $2^{n+o(n)}$ time.

The analysis of both steps depends crucially on some new properties of the discrete Gaussian distribution, which might be of independent interest.

Keywords. Discrete Gaussian, Closest Vector Problem, Lattice Problems.

*Department of Computer Science, EPFL.
†Centrum Wiskunde & Informatica, Amsterdam.
‡Courant Institute of Mathematical Sciences, New York University.
§This material is based upon work supported 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.

1 Introduction

A lattice $\mathcal{L}$ is defined as the set of all integer combinations of some 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 is to compute a non-zero vector in $\mathcal{L}$ of minimal length, and CVP is to compute a lattice vector nearest in Euclidean distance to a target vector $\mathbf{t}$.

Starting with the seminal work of [LLL82], algorithms for solving these problems either exactly or approximately have been studied intensely. Such algorithms have found applications in factoring polynomials over rationals [LLL82], integer programming [LJ83, Kan87, DPV11], cryptanalysis [Odl90, JS98, NS01], checking the solvability by radicals [LM83], and solving low-density subset-sum problems [CJL$^+$92]. More recently, several powerful cryptographic primitives have been constructed whose security is based on the *worst-case* hardness of these or related lattice problems [Ajt96, MR07, Gen09, Reg09, BV11, BLP$^+$13, BV14].

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 [ABSS93, Ajt98, CN98, BS99, DKRS03, Mic01, Kho05, HR12]. 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 [GMSS99], even in the approximate case, while there is no known reduction in the other direction that preserves the dimension.[1] Indeed, CVP is in some sense nearly "complete for lattice problems," as there are known dimension-preserving reductions from 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 [Mic08, BN09, MV13]. (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 [Kan87], and many others improved upon his technique to lower the constant in the exponent [Hel85, HS07, MW15]. 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 [AKS01]. 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 [AKS01, AKS02, NV08, AJ08, BN09, PS09, MV10, HPS11], culminating in a $\widetilde{O}(2^{2.456n})$-time algorithm for SVP and a $2^{O(n)} \cdot (1 + 1/\varepsilon)^{O(n)}$-time algorithm for $(1+\varepsilon)$-approximate CVP.

But, exact CVP is a much more subtle problem than approximate CVP. 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

[1] Since both problems are NP-complete, there is necessarily an efficient reduction from CVP to SVP. However, all known reductions either blow up the approximation factor or the dimension of the lattice by a polynomial factor [Kan87, DH11]. Since we are interested in an algorithm for solving exact CVP whose running time is exponential in the dimension, such reductions are not useful for us.

between" γ-approximate closest vectors and exact closest vectors for 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 [MV13] (MV), which built upon the approach of Sommer, Feder, and Shalvi [SFS09]. 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 [ADRS15]. 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)?

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 two 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 that approximates CVP to within any approximation factor $\gamma = 1+2^{-o(n/\log n)}$. Second, we show that the near-closest vectors to a target can be grouped into "lower-dimensional clusters" and use this to obtain a recursive algorithm that solves *exact* CVP in $2^{n+o(n)}$ time. We find this result to be quite surprising as, in spite of much research in this area, all previous randomized algorithms only gave approximate solutions to CVP. Indeed, this barrier seemed inherent, as we described above. Our solution depends crucially on new properties of the discrete Gaussian distribution and on the specific behavior of our sampling algorithm.

1.1 Our techniques

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* [ADRS15, 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 [GPV08, BLP+13]) 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. It is easy to see that the central coset, $2\mathcal{L}$, has maximal weight proportional to $\rho_{s/2}(\mathcal{L})$, and ADRS show that the collision probability is proportional to $\rho_{s/\sqrt{2}}(\mathcal{L})^2$. Indeed, the loss factor for a single step is given by $\beta(s) = \rho_{s/\sqrt{2}}(\mathcal{L})^2/(\rho_s(\mathcal{L})\rho_{s/2}(\mathcal{L}))$. Therefore, the *total* loss factor $\beta(s)\beta(s/\sqrt{2})\cdots\beta(s/2^{-\ell/2})$ accumulated after running the combiner ℓ times is given by a telescoping product, which is easily bounded by $2^{-n/2-o(n)}$. So, their sampler returns at least $2^{-n/2-o(n)} \cdot M$ 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 4.1 and Proposition 4.2.) 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 $\beta(s)\beta(s/\sqrt{2})\cdots\beta(s/2^{-\ell/2})$. While the loss factor for a single run of the combiner $\beta(s)$ is again equal to the ratio of the collision probability over the cosets to the maximal weight of a coset, this ratio does not seem to have such a nice representation in the shifted case. (See Corollary 4.2.) In particular, it is no longer clear which coset has maximal weight, and this coset can even vary with s! To solve this problem, we first introduce a new inequality (Corollary 3.3), which relates the maximal weight of a coset with parameter s to the maximal weight of a coset with parameter $s/\sqrt{2}$.[2] We then show how to use this inequality to inductively bound the accumulated loss factor by $2^{-n-o(n)}$. So, we only need to start out with $2^{n+o(n)}$ 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 4.8.) In order to obtain our $2^{n+o(n)}$-time algorithm for *exact* CVP, we observe that the loss factor is often quite a bit better than this. In fact, as we explain below, we obtain essentially the precise number of vectors that we need in order for our $2^{n+o(n)}$-time exact CVP algorithm to work! (We find this fact to be quite surprising, and we have no satisfying explanation for it.)

Finally, we note that our DGS solution actually works for parameters that are much lower than we need for our exact CVP algorithm. In particular, we show how to sample from $D_{\mathcal{L}-\mathbf{t},s}$ for any $s \geq \mathrm{dist}(\mathbf{t},\mathcal{L})/2^{o(n/\log n)}$ (see Theorem 4.7), but we only need such samples for $s \approx \max_{\mathbf{u}\in\mathbb{R}^n} \mathrm{dist}(\mathbf{u},\mathcal{L})/2^{o(n/\log n)}$. So, when $\mathbf{t}$ is much closer to the lattice than the farthest possible point, our sampler works for much lower parameters than we need. It takes slightly more work to achieve the stronger result, but we include it in case it finds other applications.

A reduction from CVP to DGS. Note that samples from $D_{\mathcal{L}-\mathbf{t},s}$ immediately yield approximate closest vectors to $\mathbf{t}$. In particular, if $\mathbf{X} \sim D_{\mathcal{L}-\mathbf{t},s}$, then $\mathbf{X}$ should be concentrated on relatively short vectors, so that $\mathbf{X} + \mathbf{t} \in \mathcal{L}$ is likely to be fairly close to $\mathbf{t}$. (See Corollary 2.7 for the concentration bound. This idea is frequently used in cryptography, though with much different parameters. E.g., [GPV08].) But, as we described above, it seems much harder to obtain an *exact* closest vector.

[2] This inequality is closely related to that of [RS15], and it (or the more general Lemma 3.2) may be of independent interest. Indeed, we use it in two seemingly unrelated contexts in the sequel—to bound the loss factor of the sampler, and to show that our recursive CVP algorithm will find an exact closest vector with high probability.

In order to solve this harder problem, 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 *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 relatively small ball whose radius depends on the approximation factor. (See Lemma 5.1.) Indeed, for low enough approximation factors, the clusters are contained in at most 2^n *shifts* of a sublattice $\mathcal{L}'$ with dimension strictly less than n. (See Corollary 5.2.) If we could find a cluster that contains a closest lattice vector to $\mathbf{t}$, then we could solve CVP recursively "inside the shift of $\mathcal{L}'$," and we would have an *exact* CVP algorithm.

So, our first technical challenge is to find a cluster that contains a closest lattice vector to $\mathbf{t}$. One might hope that "not-too-many" samples from $D_{\mathcal{L}-\mathbf{t},s} + \mathbf{t}$ would be guaranteed to land in such a cluster, but it is not hard to find examples in which clusters containing closest points to $\mathbf{t}$ have very low weight relative to clusters that merely contain many approximate closest points. However, we can show that this "cannot happen for too many parameters s." In particular, if we sample from many properly chosen parameters $s_0, \ldots, s_\ell$, then for one of these parameters, the weight of a cluster containing a near-closest vector will not be much smaller than the maximal weight of a cluster. (See Corollary 3.6, which follows from Corollary 3.3.)

So, a sample from $D_{\mathcal{L}-\mathbf{t},s_i} + \mathbf{t}$ for one of the s_i will be almost as likely to land in a cluster containing a closest vector as it is to land in any other cluster. But, there could be as many as 2^n distinct clusters with nearly equal weight, and it is possible that only one of them contains a closest lattice vector. In such a case, we would clearly need 2^n samples to find the correct cluster, but our sampler is only guaranteed to yield a single sample in general. We could solve this by simply running the sampler 2^n times, and this would lead to a $4^{n+o(n)}$-time algorithm (essentially matching MV). But, as we mentioned above, more careful analysis of our sampler shows that it essentially returns exactly the number of vectors that we need! In particular, the number of vectors returned by our sampler is essentially the ratio of the total Gaussian mass of the (shifted) lattice divided by the mass of the maximum cluster. (See Theorem 4.7.) As an example, if we sample from the discrete Gaussian over $\mathbb{Z}^n - (1/2, \ldots, 1/2)$ (e.g., to find vectors close to $(1/2, \ldots, 1/2)$ in $\mathbb{Z}^n$), then coset mod $2\mathcal{L}$ has exactly the same weight for every parameter s by symmetry, and hence the total loss factor is 1 instead of 2^{-n} (ignoring small factors)! So, by running the sampler N times, we can obtain enough samples to "see" all clusters whose mass is within a factor of roughly N of maximal. By choosing N appropriately, we can find a cluster containing a closest point to $\mathbf{t}$ with high probability. We can then solve CVP recursively "inside each cluster" that appears in our samples, and obtain an exact closest vector.

Our final technical challenge is to bound the running time of the algorithm. We do this by bounding the number of recursive calls that we must make relative to the dimension of $\mathcal{L}'$. By the argument above, all our samples correspond to very good approximations of the closest vector, and hence they land in at most 2^n shifts of $\mathcal{L}'$ (one for each coset of $2\mathcal{L}$). Since we only make recursive calls on shifts of $\mathcal{L}'$ that appear in our Gaussian samples, this shows that the algorithm makes at most 2^n recursive calls, which shows that the running time is at most $2^{O(n^2)}$. If we could show that we only need to make 2^{n-k} recursive calls where $k := \dim \mathcal{L}'$, then the algorithm would run in $2^{n+o(n)}$ time as needed. Unfortunately, this is not quite the case. But, we show that we can always choose $\mathcal{L}'$ so that either (1) all near-closest vectors are contained in at most 2^{n-k} shifts of $\mathcal{L}'$; or (2) k is relatively small and all near-closest vectors are contained in "not too many more" than 2^{n-k} shifts of $\mathcal{L}'$. (See Lemma 5.5.) Since $k < n$ is decreasing, the second case cannot happen "too often," and this suffices to obtain a $2^{n+o(n)}$-time algorithm.

Finally, we note that our algorithm actually returns *all* closest vectors to the target, not just one. (Micciancio and Voulgaris note that their algorithm can be easily modified to do the same.) Since there are examples in which there are 2^n closest lattice vectors to the target (e.g., $\mathcal{L} = \mathbb{Z}^n$ and $\mathbf{t} = (1/2, 1/2, \ldots, 1/2)$), our running time is trivially optimal up to a factor of $2^{o(n)}$ in the worst case for this harder problem.

1.2 Comparison to prior 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 a Hermite-Korkine-Zolatoreff (HKZ) basis and "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 [Kan87, Hel85, LJS90, HS07, MW15], 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., [BGJ14].) The main reason for this is that such techniques 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 sampler, we never need to project, and we are therefore able to ignore these useless points (they simply do not come up, except with negligible probability) 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 near-closest lattice vectors form clusters, mostly in the context of AKS-like sieving algorithms. For example, the $(1 + \varepsilon)$-approximate closest vectors can be grouped into $2^{O(n)}(1 + 1/\varepsilon)^n$ clusters of diameter ε times the target distance (see, e.g., [AJ08, DK13]). 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 $\gamma = 1 + 2^{-o(n/\log n)}$. Indeed, with our current analysis, if our sampler could only return $(1 + 1/2^{\sqrt{n}})$-approximate closest vectors, it would not yield a singly exponential running time for exact CVP (we would recurse on too many subproblems). For higher approximation factors and slower running times, our clustering bound is both less natural and less useful.

[BD15] 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.

1.3 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 improving our technique, 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 a $O(2^{(1-\varepsilon)n})$-time algorithm [MV13]. Indeed, as we mentioned above, our techniques and those of MV seem to inherently solve the harder (though likely not very important) problem of finding *all* closest vector. 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 running time of $2^{(1-\varepsilon)n}$ exists, it would likely require substantial new ideas. (We think it is plausible that no such algorithm exists, and we ask whether there is a matching lower bound to our upper bound.)

In this work, we show how to use a technique that seems "inherently approximate" to solve *exact* CVP. E.g., 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 solutions

to CVP achieved only constant approximation factors. Our reduction is a bit delicate, as it relies on many specific properties of the discrete Gaussian distribution and the surprisingly convenient number of samples returned by our sampler, but it is plausible that our techniques could be modified to show how to obtain solutions to *exact* CVP with a sieving algorithm. 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 $\Omega(2^n)$).

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 methods of Kannan, 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 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 completely 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 simply 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.

Finally, this work and [ADRS15] show that Discrete Gaussian Sampling is a lattice problem of central importance. However, DGS for parameters below smoothing is not nearly as well-understood as many other lattice problems, and many natural questions remain open. For example, is there a dimension-preserving reduction from DGS to CVP (even in the centered case)? Is centered DGS NP-hard (under a randomized reduction, of course), and if so, for which parameters? For which parameters is shifted DGS NP-hard?[3]

Organization

In Section 2, we overview the necessary background material and give the basic definitions used throughout the paper. In Section 3, we derive two inequalities, Corollary 3.3 and Corollary 3.6, that will allow us to bound the "loss factor" of our sampler and the running time of our exact CVP algorithm respectively. In Section 4, we present our discrete Gaussian sampler. Finally, in Section 5, we give our exact CVP algorithm, after analyzing the structure of "clusters of near-closest vectors."

2 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 $\mathbb{R}^n$.

2.1 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

[3]The hardness of shifted DGS for small parameters follows from the hardness of approximate CVP with relatively small approximation factors. But, it is not clear how to solve SVP or CVP with small approximation factors with only polynomially many samples from the centered discrete Gaussian. In particular, the reductions in ADRS require exponentially many samples.

$n = d$, 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 $\mathbf{t}$.

Definition 2.1. *For a lattice $\mathcal{L}$, the ith successive minimum of $\mathcal{L}$ is*

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

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 use the following theorem from [BHW93], which bounds the number of lattice points in ball of radius proportional to $\lambda_1(\mathcal{L})$.

Theorem 2.2 ([BHW93, Theorem 2.1])**.** *For any lattice $\mathcal{L} \subset \mathbb{R}^n$ and $s > 0$,*

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

2.2 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 2.3. *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 on the discrete Gaussian [Ban93].

Lemma 2.4 ([Ban93, 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 2.5. *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 2.6 ([DRS14, Lemma 2.13])**.** *For any lattice $\mathcal{L} \subset \mathbb{R}^n$, $s > 0$, $\mathbf{u} \in \mathbb{R}^n$, 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.

Corollary 2.7. *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 $r \geq 1/\sqrt{2\pi}$,*

$$\Pr_{\mathbf{X}\sim D_{\mathcal{L}-\mathbf{t},s}}[\|\mathbf{X}\| \geq rs\sqrt{n}] < e^{\pi n\alpha^2}\left(\sqrt{2\pi e r^2}\exp(-\pi r^2)\right)^n . \tag{1}$$

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

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

Proof. We can assume without loss of generality that $\mathbf{0}$ is a closest vector to $\mathbf{t}$ in $\mathcal{L}$ and therefore $d := \mathrm{dist}(\mathbf{t}, \mathcal{L}) = \|\mathbf{t}\|$. Equation 1 then follows from combining Lemma 2.5 with Lemma 2.6.

Let $r = \sqrt{\alpha^2 + 2n} \geq 1/\sqrt{2\pi}$, and note that $rs\sqrt{n} = \sqrt{d^2 + 2(ns)^2}$. Then, by the first part of the corollary, we have that

$$\begin{aligned}
\Pr[\|\mathbf{X}\|^2 \geq d^2 + 2(sn)^2] &= \Pr[\|\mathbf{X}\| \geq rs\sqrt{n}] \\
&\leq e^{\pi\alpha^2 n} \cdot \left(2\pi e(\alpha^2 + 2n)\right)^{n/2} \cdot e^{-n\pi(\alpha^2+2n)} \\
&\leq \left(4\pi e 2^{2n}\right)^{n/2} e^{-2\pi n^2} \\
&\leq e^{(\ln(4\pi e)/2)n + (\ln 2)n^2 - 2\pi n^2} \\
&\leq e^{-3n^2} ,
\end{aligned}$$

as needed. □

2.3 The Gram-Schmidt orthogonalization and 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 $\mathbf{b}_1, \ldots, \mathbf{b}_{i-1}$. For ease of notation, we write π_i to denote the orthogonal projection onto the subspace $\{\mathbf{b}_1, \ldots, \mathbf{b}_{i-1}\}^\perp$ when the basis in question is clear.

Definition 2.8. *A basis $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ of $\mathcal{L}$ is a Hermite-Korkin-Zolotarev (HKZ) basis if*

1. $\|\mathbf{b}_1\| = \lambda_1(\mathcal{L})$;
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 an HKZ basis of $\pi_{\{\mathbf{b}_1\}^\perp}(\mathcal{L})$.*

We use 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 an 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 $\|\mathbf{y}\| < \|\widetilde{\mathbf{b}}_k\|$.

Lemma 2.9. *For any lattice $\mathcal{L} \subset \mathbb{R}^n$ with basis $(\mathbf{b}_1, \ldots, \mathbf{b}_n)$ and target $\mathbf{t} \in \mathbb{R}^n$,*

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

2.4 Lattice problems

We define the following lattice problems.

Definition 2.10. *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 2.11. *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 non-negative real numbers, ε-DGS_σ^m (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 $D_{\mathcal{L}-\mathbf{t},s}^{\hat{m}}$.*

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

2.5 Some known algorithms

We will also make use of two $2^{n+o(n)}$-time algorithms from [ADRS15]. Though we could instead use significantly faster algorithms that achieve weaker but sufficient results, using these "heavy hammers" will cost us nothing asymptotically and simplify our analysis.

Theorem 2.12 ([ADRS15, Corollary 7.2])**.** *There is an algorithm that solves 1.97-CVP in time $2^{n+o(n)}$.*

Theorem 2.13 ([ADRS15, Corollary 4.5])**.** *There is an algorithm that finds a shortest non-zero lattice vector in time $2^{n+o(n)}$.*

The problem of finding a shortest non-zero vector of a lattice is trivially polynomial-time equivalent to finding an HKZ basis, giving the following useful corollary.

Corollary 2.14. *There is an algorithm that finds an HKZ basis in time $2^{n+o(n)}$.*

We will also need the following algorithm.

Theorem 2.15 ([ADRS15, Theorem 3.3])**.** *There is an algorithm that takes as input $\kappa \geq 2$ (the confidence parameter) and M elements from $\{1, \ldots, N\}$ and outputs a sequence of elements from the same set such that*

1. *the running time is $M \cdot \mathrm{poly}(\log \kappa, \log N)$;*
2. *each $i \in \{1, \ldots, N\}$ appears at least twice as often in the input as in the output; and*
3. *if the input consists of $M \geq 10\kappa^2 / \max p_i$ independent samples from the distribution that assigns probability p_i to element i, then the output is within statistical distance $C_1 MN \log N \exp(-C_2\kappa)$ of m independent samples with respective probabilities $\mathbf{p}^2$ where $m \geq M \cdot \sum p_i^2 / (32\kappa \max p_i)$ is a random variable.*

3 Some inequalities concerning Gaussians on shifted lattices

We first prove an inequality (Corollary 3.3) 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 [RS15], and it (or the more general form given in Lemma 3.2) may have additional applications. The proof follows easily from the following identity from [RS15].

Lemma 3.1 ([RS15, 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 3.2. *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 3.1, 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 3.3. *For any lattice $\mathcal{L} \subset \mathbb{R}^n$, and any $\mathbf{t} \in \mathbb{R}^n$, and $s > 0$, we have*

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

The next lemma shows that if $\mathbf{c} \in \mathcal{L}/(2\mathcal{L})$ contains a closest vector to $\mathbf{t}$, then the Gaussian mass of $\rho(\mathbf{c} - \mathbf{t})$ cannot be too much smaller than the mass of a different coset.

Lemma 3.4. *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 2.5. 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 3.3, 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 2.4, $\theta(1) \leq 2^{n/2}\theta(2)$. 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 3.4.

Lemma 3.5. *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 3.4, we have that for all i, $1 \leq S_i \leq 2^{n/4}$, and by Lemma 2.4, 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 3.4 and 3.5, and Lemma 2.5. 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 3.4 guarantees for at least one parameter s.

Corollary 3.6. *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*

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

4 Sampling from the discrete Gaussian

4.1 Combining discrete Gaussian samples

The following lemma and proposition are the shifted analogues of [ADRS15, Lemma 3.4] and [ADRS15, Proposition 3.5] respectively. Their proofs are nearly identical to the related proofs in [ADRS15], and we include them in the appendix for completeness. (We note that Lemma 4.1 can be viewed as a special case of Lemma 3.1.)

Lemma 4.1. *Let $\mathcal{L} \subset \mathbb{R}^n$, $s > 0$ and $\mathbf{t} \in \mathbb{R}^n$. Then for all $\mathbf{y} \in \mathcal{L} - \mathbf{t}$,*

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

Proposition 4.2. *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 4.3 that by calling the algorithm from Proposition 4.2 repeatedly, we obtain a general discrete Gaussian combiner.

Theorem 4.3. *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 vector 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 4.2 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

$$M_i \geq (32\kappa)^{\ell-i+1} \cdot \frac{\psi(i)}{\phi(i)} \tag{3}$$

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 3.3, we have

$$L(i) \geq \frac{\psi(i+1)}{\phi(i+1)} \cdot \frac{\phi(i)}{\psi(i)} . \tag{4}$$

By Proposition 4.2, 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. (4) with $i = 0$. By noting that $\psi(0) \leq 2^n \phi(0)$, we see that (3) holds when $i = 1$.

Suppose that $\mathcal{X}_i$ has the correct distribution and (3) 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 4.2. 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. (4). □

4.2 Initializing the sampler

In order to use our combiner, we need to start with samples from the discrete Gaussian distribution with some large parameter $\hat{s}$. For very large parameters, the algorithm given by Gentry, Peikert, and Vaikuntanathan [GPV08] suffices. For convenience, we use the following strengthening of their result, which provides exact samples and gives better bounds on the parameter s.

Theorem 4.4 ([BLP+13, Lemma 2.3])**.** *There is a probabilistic polynomial-time algorithm that takes as input a basis $\mathbf{B}$ for a lattice $\mathcal{L} \subset \mathbb{R}^n$ with $n \geq 2$, a shift $\mathbf{t} \in \mathbb{R}^n$, and $\hat{s} > C\sqrt{\log n} \cdot \|\widetilde{\mathbf{B}}\|$ and outputs a vector that is distributed exactly as $D_{\mathcal{L}-\mathbf{t},\hat{s}}$, where $\|\widetilde{\mathbf{B}}\| := \max\|\widetilde{\mathbf{b}}_i\|$.*

When instantiated with an HKZ basis, Theorem 4.4 allows us to sample with parameter $\hat{s} = \mathrm{poly}(n) \cdot \lambda_n(\mathcal{L})$. After running our combiner $o(n/\log n)$ times, this will allow us to sample with any parameter $s = \lambda_n(\mathcal{L})/2^{o(n/\log n)}$. This is already sufficient for our exact CVP algorithm. For completeness, we include the following proposition and corollary, which will allow us to sample with any parameter $s = \mathrm{dist}(\mathbf{t},\mathcal{L})/2^{o(n/\log n)}$. Their proofs are in Appendix C.

Proposition 4.5. *There is a $2^{n+o(n)}$-time algorithm that takes as input a lattice $\mathcal{L} \subset \mathbb{R}^n$, shift $\mathbf{t} \in \mathbb{R}^n$, and $r > 0$, such that if $r \geq 3\,\mathrm{dist}(\mathbf{t},\mathcal{L})$, 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 $r - \mathrm{dist}(\mathbf{t},\mathcal{L})$ are also contained in $\mathcal{L}' - \mathbf{y} - \mathbf{t}$, and $\|\widetilde{\mathbf{B}}'\| \leq r$.*

Corollary 4.6. *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 $\hat{s} > 0$ and outputs $\mathbf{y} \in \mathcal{L}$, a sublattice $\mathcal{L}' \subseteq \mathcal{L}$, and M vectors from $\mathcal{L}' - \mathbf{y} - \mathbf{t}$ such that if $\hat{s} \geq C\sqrt{\log n} \cdot \mathrm{dist}(\mathbf{t},\mathcal{L})$, then the output vectors 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 $C\hat{s}/\sqrt{\log n}$. The algorithm runs in time $2^{n+o(n)} + \mathrm{poly}(n) \cdot M$.*

4.3 The sampler

Theorem 4.7. *For any computable function $f(n) \leq 2^{o(n/\log n)}$, let σ be the function such that for any lattice $\mathcal{L} \subset \mathbb{R}^n$ and shift $\mathbf{t} \in \mathbb{R}^n$, $\sigma(\mathcal{L} - \mathbf{t}) := \mathrm{dist}(\mathbf{t},\mathcal{L})/f(n)$. 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 a $2^{n+o(n)}$-time algorithm solving ε-DGS$^m_\sigma$ with $\varepsilon = 2^{-2n}$.

Proof. 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 4.6 with input $\mathcal{L}$, $\mathbf{t}$, $M = (Cn^2)^{\ell+2} \cdot 2^n$ with $\ell = C\lceil 1 + \log\log n + \log f(n)\rceil$, and $\hat{s} = 2^\ell s > C\sqrt{\log n}\,\mathrm{dist}(\mathbf{t},\mathcal{L})$. 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 4.3 *twice*, first with input $\mathcal{L}'$, ℓ, $\kappa := Cn^2$, $\mathbf{t}$, and $(\mathbf{X}_1,\ldots,\mathbf{X}_{M/2})$, and next with input $\mathcal{L}'$, ℓ, κ, $\mathbf{t}$, and $(\mathbf{X}_{M/2+1},\ldots,\mathbf{X}_M)$. Finally, it outputs the resulting vectors.

The running time follows from the running times of the two subprocedures. By Corollary 4.6, 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 $C\hat{s}/\sqrt{\log n}$. By Theorem 4.3, 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 $d = \mathrm{dist}(\mathbf{t},\mathcal{L})$. The statistical distance is exactly

$$\begin{aligned}\Pr_{\mathbf{w}\sim D_{\mathcal{L}-\mathbf{t},s}}\left[\mathbf{w} \notin \mathcal{L}' - \mathbf{y} - \mathbf{t}\right] &< \Pr_{\mathbf{w}\sim D_{\mathcal{L}-\mathbf{t},s}}\left[\|\mathbf{w}\| > C\hat{s}/\sqrt{\log n}\right] \\ &< e^{\pi d^2/s^2} e^{-C_1 n^{C_2} f(n)} \\ &< 2^{-Cn^2}\,,\end{aligned}$$

where we have used Corollary 2.7. 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 $\varepsilon = 2^{-2n}$ 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 4.7 and Corollary 2.7, we immediately get a weaker version of our main result, a $2^{n+o(n)}$-time algorithm for $(1 + 2^{-o(n/\log n)})$-CVP.

Corollary 4.8. *For any efficiently computable positive function $f(n) \leq 2^{o(n/\log n)}$, there is an algorithm solving $(1 + 1/f(n))$-CVP (with high probability) in time $2^{n+o(n)}$.*

5 Reduction from exact CVP to DGS

We now show that, in order to solve *exact* CVP in time $2^{n+o(n)}$, it suffices to solve ε-DGS$_\sigma^m$ in time $2^{n+o(n)}$, where $\sigma(\mathcal{L} - \mathbf{t}) > \lambda_n(\mathcal{L})/2^{o(n/\log n)}$, and m is essentially "the number of cosets of $2\mathcal{L}$ that contain very close vectors to the target $\mathbf{t}$," as in Theorem 4.7. The next lemma shows that the near-closest vectors to $\mathbf{t}$ form "clusters" such that near-closest vectors in the same coset of $2\mathcal{L}$ must be in the same cluster.

Lemma 5.1. *For any lattice $\mathcal{L} \subset \mathbb{R}^n$, $\mathbf{t} \in \mathcal{L}$, $r_1, r_2 > 0$, 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

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

□

We now derive an immediate corollary, which shows that we can view these clusters as shifts of a lower-dimensional lattice $\mathcal{L}'$, defined in terms of an HKZ basis.

Corollary 5.2. *For any $\mathcal{L} \subset \mathbb{R}^n$ with HKZ basis $(\mathbf{b}_1, \ldots, \mathbf{b}_n)$, $\mathbf{t} \in \mathbb{R}^n$, 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 $\|\mathbf{w}_i\| < \operatorname{dist}(\mathbf{t}, \mathcal{L})^2 + \|\widetilde{\mathbf{b}}_k\|^2$, then $\mathbf{w}_1 \in \mathcal{L}' + \mathbf{w}_2$.*

Proof. Let $2\mathbf{v} = \mathbf{w}_1 - \mathbf{w}_2$. Note that $\mathbf{v} \in \mathcal{L}$ by hypothesis, and by Lemma 5.1, we have that $\|\mathbf{v}\| < \|\widetilde{\mathbf{b}}_k\|$. It follows that $\mathbf{v} \in \mathcal{L}'$, as needed. □

Lemma 5.3. *For any lattice $\mathcal{L} \subset \mathbb{R}^n$, $\mathbf{t} \in \mathbb{R}^n$, $s > 0$ satisfying $\operatorname{dist}(\mathbf{t}, \mathcal{L}) \leq 2^n \cdot \sqrt{n}s$, if $\mathbf{X}_1, \ldots, \mathbf{X}_N$ are independent samples from $D_{\mathcal{L}-\mathbf{t},s}$, then the following holds with probability at least $1 - Ne^{-3n^2}$:*

1. *$\forall i \in [N]$, $\|\mathbf{X}_i\|^2 < d^2 + 2(sn)^2$; and*
2. *$\forall i, j \in [N]$ satisfying $\mathbf{X}_i \equiv \mathbf{X}_j \pmod{2\mathcal{L}}$, $\|\mathbf{X}_i - \mathbf{X}_j\|^2 < (3sn)^2$.*

Proof. By Corollary 2.7 and union bound, we have that $\|\mathbf{X}_i\|^2 < d^2 + 2(sn)^2$ holds for all i with probability at least $1 - Ne^{-3n^2}$. Conditioning on the above, by Lemma 5.1, if $\mathbf{X}_i \equiv \mathbf{X}_j \pmod{2\mathcal{L}}$ then $\|\mathbf{X}_i - \mathbf{X}_j\|^2 < 4 \cdot 2(sn)^2 < (3sn)^2$. The lemma thus follows. □

The next lemma bounds the number of shifts of $\mathcal{L}'$ that can contain approximate closest vectors.

Lemma 5.4. *For any $\mathcal{L} \subset \mathbb{R}^n$ with HKZ basis $(\mathbf{b}_1, \ldots, \mathbf{b}_n)$, $\mathbf{t} \in \mathcal{L}$, and $k \in [n]$, let $\mathcal{L}' := \mathcal{L}(\mathbf{b}_1, \ldots, \mathbf{b}_{k-1})$. If $r > 0$ and $k \leq \ell \leq n + 1$ satisfy*

$$r^2 + \frac{(k-1)}{2} \cdot \sum_{i=1}^{k-1} \|\widetilde{\mathbf{b}}_i\|^2 \leq \begin{cases} 4n^2\|\widetilde{\mathbf{b}}_k\|^2 & : \ell = n+1 \\ \min\{4n^2\|\widetilde{\mathbf{b}}_k\|^2, \|\widetilde{\mathbf{b}}_\ell\|^2\} & : \textit{otherwise} \end{cases}, \tag{5}$$

then we have that

$$\left|\{\mathbf{c} \in \mathcal{L}/\mathcal{L}' \ :\ \operatorname{dist}(\mathbf{t}, \mathbf{c})^2 < \operatorname{dist}(\mathbf{t}, \mathcal{L})^2 + r^2\}\right| \leq 2^{n-k+1}(2(4n)^{\ell-k} - 1)\ .$$

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 cosets of $\mathcal{L}'$ that are subsets of $\mathbf{d}$ and contain 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 $|S_{\mathbf{d}}| \le (2(4n)^{l-k}-1)$ for all $\mathbf{d} \in \mathcal{L}/(2\mathcal{L}+\mathcal{L}')$.

Fix $\mathbf{d} \in \mathcal{L}/(2\mathcal{L}+\mathcal{L}')$. Let $\mathbf{w}_1, \mathbf{w}_2 \in \mathbf{d}-\mathbf{t}$. Suppose $\|\mathbf{w}_i\|^2 < \operatorname{dist}(\mathbf{t},\mathcal{L})^2 + r^2$. Note that there exist coefficients $a_i \in \{0,1\}$ such that $\mathbf{w}_1 - \sum_{i=1}^{k-1} a_i \mathbf{b}_i \equiv \mathbf{w}_2 \pmod{2\mathcal{L}}$. Let $s_i \in \{-1,1\}$ be independent and uniform random variables. Then, we have

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

Since the $\mathbf{b}_i$ are 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, there exist $s_i \in \{-1,1\}$ such that

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

Let $2\mathbf{v} := \mathbf{w}_1 - \sum_{i=1}^{k-1} s_i a_i - \mathbf{w}_2 \in \mathcal{L}$. Since $\mathbf{w}_1 - \sum_{i=1}^{k-1} s_i a_i \mathbf{b}_i \equiv \mathbf{w}_2 \pmod{2\mathcal{L}}$, we may apply Lemma 5.1 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. (5), 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 $\lambda_1(\mathcal{M}) = \|\widetilde{\mathbf{b}}_k\|$ and $\dim \mathcal{M} = \ell - k$. The result then follows from applying Theorem 2.2.

□

Finally, we show how to find a specific index k as in Lemma 5.4 with desirable properties.

Lemma 5.5. *For any lattice $\mathcal{L} \subset \mathbb{R}^n$ with HKZ basis $(\mathbf{b}_1,\ldots,\mathbf{b}_n)$ with $n \ge 2$, any efficiently computable function $f : \mathbb{Z}^+ \to \mathbb{Z}^+$, and*

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

there exists $k \in [n]$ such that if $\mathcal{L}' := \mathcal{L}(\mathbf{b}_1,\ldots,\mathbf{b}_{k-1})$, then

$$\left|\{\mathbf{c} \in \mathcal{L}/\mathcal{L}' \ : \ \operatorname{dist}(\mathbf{t},\mathbf{c})^2 < \operatorname{dist}(\mathbf{t},\mathcal{L})^2 + r^2\}\right| \le \begin{cases} 2^{n-k+1} & : \text{ if } n - f(n) < k \le n \\ 2^{n-k+2}(4n)^{n/f(n)} & : \text{ otherwise} \end{cases}$$

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

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

$$\begin{aligned} r^2 + \frac{m_j - 1}{2} \cdot \sum_{i=1}^{m_j - 1} \|\widetilde{\mathbf{b}}_i\|^2 &< R^2 \cdot \Big(\frac{1}{(2n)^{4f(n)}} + \frac{(m_j-1)^2}{(2n)^{2j}} \Big) \\ &\le \frac{R^2}{(2n)^{2j}} \cdot \Big(\frac{1}{(2n)^{4f(n)-2j}} + (n-1)^2 \Big) \\ &< \frac{R^2}{(2n)^{2j-2}} \,. \end{aligned} \tag{6}$$

First, consider the case when there exists $j \in [f(n)]$ such that $m_j = m_{j-1}$. In this case, we claim that the required index is $k := m_j$. To see this, note that $\|\widetilde{\mathbf{b}}_k\| \ge \frac{R}{(2n)^{j-1}}$. Then, by Eq. (6), the conditions for Lemma 5.4 are satisfied with $\ell = k$, giving the desired result.

So, it is suffices to assume that $m_0 > m_1 > \cdots > m_{f(n)}$. In this case, clearly $m_{f(n)} \le n - f(n)$. Now, by the pigeonhole principle, there exists $j \in \{f(n)+1, f(n)+2, \ldots, 2f(n)\}$ such that $m_{j-1} - m_j < \frac{n}{f(n)}$. Then, let $k = m_j$, and $\ell = m_{j-1}$. Now, noting the fact that $\|\widetilde{\mathbf{b}}_k\| \ge \frac{R}{(2n)^j}$, and $\|\widetilde{\mathbf{b}}_\ell\| \ge \frac{R}{(2n)^{j-1}}$, the result follows from Lemma 5.4, and Eq. (6). □

We are now ready to present the reduction. We actually show how to find *all* closest vectors with high probability.

Theorem 5.6. *For any lattice $\mathcal{L} \subset \mathbb{R}^n$ and target $\mathbf{t} \in \mathbb{R}^n$, let $\sigma(\mathcal{L} - \mathbf{t}) = \lambda_n(\mathcal{L})/2^{Cn/\log^2 n}$, and 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, for $\varepsilon = 2^{-2n}$, a $2^{n+o(n)}$-time algorithm for ε-DGS$_\sigma^m$ implies a $2^{n+o(n)}$-time algorithm that finds all closest vectors to a target $\mathbf{t}$ with high probability.

Proof. We assume without loss of generality that $n \ge 20$ and that the target vector $\mathbf{t}$ is not in the lattice. Let $\bar{\mathbf{y}}$ be an arbitrary closest vector to $\mathbf{t}$. We first give an algorithm that outputs a set of closest vectors that contains $\bar{\mathbf{y}}$ with probability at least $1/2$.

Let D be an oracle solving ε-DGS$_\sigma^m$. We construct a recursive algorithm A^{D} with access to D that behaves as follows. On input $\mathcal{L} \subset \mathbb{R}^n$ and $\mathbf{t} \in \mathbb{R}^n$, A^{D} first calls the procedure from Corollary 2.14 on input $\mathcal{L}$ to compute an HKZ basis $(\mathbf{b}_1, \ldots, \mathbf{b}_n)$ of $\mathcal{L}$. Let $s := 2^{-Cn/\log^2 n} \max_q \|\widetilde{\mathbf{b}}_q\|$. For $i = 0, \ldots, \lceil Cn/\log^2 n \rceil$, A^{D} makes $2^{C\log^2 n}$ calls to $\mathrm{D}(\mathcal{L}, \mathbf{t}, s_i)$ where $s_i := 2^{-i/2} s$, receiving as output $\hat{m}_i \ge 2^{C\log^2 n} m(\mathcal{L} - \mathbf{t}, s_i)$ vectors $(\mathbf{X}_{i,1}, \ldots, \mathbf{X}_{i,\hat{m}_i}) \in \mathcal{L} - \mathbf{t}$.

Next, A^{D} finds k as in Lemma 5.5 with $f(n) := \lceil n/\log^4 n \rceil$. Let $\mathcal{L}' := \mathcal{L}(\mathbf{b}_1, \ldots, \mathbf{b}_{k-1})$. A^{D} groups the $\mathbf{X}_{i,j}$ according to the coset of $\mathbf{X}_{i,j} + \mathbf{t} \bmod \mathcal{L}'$. For each coset $\mathbf{c} \in \mathcal{L}/\mathcal{L}'$ that appears in the grouping, the algorithm chooses an arbitrary representative $\mathbf{X}_{i,j}$ corresponding to $\mathbf{c}$ and makes the recursive call $\mathrm{A}^{\mathrm{D}}(\mathcal{L}', -\mathbf{X}_{i,j})$, receiving as output a set of vectors $\mathcal{Y}_{\mathbf{c}} \subset \mathcal{L}'$. (Note that if $k = 1$ so that $\dim \mathcal{L}' = 0$, then $\mathcal{Y}_{\mathbf{c}} = \{\mathbf{0}\}$.) Finally, it returns all closest vectors amongst the $\mathbf{X}_{i,j} + \mathbf{y}_{\mathbf{c}} + \mathbf{t}$, where $\mathbf{X}_{i,j}$ is the representative of $\mathbf{c}$ and $\mathbf{y}_{\mathbf{c}} \in \mathcal{Y}_{\mathbf{c}}$.

First, we show that the algorithm terminates and has the desired running time. Note that since $\dim \mathcal{L}' = k - 1 < n$, the dimension drops at each level of recursion, and therefore the algorithm terminates. By Lemma 2.9, we have that $s_i > \lambda_n(\mathcal{L})/2^{Cn/\log^2 n} = \sigma(\mathcal{L} - \mathbf{t})$, so we may assume that the $\mathbf{X}_{i,j}$ are distributed exactly as independent discrete Gaussians, incurring statistical distance at most $2^{-2n + C\log^2 n} \le 2^{-n}$. By Lemma 5.3, we may also assume that for all i, j, $\|\mathbf{X}_{i,j}\|^2 < \mathrm{dist}(\mathbf{t}, \mathcal{L})^2 + 2(sn)^2 < \mathrm{dist}(\mathbf{t}, \mathcal{L})^2 + (2n)^{-4f(n)} \max_q \|\widetilde{\mathbf{b}}_q\|^2$, incurring additional statistical distance at most $2^{-n^2/C}$. (Here, we

use the fact that $\mathrm{dist}(\mathbf{t}, \mathcal{L})/(s_i\sqrt{n}) \leq 2^n$, which follows from Lemma 2.9.) Note that the number of recursive calls made by a single thread is given by

$$\begin{aligned} L &:= \left|\{\mathbf{c} \in \mathcal{L}/\mathcal{L}' \ : \ \exists\, i,j \text{ with } \mathbf{X}_{i,j} \in \mathbf{c}\}\right| \\ &\leq \left|\{\mathbf{c} \in \mathcal{L}/\mathcal{L}' \ : \ \mathrm{dist}(\mathbf{t},\mathbf{c})^2 < \mathrm{dist}(\mathbf{t},\mathcal{L})^2 + (2n)^{-4f(n)}\}\right| . \end{aligned}$$

It then follows from Lemma 5.5 that either $L \leq 2^{n-k+1}$ or $k \leq n - n/\log^4 n$ and $L \leq 2^{n-k+2}(4n)^{\log^4 n} \leq 2^{n-k+C\log^5 n}$. The other steps in a single recursive call run in time $2^{n+g(n)}$ for some efficiently computable increasing function $g : \mathbb{N} \mapsto \mathbb{N}$ such that $g(n) = o(n)$. So, our running time $T(n)$ satisfies the recurrence relation

$$T(n) \leq \begin{cases} 2^{n+g(n)} + 2^{n-k+C\log^5 n}T(k-1) & : \text{ if } 1 \leq k \leq n - n/\log^4 n \\ 2^{n+g(n)} + 2^{n-k+1}T(k-1) & : \text{ otherwise .} \end{cases} \tag{7}$$

We prove by induction that $T(n) \leq 2^{n+g(n)+C^*\sqrt{n}}$ for some fixed positive constant C^*. Assume the result holds for $m < n$. If $k > n - n/\log^4 n$, then

$$T(n) \leq 2^{n+g(n)} + 2^{n-k+1} \cdot 2^{k-1+g(k)+C^*\sqrt{k}} \leq 2^{n+g(n)+C^*\sqrt{n}} ,$$

as needed. For $k \leq n - n/\log^4 n$, we have that

$$\begin{aligned} T(n) &\leq 2^{n+g(n)} + 2^{n-k+C\log^5 n} \cdot 2^{k-1+g(k)+C^*\sqrt{k}} \\ &\leq 2^{n+g(n)} \cdot \left(1 + 2^{C^*\sqrt{n-n/\log^4 n}+C\log^5 n}\right) \\ &\leq 2^{n+g(n)} \cdot \left(1 + 2^{C^*\sqrt{n}-C^*\sqrt{n}/(2\log^4 n)+C\log^5 n}\right) \\ &\leq 2^{n+g(n)+C^*\sqrt{n}} . \end{aligned}$$

We now prove correctness. Suppose for induction that for any arbitrary closest vector, A^{D} finds it if the input lattice has dimension $k-1 < n$ with probability at least $\min(1, 1/2 + 2^{1-k})$. We show that with high probability there exists a vector $\mathbf{X}_{i,j}$ in the output of D with $\mathbf{X}_{i,j} + \mathbf{t} \equiv \bar{\mathbf{y}} \pmod{2\mathcal{L}}$. Again, we may assume that the $\mathbf{X}_{i,j}$ are distributed exactly as independent discrete Gaussians, incurring statistical distance at most $2^{-2n+C\log^2 n} \leq 2^{-n}$. Then, for all i, j,

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

By Corollary 3.6, we have that for some i, the above expression is at least, say, $n^2/\hat{m}_i$. It follows that there exists a $\mathbf{X}_{i,j}$ with $\mathbf{X}_{i,j} + \mathbf{t} \equiv \bar{\mathbf{y}} \pmod{2\mathcal{L}}$ with probability at least $1 - 2^{-n^2}$.

Fix i, j as above. Again by Lemma 5.3, we may assume that $\|\mathbf{X}_{i,j}\|^2 < \mathrm{dist}(\mathbf{t}, \mathcal{L})^2 + 2(sn)^2$, incurring additional statistical distance at most $2^{-n^2/C}$. Since we also have $\mathbf{X}_{i,j} + \mathbf{t} \equiv \bar{\mathbf{y}} \pmod{2\mathcal{L}}$, it follows from Lemma 5.1 that $\|\mathbf{X}_{i,j} + \mathbf{t} - \bar{\mathbf{y}}\|^2 < 4(sn)^2$. Therefore, $\bar{\mathbf{y}} \in \mathcal{L}' + \mathbf{X}_{i,j} + \mathbf{t}$, and we have that $\mathrm{CVP}(\mathcal{L}', -\mathbf{X}_{i,j}) + \mathbf{X}_{i,j} + \mathbf{t} = \mathrm{CVP}(\mathcal{L}, \mathbf{t})$. It then follows from the induction hypothesis and union bound that $\mathrm{CVP}(\mathcal{L}', -\mathbf{X}_{i,j})$ returns $\bar{\mathbf{y}} - \mathbf{X}_{i,j} - \mathbf{t}$, and hence our algorithm finds $\bar{\mathbf{y}}$ with probability at least $1/2 + 2^{-n}$, as needed.

Thus, repeating the algorithm $2n$ times, we obtain $\bar{\mathbf{y}}$ with probability at least $1 - 2^{-2n}$. Since the total number of closest vectors is at most 2^n, and the choice of $\bar{\mathbf{y}}$ in our analysis was arbitrary among all closest vectors, by the union bound we find *all* closest vectors with probability at least $1 - 2^{-n}$. □

Corollary 5.7. *There is an algorithm that takes input a lattice $\mathcal{L} \subset \mathbb{R}^n$ and target $\mathbf{t} \in \mathbb{R}^n$ and finds all the closest vectors in $\mathcal{L}$ to $\mathbf{t}$ (with high probability) in time $2^{n+o(n)}$. In particular, this algorithm solves* exact *CVP in time $2^{n+o(n)}$.*

Proof. Combine the algorithm from Theorem 4.7 with the reduction from Corollary 5.6. □

Acknowledgments

We thank Oded Regev and Alexander Golovnev for many helpful conversations.

References

[ABSS93] Sanjeev Arora, László Babai, Jacques 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*, pages 724–733. IEEE, 1993.

[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.

[AJ08] Vikraman Arvind and Pushkar S Joglekar. Some sieving algorithms for lattice problems. In *FSTTCS*, pages 25–36, 2008.

[Ajt96] Miklós Ajtai. Generating hard instances of lattice problems. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 99–108. ACM, 1996.

[Ajt98] Miklós 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*, pages 10–19. ACM, 1998.

[AKS01] Miklós Ajtai, Ravi Kumar, and D. Sivakumar. A sieve algorithm for the shortest lattice vector problem. In *STOC*, pages 601–610, 2001.

[AKS02] Miklós Ajtai, Ravi Kumar, and D. Sivakumar. Sampling short lattice vectors and the closest lattice vector problem. In *CCC*, pages 41–45, 2002.

[Ban93] W. Banaszczyk. New bounds in some transference theorems in the geometry of numbers. *Mathematische Annalen*, 296(4):625–635, 1993.

[BD15] Nicolas Bonifas and Daniel Dadush. Short paths on the Voronoi graph and the closest vector problem with preprocessing. In *SODA*, 2015.

[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.

[BHW93] U. Betke, M. Henk, and J.M. Wills. Successive-minima-type inequalities. *Discrete & Computational Geometry*, 9(1):165–175, 1993.

[BLP+13] Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. Classical hardness of learning with errors. In *STOC*, pages 575–584, 2013.

[BN09] Johannes Blömer and Stefanie Naewe. Sampling methods for shortest vectors, closest vectors and successive minima. *Theoret. Comput. Sci.*, 410(18):1648–1665, 2009.

[BS99] Johannes Blömer and Jean-Pierre 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*, pages 711–720. ACM, 1999.

[BV11] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In *FOCS*, pages 97–106. IEEE, 2011.

[BV14] Zvika Brakerski and Vinod Vaikuntanathan. Lattice-based FHE as secure as PKE. In *ITCS*, pages 1–12, 2014.

[CJL+92] Matthijs J Coster, Antoine Joux, Brian A LaMacchia, Andrew M Odlyzko, Claus-Peter Schnorr, and Jacques Stern. Improved low-density subset sum algorithms. *computational complexity*, 2(2):111–128, 1992.

[CN98] J-Y Cai and Ajay Nerurkar. Approximating the svp to within a factor (1-1/dim ϵ) is np-hard under randomized conditions. In *Computational Complexity, 1998. Proceedings. Thirteenth Annual IEEE Conference on*, pages 46–55. IEEE, 1998.

[DH11] Chandan Dubey and Thomas Holenstein. Approximating the closest vector problem using an approximate shortest vector oracle. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 184–193. Springer, 2011.

[DK13] Daniel Dadush and Gabor Kun. Lattice sparsification and the approximate closest vector problem. In *SODA*, 2013.

[DKRS03] Irit Dinur, Guy Kindler, Ran Raz, and Shmuel Safra. Approximating cvp to within almost-polynomial factors is np-hard. *Combinatorica*, 23(2):205–243, 2003.

[DPV11] Daniel Dadush, Chris Peikert, and Santosh Vempala. Enumerative lattice algorithms in any norm via m-ellipsoid coverings. In *Foundations of Computer Science (FOCS), 2011 IEEE 52nd Annual Symposium on*, pages 580–589. IEEE, 2011.

[DRS14] Daniel Dadush, Oded Regev, and Noah Stephens-Davidowitz. On the closest vector problem with a distance guarantee. In *IEEE 29th Conference on Computational Complexity*, pages 98–109, 2014. Full version available at `http://arxiv.org/abs/1409.8063`.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *STOC'09—Proceedings of the 2009 ACM International Symposium on Theory of Computing*, pages 169–178. ACM, New York, 2009.

[GMSS99] 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*, 71(2):55 – 61, 1999.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC*, pages 197–206, 2008.

[Hel85] Bettina Helfrich. Algorithms to construct Minkowski reduced and Hermite reduced lattice bases. *Theoret. Comput. Sci.*, 41(2-3):125–139 (1986), 1985.

[HPS11] Guillaume Hanrot, Xavier Pujol, and Damien Stehlé. Algorithms for the shortest and closest lattice vector problems. In *Coding and Cryptology*, pages 159–190. Springer, 2011.

[HR12] Ishay Haviv and Oded Regev. Tensor-based hardness of the shortest vector problem to within almost polynomial factors. *Theory of Computing*, 8(23):513–531, 2012. Preliminary version in STOC'07.

[HS07] Guillaume Hanrot and Damien Stehlé. Improved analysis of Kannan's shortest lattice vector algorithm (extended abstract). In *Advances in cryptology—CRYPTO 2007*, volume 4622 of *Lecture Notes in Comput. Sci.*, pages 170–186. Springer, Berlin, 2007.

[JS98] Antoine Joux and Jacques Stern. Lattice reduction: A toolbox for the cryptanalyst. *Journal of Cryptology*, 11(3):161–185, 1998.

[Kan87] Ravi Kannan. Minkowski's convex body theorem and integer programming. *Mathematics of Operations Research*, 12(3):pp. 415–440, 1987.

[Kho05] Subhash Khot. Hardness of approximating the shortest vector problem in lattices. *Journal of the ACM*, 52(5):789–808, September 2005. Preliminary version in FOCS'04.

[LJ83] Hendrik W Lenstra Jr. Integer programming with a fixed number of variables. *Mathematics of operations research*, 8(4):538–548, 1983.

[LJS90] J. C. Lagarias, Hendrik W. Lenstra Jr., and Claus-Peter Schnorr. Korkin-Zolotarev bases and successive minima of a lattice and its reciprocal lattice. *Combinatorica*, 10(4):333–348, 1990.

[LLL82] A. K. Lenstra, H. W. Lenstra, Jr., and L. Lovász. Factoring polynomials with rational coefficients. *Math. Ann.*, 261(4):515–534, 1982.

[LM83] Susan Landau and Gary Lee Miller. Solvability by radicals is in polynomial time. In *Proceedings of the fifteenth annual ACM symposium on Theory of computing*, pages 140–151. ACM, 1983.

[Mic01] Daniele Micciancio. The shortest vector problem is NP-hard to approximate to within some constant. *SIAM Journal on Computing*, 30(6):2008–2035, March 2001. Preliminary version in FOCS 1998.

[Mic08] Daniele Micciancio. Efficient reductions among lattice problems. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 84–93. ACM, New York, 2008.

[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.

[MV10] Daniele Micciancio and Panagiotis Voulgaris. Faster exponential time algorithms for the shortest vector problem. In *SODA*, pages 1468–1480, 2010.

[MV13] Daniele Micciancio and Panagiotis Voulgaris. A deterministic single exponential time algorithm for most lattice problems based on Voronoi cell computations. *SIAM Journal on Computing*, 42(3):1364–1391, 2013.

[MW15] Daniele Micciancio and Michael Walter. Fast lattice point enumeration with minimal overhead. In *SODA*, 2015.

[NS01] Phong Q Nguyen and Jacques Stern. The two faces of lattices in cryptology. In *Cryptography and lattices*, pages 146–180. Springer, 2001.

[NV08] Phong Q. Nguyen and Thomas Vidick. Sieve algorithms for the shortest vector problem are practical. *J. Math. Cryptol.*, 2(2):181–207, 2008.

[Odl90] Andrew M Odlyzko. The rise and fall of knapsack cryptosystems. *Cryptology and computational number theory*, 42:75–88, 1990.

[PS09] Xavier Pujol and Damien Stehlé. Solving the shortest lattice vector problem in time $2^{2.465n}$. *IACR Cryptology ePrint Archive*, 2009:605, 2009.

[Reg09] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM*, 56(6):Art. 34, 40, 2009.

[RS15] Oded Regev and Noah Stephens-Davidowitz. An inequality for Gaussians on lattices, 2015.

[SFS09] Naftali Sommer, Meir Feder, and Ofir Shalvi. Finding the closest lattice point by iterative slicing. *SIAM J. Discrete Math.*, 23(2):715–731, 2009.

A Proof of Lemma 4.1

Proof of Lemma 4.1. Multiplying the left-hand side of (2) by $\Pr_{(\mathbf{X}_1,\mathbf{X}_2)\sim D^2_{\mathcal{L}-\mathbf{t},s}}[\mathbf{X}_1+\mathbf{X}_2 \in 2\mathcal{L}-2\mathbf{t}]$, we get for any $\mathbf{y} \in \mathcal{L}-\mathbf{t}$,

$$\begin{aligned}\Pr_{(\mathbf{X}_1,\mathbf{X}_2)\sim D^2_{\mathcal{L}-\mathbf{t},s}}[(\mathbf{X}_1+\mathbf{X}_2)/2=\mathbf{y}] &= \frac{1}{\rho_s(\mathcal{L}-\mathbf{t})^2}\cdot\sum_{\mathbf{x}\in\mathcal{L}-\mathbf{t}}\rho_s(\mathbf{x})\rho_s(2\mathbf{y}-\mathbf{x})\\ &= \frac{\rho_{s/\sqrt{2}}(\mathbf{y})}{\rho_s(\mathcal{L}-\mathbf{t})^2}\cdot\sum_{\mathbf{x}\in\mathcal{L}-\mathbf{t}}\rho_{s/\sqrt{2}}(\mathbf{x}-\mathbf{y})\\ &= \frac{\rho_{s/\sqrt{2}}(\mathbf{y})}{\rho_s(\mathcal{L}-\mathbf{t})^2}\cdot\rho_{s/\sqrt{2}}(\mathcal{L})\ .\end{aligned}$$

Hence both sides of (2) are proportional to each other. Since they are probabilities, they are actually equal. □

B Proof of Proposition 4.2

Proof of Proposition 4.2. Let $(\mathbf{X}_1,\ldots,\mathbf{X}_M)$ be the input vectors. For each i, let $\mathbf{c}_i \in \mathcal{L}/(2\mathcal{L})$ be such that $\mathbf{X}_i \in \mathbf{c}_i-\mathbf{t}$. The combiner runs the algorithm from Theorem 2.15 with input κ and $(\mathbf{c}_1,\ldots,\mathbf{c}_M)$, receiving output $(\mathbf{c}'_1,\ldots,\mathbf{c}'_m)$. (Formally, we must encode the cosets as integers in $\{1,\ldots,2^n\}$.) Finally, for each $\mathbf{c}'_i$, it chooses a pair of unpaired vectors $\mathbf{X}_j,\mathbf{X}_k$ with $\mathbf{c}_j=\mathbf{c}_k=\mathbf{c}'_i$ and outputs $\mathbf{Y}_i=(\mathbf{X}_j+\mathbf{X}_k)/2$.

The running time of the algorithm follows from Item 1 of Theorem 2.15. Furthermore, we note that by Item 2 of the same theorem, there will always be a pair of indices j,k for each i as above.

To prove correctness, we observe that for $\mathbf{c}\in\mathcal{L}/(2\mathcal{L})$ and $\mathbf{y}\in\mathbf{c}-\mathbf{t}$,

$$\Pr[\mathbf{X}_i=\mathbf{y}] = \frac{\rho_s(\mathbf{c}-\mathbf{t})}{\rho_s(\mathcal{L}-\mathbf{t})}\cdot\Pr_{\mathbf{x}\sim D_{\mathbf{c}-\mathbf{t},s}}[\mathbf{X}=\mathbf{y}]\ .$$

In particular, we have that $\Pr[\mathbf{c}_i=\mathbf{c}]=\rho_s(\mathbf{c}-\mathbf{t})/\rho_s(\mathcal{L}-\mathbf{t})$. Then, the cosets $(\mathbf{c}_1,\ldots,\mathbf{c}_M)$ satisfy the conditions necessary for Item 3 of Theorem 2.15.

Applying the theorem, up to statistical distance $M\exp(C_1 n - C_2\kappa)$, we have that the output vectors are independent, and

$$\begin{aligned}m &\geq M\cdot\frac{1}{32\kappa}\cdot\frac{\sum_{\mathbf{c}\in\mathcal{L}/(2\mathcal{L})}\rho_s(\mathbf{c}-\mathbf{t})^2}{\rho_s(\mathcal{L}-\mathbf{t})\max_{\mathbf{c}\in\mathcal{L}/(2\mathcal{L})}\rho_s(\mathbf{c}-\mathbf{t})}\\ &= M\cdot\frac{1}{32\kappa}\cdot\frac{\rho_{s/\sqrt{2}}(\mathcal{L})\cdot\rho_{s/\sqrt{2}}(\mathcal{L}-\mathbf{t})}{\rho_s(\mathcal{L}-\mathbf{t})\max_{\mathbf{c}\in\mathcal{L}/(2\mathcal{L})}\rho_s(\mathbf{c}-\mathbf{t})}\ ,\end{aligned}$$

where the equality follows from Lemma 3.1 by setting $\mathbf{x}=\mathbf{t}$, and $\mathbf{y}=\mathbf{0}$. Furthermore, we have $\Pr[\mathbf{c}'_i=\mathbf{c}]=\rho_s(\mathbf{c}-\mathbf{t})^2/\sum_{\mathbf{c}'}\rho_s(\mathbf{c}'-\mathbf{t})^2$ for any coset $\mathbf{c}\in\mathcal{L}/(2\mathcal{L})$. Therefore, for any $\mathbf{y}\in\mathcal{L}$,

$$\begin{aligned}\Pr[\mathbf{Y}_i=\mathbf{y}] &= \frac{1}{\sum\rho_s(\mathbf{c}-\mathbf{t})^2}\cdot\sum_{\mathbf{c}\in\mathcal{L}/(2\mathcal{L})}\rho_s(\mathbf{c}-\mathbf{t})^2\cdot\Pr_{(\mathbf{X}_j,\mathbf{X}_k)\sim D^2_{\mathbf{c}-\mathbf{t},s}}[(\mathbf{X}_j+\mathbf{X}_k)/2=\mathbf{y}]\\ &= \Pr_{(\mathbf{X}_1,\mathbf{X}_2)\sim D^2_{\mathcal{L}-\mathbf{t},s}}[(\mathbf{X}_1+\mathbf{X}_2)/2=\mathbf{y}\mid\mathbf{X}_1+\mathbf{X}_2\in 2\mathcal{L}-2\mathbf{t}]\ .\end{aligned}$$

The result then follows from Lemma 4.1. □

C Proof of Proposition 4.5 and Corollary 4.6

Proof of Proposition 4.5. On input a lattice $\mathcal{L} \subset \mathbb{R}^n$, $\mathbf{t} \in \mathbb{R}^n$, and $r > 0$, the algorithm behaves as follows. First, it calls the procedure from Corollary 2.14 to compute an HKZ basis $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ of $\mathcal{L}$. Let $(\widetilde{\mathbf{b}}_1, \ldots, \widetilde{\mathbf{b}}_n)$ be the corresponding Gram-Schmidt vectors. Let $k \geq 0$ be maximal such that $\|\widetilde{\mathbf{b}}_i\| \leq r$ for $1 \leq i \leq k$, and let $\mathbf{B}' = (\mathbf{b}_1, \ldots, \mathbf{b}_k)$. Let $\pi_k = \pi_{\{\mathbf{b}_1,\ldots,\mathbf{b}_k\}^\perp}$ and $\mathcal{M} = \pi_k(\mathcal{L})$. The algorithm calls the procedure from Theorem 2.12 with input $\pi_k(\mathbf{t})$ and $\mathcal{M}$, receiving as output $\mathbf{x} = \sum_{i=k+1}^{n} a_i \pi_k(\mathbf{b}_i)$ where $a_i \in \mathbb{Z}$. Finally, the algorithm returns $\mathbf{y} = -\sum_{i=k+1}^{n} a_i \mathbf{b}_i$ and $\mathbf{B}' = (\mathbf{b}_1, \ldots, \mathbf{b}_k)$.

The running time is clear, as is the fact that $\|\widetilde{\mathbf{B}}'\| \leq r$. It remains to prove that $\mathcal{L}' - \mathbf{y} - \mathbf{t}$ contains all sufficiently short vectors in $\mathcal{L} - \mathbf{t}$. If $k = n$, then $\mathcal{L}' = \mathcal{L}$ and $\mathbf{y}$ is irrelevant, so we may assume that $k < n$. Note that, since $\mathbf{B}$ is an HKZ basis, $\lambda_1(\mathcal{M}) = \|\widetilde{\mathbf{b}}_{k+1}\| > r$. In particular, $\lambda_1(\mathcal{M}) > 3\,\mathrm{dist}(\mathbf{t}, \mathcal{L}) \geq 3\,\mathrm{dist}(\pi_k(\mathbf{t}), \mathcal{M})$, so there is a unique closest vector to $\pi_k(\mathbf{t})$. And, by triangle inequality, the next closest vector is at distance at least $2\,\mathrm{dist}(\pi_k(\mathbf{t}), \mathcal{M})$. Therefore, the call to the subprocedure from Theorem 2.12 will output an exact closest vector $\mathbf{x} \in \mathcal{M}$ to $\pi_k(\mathbf{t})$.

Let $\mathbf{w} \in \mathcal{L} \setminus (\mathcal{L}' - \mathbf{y})$ so that $\pi_k(\mathbf{w}) \neq \pi_k(\mathbf{y}) = \mathbf{x}$. We need to show that $\mathbf{w} - \mathbf{t}$ is relatively long. Since $\mathbf{B}$ is an HKZ basis, it follows that

$$\|\pi_k(\mathbf{w}) - \mathbf{x}\| \geq \lambda_1(\mathcal{M}) > r\,.$$

Applying triangle inequality, we have

$$\|\mathbf{w} - \mathbf{t}\| \geq \|\pi_k(\mathbf{w}) - \pi_k(\mathbf{t})\| \geq \|\pi_k(\mathbf{w}) - \mathbf{x}\| - \|\mathbf{x} - \pi_k(\mathbf{t})\| > r - \mathrm{dist}(\mathbf{t}, \mathcal{L})\,,$$

as needed. □

Proof of Corollary 4.6. The algorithm first calls the procedure from Proposition 4.5 with input $\mathcal{L}$, $\mathbf{t}$, and $r = C\hat{s}/\sqrt{\log n} \geq 3\,\mathrm{dist}(\mathbf{t}, \mathcal{L})$, receiving as output $\mathbf{y} \in \mathcal{L}$ and a basis $\mathbf{B}'$ of a sublattice $\mathcal{L}' \subset \mathcal{L}$. It then runs the algorithm from Theorem 4.4 M times with input $\mathcal{L}'$, $\mathbf{y} + \mathbf{t}$, and $\hat{s}$ and outputs the resulting vectors, $\mathbf{y}$, and $\mathcal{L}'$.

The running time is clear. By Proposition 4.5, $\mathcal{L}' - \mathbf{y} - \mathbf{t}$ contains all vectors of length at most $r - \mathrm{dist}(\mathbf{t}, \mathcal{L}) \geq C\hat{s}/\sqrt{\log n}$ in $\mathcal{L} - \mathbf{t}$, and $\|\widetilde{\mathbf{B}}'\| \leq r \leq C\hat{s}/\sqrt{\log n}$. So, it follows from Theorem 4.4 that the output has the correct distribution. □