SIAM J. COMPUT.
Vol. 54, No. 2, pp. 233–278




# IMPROVED CLASSICAL AND QUANTUM ALGORITHMS FOR THE SHORTEST VECTOR PROBLEM VIA BOUNDED DISTANCE DECODING*

DIVESH AGGARWAL†, YANLIN CHEN‡, RAJENDRA KUMAR§, AND YIXIN SHEN¶

**Abstract.** The most important computational problem on lattices is the shortest vector problem (SVP). In this paper, we present new algorithms that improve the state-of-the-art for provable classical/quantum algorithms for SVP. We present the following results: (1) A new algorithm for SVP that provides a smooth tradeoff between time complexity and memory requirement. For any positive integer $4 \leq q \leq \sqrt{n}$, our algorithm takes $q^{13n+o(n)}$ time and requires $poly(n) \cdot q^{16n/q^2}$ memory. This tradeoff, which ranges from enumeration ($q = \sqrt{n}$) to sieving ($q$ constant), is a consequence of a new time-memory tradeoff for discrete Gaussian sampling above the smoothing parameter. (2) A quantum algorithm for SVP that runs in time $2^{0.950n+o(n)}$ and requires $2^{0.5n+o(n)}$ classical memory and poly($n$) qubits. In a quantum random access memory (QRAM) model, this algorithm takes only $2^{0.835n+o(n)}$ time and requires a QRAM of size $2^{0.293n+o(n)}$, poly($n$) qubits and $2^{0.5n}$ classical space. This improves over the previously fastest classical (which is also the fastest quantum) algorithm due to [D. Aggarwal et al., *Solving the shortest vector problem in* $2^n$ *time using discrete Gaussian sampling: Extended abstract*, in Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing (STOC), 2015, pp. 733–742] that has a time and space complexity $2^{n+o(n)}$. (3) A classical algorithm for SVP that runs in time $2^{1.669n+o(n)}$ time and $2^{0.5n+o(n)}$ space. This improves over an algorithm of [Y. Chen, K. Chung, and C. Lai, *Quantum Inf. Comput.*, 18 (2018), pp. 285–306] that has the same space complexity. The time complexity of our classical and quantum algorithms are obtained using a known upper bound on a quantity related to the lattice kissing number, which is $2^{0.402n}$. We conjecture that for most lattices this quantity is a $2^{o(n)}$. Assuming that this is the case, our classical algorithm runs in time $2^{1.292n+o(n)}$, our quantum algorithm runs in time $2^{0.750n+o(n)}$, and our quantum algorithm in a QRAM model runs in time $2^{0.667n+o(n)}$. As a direct application of our result, using the reduction in [L. Ducas, *Des. Codes. Cryptogr.*, 92 (2024), pp. 909–916], we obtain a provable quantum algorithm for the lattice isomorphism problem in the case of the trivial lattice $\mathbb{Z}^n$ ($\mathbb{Z}$LIP) that runs in time $2^{0.417n+o(n)}$. Our algorithm requires a QRAM of size $2^{0.147n+o(n)}$, poly($n$) qubits and $2^{0.25n}$ classical space.

**Key words.** lattices, shortest vector problem, discrete Gaussian sampling, time-space tradeoff, quantum computation, bounded distance decoding

**MSC codes.** 68R01, 00A20, 00B10

**DOI.** 10.1137/22M1486959

*Received by the editors March 28, 2022; accepted for publication (in revised form) October 14, 2024; published electronically March 3, 2025. A preliminary version of this paper titled "Improved (Provable) Algorithms for the Shortest Vector Problem via Bounded Distance Decoding" appeared in *Proceedings of the 38th International Symposium on Theoretical Aspects of Computer Science* (STACS), Schloss Dagstuhl-Leibniz-Zentrum für Informatik, Wadern, Germany, 2021.

https://doi.org/10.1137/22M1486959

**Funding:** The work of the first author was supported by Singapore Ministry of Education grant MOE2019-T2-1-145 and National Research Foundation grant R-710-000-012-135. The work of the third author was supported by National Research Foundation Singapore award AISG-RP-2018-005. The work of the fourth author was supported by Engineering and Physical Sciences Research Council grant EP/W02778X/2 and by the French Agence Nationale de la Recherche through the France 2030 program under grant ANR-22-PETQ-0008 PQ-TLS.

†Centre for Quantum Technologies and National University of Singapore, Singapore 117543 Singapore (dcsdiva@nus.edu.sg).

‡QuSoft and CWI, 94079 Amsterdam, The Netherlands (yanlin@cwi.nl).

§Indian Institute of Technology Delhi, 110016 New Delhi, India (rjndr2503@gmail.com).

¶Univ Rennes, Inria, CNRS, IRISA, 350402 Rennes, France (yixin.shen@inria.fr).

**1. Introduction.** A lattice $\mathcal{L} = \mathcal{L}(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n) := \{\sum_{i=1}^{n} z_i \boldsymbol{b}_i : z_i \in \mathbb{Z}\}$ is the set of all integer combinations of linearly independent vectors $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n \in \mathbb{R}^n$. We call $n$ the rank of the lattice and $(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n)$ a basis of the lattice. The most important computational problem on lattices is the shortest vector problem (SVP). Given a basis for a lattice $\mathcal{L} \subseteq \mathbb{R}^n$, SVP asks us to compute a nonzero vector in $\mathcal{L}$ with the smallest Euclidean norm. Starting from the '80s, the use of approximate and exact solvers for SVP (and other lattice problems) gained prominence for their applications in algorithmic number theory [55], convex optimization [33, 46, 48], coding theory [28], and cryptanalysis tools [23, 54, 73]. The security of many cryptographic primitives is based on the worst-case hardness of (a decision variant of) approximate SVP to within polynomial factors [8, 21, 36, 60, 61, 69, 70] in the sense that any cryptanalytic attack on these cryptosystems that runs in time polynomial in the security parameter implies a polynomial time algorithm to solve approximate SVP to within polynomial factors. Such cryptosystems have attracted a lot of research interest due to their conjectured resistance to quantum attacks. The SVP is a well-studied computational problem in both its exact and approximate (decision) versions. By a randomized reduction, it is known to be NP-hard to approximate within any constant factor and hard to approximate within a factor $n^{c/\log\log n}$ for some $c > 0$ under reasonable complexity-theoretic assumptions [41, 49, 58]. For an approximation factor $2^{\mathcal{O}(n)}$, one can solve SVP in time polynomial in $n$ using the celebrated LLL lattice basis reduction algorithm [55]. In general, the fastest known algorithm(s) for approximating SVP within factors polynomial in $n$ rely on (a variant of) the BKZ lattice basis reduction algorithm [4, 9, 34, 39, 71, 72], which can be seen as a generalization of the LLL algorithm and gives an $r^{n/r}$ approximation in $2^{\mathcal{O}(r)}\mathrm{poly}(n)$ time. All these algorithms internally use an algorithm for solving (near) exact SVP in lower-dimensional lattices. Therefore, finding faster algorithms to solve exact SVP is critical to choosing security parameters of cryptographic primitives. As one would expect from the hardness results above, all known algorithms for solving exact SVP, including the ones we present here, require at least exponential time. In fact, the fastest known algorithms also require exponential space. There has been some recent evidence [5] showing that one cannot hope to get a $2^{o(n)}$ time algorithm for SVP if one believes in complexity theoretic conjectures such as the (gap) exponential time hypothesis. Most of the known algorithms for SVP can be broadly classified into two classes: (i) the algorithms that require memory polynomial in $n$ but run in time $n^{\mathcal{O}(n)}$ and (ii) the algorithms that require memory $2^{\mathcal{O}(n)}$ and run in time $2^{\mathcal{O}(n)}$. The first class, initiated by Kannan (see [10, 35, 40, 42, 48, 64]), combines basis reduction with exhaustive enumeration inside Euclidean balls. While enumerating vectors requires $2^{\mathcal{O}(n \log n)}$ time, it is much more space-efficient than other kinds of algorithms for exact SVP. Another class of algorithms, and currently the fastest, is based on sieving. First developed by Ajtai, Kumar, and Sivakumar [9], they generate many lattice vectors and then divide-and-sieve to create shorter and shorter vectors iteratively. A sequence of improvements [3, 6, 62, 65, 67, 68], has led to a $2^{n+o(n)}$ time and space algorithm by sieving the lattice vectors and carefully controlling the distribution of the output, thereby outputting a set of lattice vectors that contains the shortest vector with overwhelming probability. An alternative approach using the Voronoi cell of the lattice was proposed by Micciancio and Voulgaris [63] and gives a deterministic $2^{2n+o(n)}$-time and $2^{n+o(n)}$-space algorithm for SVP (and many other lattice problems). There are variants [15, 53, 62, 65] of the above-mentioned sieving algorithms that, under some heuristic assumptions, have an asymptotically smaller (but still $2^{\Theta(n)}$) time and space complexity than their provable counterparts.

*Algorithms giving a time/space tradeoff.* Even though sieving algorithms are asymptotically the fastest known algorithms for SVP, the memory requirement, in high dimension, has historically been a limiting factor to run these algorithms. Some recent works [11, 29] have shown how to use new tricks to make it possible to use sieving on high-dimensional lattices in practice and benefit from their efficient running time [74]. Nevertheless, it would be ideal and has been a longstanding open question of how to obtain an algorithm that achieves the "best of both worlds," i.e., an algorithm that runs in time $2^{\mathcal{O}(n)}$ and requires memory polynomial in $n$. In the absence of such an algorithm, it is desirable to have a smooth tradeoff between the time and memory requirement that interpolates between the current best sieving algorithms and the current best enumeration algorithms. To this end, Bai, Laarhoven, and Stehlé [13] proposed the tuple sieving algorithm, providing such a tradeoff based on heuristic assumptions similar in nature to prior sieving algorithms. This algorithm was later proven to have time and space complexity $k^{O(n)}$ and $k^{O(n/k)}$ under the same heuristic assumptions [43]. One can vary the parameter $k$ to obtain a smooth time/space tradeoff. Nevertheless, it is still desirable to obtain a provable variant of this algorithm that does not rely on any heuristics. Kirchner and Fouque [50] attempted to do this. They claim an algorithm for solving SVP in time $k^{\Theta(n)}$ and in space $k^{\Theta(n/k)}$ for any positive integer $k > 1$. Unfortunately, their analysis falls short of supporting their claimed result, and the correctness of the algorithm is not clear. We refer the reader to section 1.3 for more details. In addition to the above, Chen, Chung, and Lai [24] propose a variant of the algorithm based on discrete Gaussian sampling in [3]. Their algorithm runs in time $2^{2.05n+o(n)}$, and the memory requirement is $2^{0.5n+o(n)}$. The quantum variant of their algorithm runs in time $2^{1.2553n+o(n)}$ and has the same space complexity. Their algorithm has the best space complexity among known provably correct algorithms that run in time $2^{O(n)}$. A number of works have also investigated the potential quantum speedups for lattice algorithms, and SVP in particular. A landscape similar to the classical one exists, although the quantum memory model has its importance. While quantum enumeration algorithms only require qubits [12], sieving algorithms require more powerful QRAMs [51, 53].

**1.1. Our results.** We first present a new algorithm for SVP that provides a smooth tradeoff between the time complexity and memory requirement of SVP without any heuristic assumptions. This algorithm is obtained by giving a new algorithm for sampling lattice vectors from the discrete Gaussian distribution that runs in time $k^{\mathcal{O}(n)}$ and requires $k^{\mathcal{O}(n/k)}$ space.

THEOREM 1.1 (time-space tradeoff for smooth discrete Gaussian, informal). *There is an algorithm that takes as input a lattice $\mathcal{L} \subset \mathbb{R}^n$, a positive integer $q$, and a parameter $s$ above the smoothing parameter of $\mathcal{L}$ and outputs $q^{16n/q^2}$ samples from $D_{\mathcal{L},s}$ using $q^{13n+o(n)}$ time and $poly(q) \cdot q^{16n/q^2}$ space.*

Using the standard reduction from bounded distance decoding (BDD) with preprocessing (where an algorithm solving the problem is allowed unlimited preprocessing time on the lattice before the algorithm receives the target vector) to discrete Gaussian sampling (DGS) from [27] and a reduction from SVP to BDD given in [24], we obtain the following.

THEOREM 1.2 (time-space tradeoff for SVP). *Let $n \in \mathbb{N}, q \in [4, \sqrt{n}]$ be a positive integer. Let $\mathcal{L}$ be a lattice of rank $n$. There is a randomized algorithm that solves SVP in time $q^{13n+o(n)}$ and in space $poly(n) \cdot q^{\frac{16n}{q^2}}$.*

If we take $k = q^2$, then the time complexity of the previous SVP algorithm becomes $k^{6.5n+o(n)}$ and the space complexity $\mathrm{poly}(n) \cdot k^{(8n/k)}$. Our tradeoff is thus the same (up to a constant in the exponents) as what was claimed by Kirchner and Fouque [50] and proven in [43] *under heuristic assumptions.* Our second result is a quantum algorithm for SVP that improves over the current *fastest quantum algorithm* for SVP [3] (Notice that the algorithm in [3] is still the fastest classical algorithm for SVP.)

THEOREM 1.3 (quantum algorithm for SVP). *There is a quantum algorithm that solves SVP in $2^{0.950n+o(n)}$ time and classical $2^{0.5n+o(n)}$ space with an additional poly$(N)$ qubits. In the quantum random access memory (QRAM) model, there is an algorithm that solves SVP in $2^{0.8345n+o(n)}$ time and requires a QRAM of size $2^{0.293n+o(n)}$, poly$(n)$ qubits, and $2^{0.5n+o(n)}$ classical space.*

Our third result is a classical algorithm for SVP that improves over the algorithm from [24] and results in the fastest classical algorithm that has a space complexity $2^{0.5n+o(n)}$.

THEOREM 1.4 (algorithm for SVP with $2^{0.5n+o(n)}$ space). *There is a classical algorithm that solves SVP in $2^{1.669n+o(n)}$ time and $2^{0.5n+o(n)}$ space.*

The time complexity of our second and third results are obtained using a known upper bound on a quantity $\beta(\mathcal{L})^n$ related to the kissing number, which depends on the lattice and is always upper bounded by $2^{0.402n}$. We analyzed the dependency of the running time of our algorithm in this quantity $\beta(\mathcal{L})$ and plotted (see Figure 3) the graph of the complexity exponent as a function of $\beta(\mathcal{L})$. In practice, for most lattices, $\beta(\mathcal{L})^n$ is often a $2^{o(n)}$.[1] In this case, the running time of our algorithm is significantly better than when using the generic upper bound on $\beta(\mathcal{L})$.

THEOREM 1.5. *For any family $(\mathcal{L}_n)_n$ of full-rank lattices such that $\mathcal{L}_n \subseteq \mathbb{R}^n$ and $\beta(\mathcal{L}_n)^n = 2^{o(n)}$, there are algorithms to solve the SVP on $\mathcal{L}_n$:*

- *in classical time $2^{1.292n+o(n)}$ and space $2^{0.5n}$,*
- *in quantum time $2^{0.750n+o(n)}$, classical space $2^{0.5n}$, and poly$(n)$ qubits,*
- *in quantum time $2^{0.667n+o(n)}$, classical space $2^{0.5n}$, and poly$(n)$ qubits and using a QRAM of size $2^{0.167n+o(n)}$.*

Below in Tables 1 and 2, we summarize known provable classical and quantum algorithms, respectively. Note that all the classical algorithms are also quantum algorithms but they don't use any quantum power.

TABLE 1
*Classical algorithms for the shortest vector problem.*

| Time complexity | Space complexity | Reference |
|---|---|---|
| $n^{\frac{n}{2e}+o(n)}$ | poly$(n)$ | [40, 48] |
| $2^{\mathcal{O}(n)}$ | $2^{\mathcal{O}(n)}$ | [9] |
| $2^{2.465n+o(n)}$ | $2^{1.233n+o(n)}$ | [67] |
| $2^{2n+o(n)}$ | $2^{n+o(n)}$ | [62] |
| $2^{n+o(n)}$ | $2^{n+o(n)}$ | [3] |
| $2^{2.05n+o(n)}$ | $2^{0.5n+o(n)}$ | [24] |
| $2^{1.669n+o(n)}$ | $2^{0.5n+o(n)}$ | This paper |

[1] Please refer to our discussion on kissing numbers in section 2.

TABLE 2
*Quantum algorithms for the shortest vector problem.*

| Time complexity | Space complexity | Reference |
|---|---|---|
| $2^{1.799n+o(n)}$ | $2^{1.286n+o(n)}$ QRAM | [53] |
| $2^{1.2553n+o(n)}$ | $2^{0.5n+o(n)}$ | [24] |
| $2^{0.950n+o(n)}$ | $2^{0.5n+o(n)}$ | This paper |
| $2^{0.835n+o(n)}$ | $2^{0.5n+o(n)}$ classical space and $2^{0.293n+o(n)}$ QRAM | This paper |

**1.1.1. Roadmap.** We give a high-level overview of our proofs in section 1.2. We compare our results with the previous known algorithms that claim/conjecture a time-space tradeoff for SVP in section 1.3. Section 2 contain some preliminaries on lattices. The proofs of the time-space tradeoff for discrete Gaussian sampling above the smoothing parameter and the time-space tradeoff for SVP are given in section 3. In section 4, we present a quantum algorithm for BDD with preprocessing that we will use later in our quantum algorithm for SVP. In section 5, we present improved algorithms for BDD. Our quantum and classical algorithms for solving SVP with space complexity $2^{0.5n+o(n)}$ are presented in section 6, respectively. Section 7 shows how the time complexity of our algorithms varies on a quantity related to the kissing number. Section 8 gives a direct application of our results to the lattice isomorphism problem in the case of the trivial lattice $\mathbb{Z}^n$ ($\mathbb{Z}$LIP).

*Remark* 1.6. The paper is an extended version of [2], published in Proceedings of the 38th International Symposium on Theoretical Aspects of Computer Science (STACS), Wadern, Germany, 2021. Compared to [2], this paper gives a quantum speedup for Bounded distance decoding in processing using QRAM. Using this quantum speedup, we get a much faster quantum algorithm for SVP. We also realized that one of the conditions in Theorem 2.19 is unnecessary,[2] and this leads to a better time complexity in all our algorithms compared to the previous version. Furthermore, we study the dependency of our algorithms on a quantity related to the kissing number in section 7.

**1.2. Proof overview.** We now include a high-level description of our proofs. Before describing our proof ideas, we emphasize that it was shown in [3, 27] that given an algorithm for DGS a constant factor $c$ above the smoothing parameter, we can solve the problem of BDD where the target vector is within distance $\alpha\lambda_1(\mathcal{L})$ of the lattice, where the constant $\alpha < 0.5$ depends on the constant $c$. Additionally, using [24], one can enumerate all lattice points within distance $p\delta$ to a target $\boldsymbol{t}$ by querying $p^n$ times a BDD oracle with decoding distance $\delta$ (or $p^{n/2}$ times if we are given a quantum BDD oracle). Thus, by choosing $p = \lceil \lambda_1(\mathcal{L})/\delta \rceil$ and $\boldsymbol{t} = \mathbf{0}$, an algorithm for BDD immediately gives us an algorithm for SVP. Therefore, it suffices to give an algorithm for DGS above the smoothing parameter.

**1.2.1. Time-space tradeoff for DGS above smoothing.** Recall that efficient algorithms are known for sampling from a discrete Gaussian with a large enough parameter (width) [20, 37, 52]. In [3], the authors begin by sampling $N = 2^{n+o(n)}$ vectors from the discrete Gaussian distribution with (large) parameter $s$ and then look for pairs of vectors whose sum is in $2\mathcal{L}$ or, equivalently, pairs of vectors that lie in the same coset $\boldsymbol{c} \in \mathcal{L}/2\mathcal{L}$. Since there are $2^n$ cosets, if we take $\Omega(2^n)$ samples from $D_{\mathcal{L},s}$, almost all of the resulting vectors (except at most $2^n$ vectors) will be paired and

[2]See Remark 2.20.

are statistically close to independent samples from the distribution $D_{\mathcal{L},s/\sqrt{2}}$, provided that the parameter $s$ is sufficiently above the smoothing parameter. To reduce the space complexity, we modify the idea of the algorithm by generating random samples and checking if the summation of $d$ of those samples is in $q\mathcal{L}$ for some integer $q$. Intuitively, if we start with two lists of vectors ($L_1$ and $L_2$) of size $q^{\mathcal{O}(n/d)}$ from $D_{\mathcal{L},s}$, where $s$ is sufficiently above the smoothing parameter, each of these vectors is contained in any coset $q\mathcal{L}+\boldsymbol{c}$ for any $\boldsymbol{c}\in\mathcal{L}/q\mathcal{L}$ with probability roughly $1/q^n$. We therefore expect that the coset of a uniformly random d-combination of vectors from $L_2$ is uniformly distributed in $\mathcal{L}/q\mathcal{L}$. The proof of this statement follows from the leftover hash lemma [45]. We therefore expect that for any vector $\boldsymbol{v}\in L_1$, with high probability, there is a set of $d$ vectors $\boldsymbol{x}_1,\ldots,\boldsymbol{x}_d$ in $L_2$ that sum to a vector in $q\mathcal{L}+\boldsymbol{v}$, and hence $\frac{1}{q}(\sum_{i=1}^{d}\boldsymbol{x}_i-\boldsymbol{v})\in\mathcal{L}$. A lemma by Micciancio and Peikert [59] shows that this vector is statistically close to a sample from the distribution, $D_{\mathcal{L},s\sqrt{d+1}/q}$. We can find such a combination by trying all subsets of $d$ vectors. We would like to repeat this and find $q^{\mathcal{O}(n/d)}$ (nearly) independent vectors in $q\mathcal{L}$. It is not immediately clear how to continue since, in order to guarantee independence, one would not want to reuse the already used vectors $\boldsymbol{x}_1,\ldots,\boldsymbol{x}_d$, and conditioned on the choice of these vectors, the distribution of the cosets containing the remaining vectors is disturbed and is no longer nearly uniform. By using a simple combinatorial argument, we show that even after removing any $1/\mathrm{poly}(d)$ fraction of vectors from the list $L_2$, the $d$-combination of vectors in $L_2$ has at least $cq^n$ different cosets. This is sufficient to output $q^{\mathcal{O}(n/d)}$ independent vectors in $q\mathcal{L}$ with overwhelming probability.

**1.2.2. Quantum algorithm for BDDP in QRAM.** In [27], Dadush, Regev, and Stephens-Davidowitz gave an algorithm for BDD with preprocessing (or BDDP), which requires advice containing vectors sampled from discrete Gaussian sampling over the dual lattice. The idea is to use the *periodic Gaussian function* $f$ (defined in section 4) to go near the closest lattice vector. The function $f$ is periodic over the lattice, and its value depends only on the distance between the input vector and the lattice. Do the gradient ascent by iteratively updating the target vector using values of $\nabla f$ and $f$ such that the distance of the target vector from the closest lattice vector decreases. In this work, we show that we can reduce the time complexity of this algorithm by using the quantum amplitude estimation technique (given in [22]) with the assumption that the advice string is stored in QRAM. More specifically, we show that just by using $\mathcal{O}(\sqrt{N})$ arithmetic operations in QRAM, we can solve BDDP where $N$ is the size of the advice string required in [27].

**1.2.3. An improved algorithm for DGS at the smoothing parameter.** The BDD to DGS reduction from [27] requires samples from $D_{\mathcal{L},\eta_\varepsilon(\mathcal{L})}$ for $\varepsilon=2^{-cn}$ for some constant $c$. In [3], the authors gave an algorithm that runs in time $2^{n/2+o(n)}$ and outputs $2^{n/2+o(n)}$ samples from $D_{\mathcal{L},s}$ for any $s\geq\sqrt{2}\eta_{0.5}(\mathcal{L})$, i.e., a factor $\sqrt{2}$ above the smoothing parameter. In order to obtain samples at the smoothing parameter, we construct a dense lattice $\mathcal{L}'$ of smaller smoothing parameter than $\mathcal{L}$. We then sample $2^{0.5n+o(n)}$ vectors from $D_{\mathcal{L}',s}$ and reject those that are not in $\mathcal{L}$. This allows us to obtain an improved algorithm for BDD with preprocessing.

**1.2.4. Covering surface of a ball by spherical caps.** This result improves the quantum algorithm from [24]. As we mentioned above, one can enumerate all lattice points within a $p\delta$ distance to a target $\boldsymbol{t}$ by querying $p^n$ times a BDD oracle with decoding distance $\delta$. Our algorithm for BDD is obtained by preparing samples from the discrete Gaussian distribution. However, note that the decoding distance of

a BDD oracle built by discrete Gaussian samples as shown in [27] is successful if the target vector is within a radius $\alpha\lambda_1(\mathcal{L})$ for $\alpha < 1/2$ (there is a tradeoff between $\alpha$ and the number of DGS samples needed), and therefore, if we choose $\boldsymbol{t}$ to be $\mathbf{0}$, as we do in the other algorithms mentioned above, then $p$ has to be at least 3 to ensure that the shortest vector is one of the vectors output by the enumeration algorithm mentioned above. We observe here that if we choose a target $\boldsymbol{t}$ to be a random vector on a sphere of a well-chosen radius centered at the origin, then the shortest vector will be within a radius $2\delta$ from the target $\boldsymbol{t}$ with some probability $P$, and thus we can find the shortest vector by making $2^n/P$ calls to the BDD oracle. An appropriate[3] choice of the target $\boldsymbol{t}$ and the factor $\alpha$ gives an algorithm that runs in time $2^n \cdot 2^{0.669n+o(n)}$. We then explain how to obtain a quantum speedup of this algorithm (with its corresponding optimized value of $\alpha$, which is different from the classical one) that runs in time $2^{n/2} \cdot 2^{0.45n+o(n)}$ with a polynomial number of qubits. Finally, if we can store the DGS samples in a QRAM, we can obtain a further speedup and an algorithm that runs in time $2^{n/2} \cdot 2^{0.3345n+o(n)}$ and uses a QRAM of size $2^{0.293n+o(n)}$.

**1.2.5. Dependency on a quantity related to the kissing number.** The running time of the above algorithms crucially depends on a quantity related to the kissing number of the input lattice. This quantity plays a role in the BDD to DGS reduction when relating the decoding radius $\alpha$ to the $\varepsilon$ when sampling at the smoothing parameter $\eta_\varepsilon$. Our algorithms takes significantly less time for the smaller values of this quantity. However, the only known upper bound on this quantity seems to be very pessimistic for most lattices. Since we have used this upper bound to derive the complexity of our algorithm (except in section 7), this means that the actual running time of this algorithm might be much better for most lattices. For a more elaborate discussion on this, see section 7. As far as we know, this is the first time that the time/space complexity dependency on a quantity related to the input lattice has been investigated. For example, the time complexity of the SVP algorithm in [62, 67] depends on the geometric kissing number, which is a universal quantity depending only on the dimension $n$, for which we know a lower bound of $2^{0.2075n}$.

**1.3. Comparison with previous algorithms giving a time/space tradeoff.** Kirchner and Fouque [50] begin their algorithm by sampling an exponential number of vectors from the discrete Gaussian distribution $D_{\mathcal{L},s}$ and then using a pigeon-hole principle show that there are two distinct sums of $d$ vectors (for an appropriate $d$) that are equal mod $q\mathcal{L}$ for some large enough integer $q$. This results in a $\{-1, 0, 1\}$ combination of input lattice vectors (of Hamming weight at most $2d$) in $q\mathcal{L}$; a similar idea was used in [13] to construct their tuple sieving algorithm. In both algorithms, it is difficult to control (i) the distribution of the resulting vectors and (ii) the dependence between resulting vectors. Bai, Laarhoven, and Stehlé [13] get around the above issues by making a heuristic assumption that the resulting vectors behave like independent samples from a "nice enough" distribution. The authors of [43] proved that this heuristic indeed leads to the time-memory tradeoff conjectured in [13], but they don't prove correctness. Kirchner and Fouque, on the other hand, use the pigeon-hole principle to argue that there exist coefficients $\alpha_1, \ldots, \alpha_{2d} \in \{-1, 0, 1\}$ and $2d$ lattice vectors in the set of input vectors $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{2d}$ such that $\frac{\sum_{i=1}^{2d} \alpha_i \boldsymbol{v}_i}{q} \in \mathcal{L}$. It is then stated that $\frac{\sum_{i=1}^{2d} \alpha_i \boldsymbol{v}_i}{q}$ has a nice enough discrete Gaussian distribution. We observe that while the resulting distribution obtained will indeed be close to a discrete

[3]The optimal choice of $\alpha$ is obtained by numerically optimization; see section 7.

Gaussian distribution, we have no control over the parameter $s$ of this distribution and it can be anywhere between $1/q$ and $\sqrt{2d}/q$ depending on the number of nonzero coordinates in $(\alpha_1, \ldots, \alpha_{2d})$. For instance, Suppose $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_5$ are input vectors which are all from $D_{\mathcal{L},s}$ for some large $s$ and we want to find the collision in $q\mathcal{L}$ for some positive integer $q$. Suppose that we find a combination $\boldsymbol{w}_1 = \boldsymbol{v}_1 + \boldsymbol{v}_2 - (\boldsymbol{v}_1 + \boldsymbol{v}_3) \in q\mathcal{L}$ and another combination $\boldsymbol{w}_2 = \boldsymbol{v}_2 + \boldsymbol{v}_3 - (\boldsymbol{v}_4 + \boldsymbol{v}_5) \in q\mathcal{L}$; then, by Theorem 2.11, one would expect that $\boldsymbol{w}_1/q \sim D_{\mathcal{L},\sqrt{2}s/q}$ and $\boldsymbol{w}_2/q \sim D_{\mathcal{L},\sqrt{4}s/q}$. This means that the output of the exhaustive search algorithm by Kirchner and Fouque will behave like samples taken from discrete Gaussian distributions with different parameters, making it difficult to keep track of the standard deviation after several steps of the algorithm and to obtain samples from the discrete Gaussian distribution at the desired parameter above the smoothing parameter. We overcome this issue by showing that there is a combination of the input vectors with a *fixed* Hamming weight that is in $q\mathcal{L}$, as mentioned in section 1.2. There are other technical details that were overlooked in [50]. In particular, one needs be careful with respect to the errors, both in the probability of failure and the statistical distance of the input/output. Indeed, the algorithm performs an exponential number of steps, and it is not enough to show that the algorithm succeeds with "overwhelming probability" and that the output has a "negligible statistical distance" from the desired output. However, many of the claimed error bounds in [50] are not proven, making it difficult to verify the proof of the exhaustive search algorithm (Theorem 3.4) and of Theorem 3.6.

**2. Preliminaries.** Let $\mathbb{N} = \{1, 2, \ldots, \}$. We use bold letters $\boldsymbol{x}$ for vectors and denote a vector's coordinates with indices $x_i$. We use log to represent the logarithm base 2 and ln to represent the natural logarithm. Throughout the paper, $n$ will always be the dimension of the ambient space $\mathbb{R}^n$. We will denote the principal branch of Lambert's W function by $W(x)$; see [26] for an introduction to this function.

**2.0.1. Lattices.** A *lattice* $\mathcal{L}$ is a discrete subgroup of $\mathbb{R}^m$ or, equivalently, the set

$$\mathcal{L}(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n) = \left\{ \sum_{i=1}^{n} x_i \boldsymbol{b}_i \ : \ x_i \in \mathbb{Z} \right\}$$

of all integer combinations of $n$ linearly independent vectors $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n \in \mathbb{R}^m$. Such $\boldsymbol{b}_i$'s form a *basis* of $\mathcal{L}$. The lattice $\mathcal{L}$ is said to be *full-rank* if $n = m$. We denote by $\lambda_1(\mathcal{L})$ the first minimum of $\mathcal{L}$, defined as the length of a shortest nonzero vector of $\mathcal{L}$. For a rank $n$ lattice $\mathcal{L} \subset \mathbb{R}^n$, the *dual lattice,* denoted by $\mathcal{L}^*$, is defined as the set of all points in $\mathsf{span}(\mathcal{L})$ that have integer inner products with all lattice points,

$$\mathcal{L}^* = \{\boldsymbol{w} \in \mathsf{span}(\mathcal{L}) : \forall \boldsymbol{y} \in \mathcal{L}, \langle \boldsymbol{w}, \boldsymbol{y} \rangle \in \mathbb{Z}\}.$$

Similarly, for a lattice basis $\mathbf{B} = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n)$, we define the dual basis $\mathbf{B}^* = (\boldsymbol{b}_1^*, \ldots, \boldsymbol{b}_n^*)$ to be the unique set of vectors in $\mathsf{span}(\mathcal{L})$ satisfying $\langle \boldsymbol{b}_i^*, \boldsymbol{b}_j \rangle = 1$ if $i = j$, and 0 otherwise. It is easy to show that $\mathcal{L}^*$ is itself a rank $n$ lattice and $\mathbf{B}^*$ is a basis of $\mathcal{L}^*$. Given a lattice $\mathbf{B} = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n)$, we denote $\|\mathbf{B}\|_2 = \max_i \|\boldsymbol{b}_i\|$.

**2.0.2. Kissing number and related quantities.** For any lattice $\mathcal{L} \subset \mathbb{R}^n$ and $d > 0$, let $N(\mathcal{L}, r)$ denote the number of nonzero lattice vectors of length at most $r$. A natural question is how to bound this quantity in terms of $r$. When $r < \lambda_1(\mathcal{L})$, only the origin lies inside the ball, so $N(\mathcal{L}, r) = 0$. When $r = \lambda_1(\mathcal{L})$, this quantity is known as the kissing number $\tau(\mathcal{L})$ of the lattice:

$$\tau(\mathcal{L}) = |\{\boldsymbol{x} \in \mathcal{L} : \|\boldsymbol{x}\| = \lambda_1(\mathcal{L})\}|.$$

Finally, when $r \to \infty$, $N(\mathcal{L}, r) = \frac{r^n \operatorname{vol}(B_n(1))}{\det \mathcal{L}} + o(r^n)$ by the geometric interpretation of the determinant of a lattice. The precise behavior for intermediate values of $r$, however, is unclear, and for that reason we introduce the quantity

$$\gamma(\mathcal{L}) = \inf\{\gamma : \forall r \geq 1,\ N(\mathcal{L}, r\lambda_1(\mathcal{L})) \leq \gamma \cdot r^n\}. \tag{1}$$

It is clear by the definition that $\gamma(\mathcal{L}) \geq \tau(\mathcal{L})$. The best known upper bound on this quantity comes from the breakthrough work of Kabatjanskiĭ and Levenšhteĭn [47], which implies [67] that

$$\gamma(\mathcal{L}) \leqslant 2^{0.402n}. \tag{2}$$

*Remark* 2.1. To the best of our knowledge, we do not know any family of lattices with kissing number $2^{\Omega(n)}$.[4] Given this, it seems reasonable to conjecture that for any lattice that one would come across in practice, the kissing number is $2^{o(n)}$. Also, given the close connection between $\tau(\mathcal{L})$ and $\gamma(\mathcal{L})$, we conjecture that $\gamma(\mathcal{L})$ is also $2^{o(n)}$ for almost all lattices. We leave it as an open problem whether there exists a family of lattices for which $\gamma(\mathcal{L})$ is exponential in the dimension of the lattice. In view of the fact that $\gamma(\mathcal{L})$ can be anywhere between 1 and $2^{0.402n}$, we will study the dependence of the time complexity of our algorithms on $\gamma(\mathcal{L})$ by introducing

$$\beta(\mathcal{L}) = \gamma(\mathcal{L})^{1/n}. \tag{3}$$

The upper bound above can then be reformulated as $\beta(\mathcal{L}) \leqslant 2^{0.402}$ for any lattice $\mathcal{L}$.

**2.0.3. Probability distributions.** Given two random variables $X$ and $Y$ on a set $E$, we denote by $\mathrm{d_{SD}}$ the *statistical distance* between $X$ and $Y$, which is defined by

$$\begin{aligned}\mathrm{d_{SD}}(X, Y) &= \frac{1}{2} \sum_{z \in E} \left| \Pr_X[X = z] - \Pr_Y[Y = z] \right| \\ &= \sum_{z \in E\,:\, \Pr_X[X=z] > \Pr_Y[Y=z]} \left( \Pr_X[X = z] - \Pr_Y[Y = z] \right).\end{aligned}$$

We write that $X$ is $\varepsilon$-close to $Y$ to denote that the statistical distance between $X$ and $Y$ is at most $\varepsilon$. Given a finite set $E$, we denote by $U_E$ a uniform random variable on $E$, i.e., for all $x \in E$, $\Pr_{U_E}[U_E = x] = \frac{1}{|E|}$.

**2.0.4. Discrete Gaussian distribution.** For any $s > 0$, define $\rho_s(\boldsymbol{x}) = \exp(-\pi \|\boldsymbol{x}\|^2 / s^2)$ for all $\boldsymbol{x} \in \mathbb{R}^n$. We write $\rho$ for $\rho_1$. For a discrete set $S$, we extend $\rho$ to sets by $\rho_s(S) = \sum_{\boldsymbol{x} \in S} \rho_s(\boldsymbol{x})$. Given a lattice $\mathcal{L}$, the *discrete Gaussian* $D_{\mathcal{L},s}$ is the distribution over $\mathcal{L}$ such that the probability of a vector $\boldsymbol{y} \in \mathcal{L}$ is proportional to $\rho_s(\boldsymbol{y})$:

$$\Pr_{X \sim D_{\mathcal{L},s}}[X = \boldsymbol{y}] = \frac{\rho_s(\boldsymbol{y})}{\rho_s(\mathcal{L})}.$$

[4] Serge Vlăduţ [75] gave a construction for a set of lattices and claimed their kissing number is $2^{0.0338n + o(n)}$, while Bennett, Golovnev, and Stephens-Davidowitz [18] recently showed Vlăduţ's construction is invalid. Therefore, showing the existence of a family of lattices with exponential kissing numbers remains an open problem.

**2.0.5. Data processing inequality.** When analyzing the output distribution of an algorithm, it is often convenient to assume that the input distribution is ideal (e.g., uniform). On the other hand, we will want to run the algorithm on nonideal input distribution (e.g., with a slight deviation from uniform). In this case, the output distribution will deviate from the ideal output distribution and it is important to quantify this divergence. The statistical distance satisfies the following useful inequality, known as the *data processing inequality*:

$$\mathrm{d}_{\mathrm{SD}}(f(X), f(Y)) \leqslant \mathrm{d}_{\mathrm{SD}}(X, Y)$$

for any two distributions $X$ and $Y$ and any (possibly randomized) algorithm $f$. In other words, the error does not increase under the application of $f$.

**2.1. Lattice problems.** The following problems play a central role in this paper. For convenience, when we discuss the running time of algorithms solving the problems below, we ignore polynomial factors in the bit-length of the individual input basis vectors (i.e., we assume that the input basis has bit-size polynomial in the ambient dimension $n$).

DEFINITION 2.2. *For $\delta = \delta(n) \geq 0$, $\sigma$ a function that maps lattices to nonnegative real numbers, and $m = m(n) \in \mathbb{N}$, $\delta$-$\mathsf{DGS}_\sigma^m$ (the discrete Gaussian sampling problem) is defined as follows: The input is a basis $\mathbf{B}$ for a lattice $\mathcal{L} \subset \mathbb{R}^n$ and a parameter $s > \sigma(\mathcal{L})$. The goal is to output a sequence of $m$ vectors whose joint distribution is $\delta$-close to $m$ independent samples from $D_{\mathcal{L},s}$.*

We omit the parameter $\delta$ if $\delta = 0$ and the parameter $m$ if $m = 1$. We stress that $\delta$ bounds the statistical distance between the *joint* distribution of the output vectors and $m$ independent samples from $D_{\mathcal{L},s}$. We consider the following lattice problems.

DEFINITION 2.3. *The search problem $\mathsf{SVP}$ (shortest vector problem) is defined as follows: The input is a basis $\mathbf{B}$ for a lattice $\mathcal{L} \subset \mathbb{R}^n$. The goal is to output a vector $\boldsymbol{y} \in \mathcal{L}$ with $\|\boldsymbol{y}\| = \lambda_1(\mathcal{L})$.*

DEFINITION 2.4. *The search problem $\mathsf{CVP}$ (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 $\boldsymbol{t} \in \mathbb{R}^n$. The goal is to output a vector $\boldsymbol{y} \in \mathcal{L}$ with $\|\boldsymbol{y} - \boldsymbol{t}\| = \mathsf{dist}(\boldsymbol{t}, \mathcal{L})$.*

DEFINITION 2.5. *For $\alpha = \alpha(n) < 1/2$, the search problem $\alpha$-$\mathsf{BDD}$ (bounded distance decoding) is defined as follows: The input is a basis $\mathbf{B}$ for a lattice $\mathcal{L} \subset \mathbb{R}^n$ and a target vector $\boldsymbol{t} \in \mathbb{R}^n$ with $\mathsf{dist}(\boldsymbol{t}, \mathcal{L}) \leq \alpha \cdot \lambda_1(\mathcal{L})$. The goal is to output a vector $\boldsymbol{y} \in \mathcal{L}$ with $\|\boldsymbol{y} - \boldsymbol{t}\| = \mathsf{dist}(\boldsymbol{t}, \mathcal{L})$.*

Note that while our other problems become more difficult as the approximation factor $\gamma$ becomes smaller, $\alpha$-$\mathsf{BDD}$ becomes more difficult as $\alpha$ gets larger.

DEFINITION 2.6. *The search problems $\gamma$-$\mathsf{CVPP}$ and $\gamma$-$\mathsf{BDDP}$ are the preprocessing analogues of $\gamma$-$\mathsf{CVP}$ and $\gamma$-$\mathsf{BDD}$, respectively. The input for preprocessing is a basis $\mathbf{B}$ of lattice $\mathcal{L} \subset \mathbb{R}^n$. Given the advice from the preprocessing algorithm and the target vector $\boldsymbol{t} \in \mathcal{R}^n$, the goal is to return solution of $\gamma$-$\mathsf{CVP}$ and $\gamma$-$\mathsf{BDD}$, respectively. The preprocessing algorithm is allowed to take arbitrary time.*

For a lattice $\mathcal{L}$ and $\varepsilon > 0$, the *smoothing parameter* $\eta_\varepsilon(\mathcal{L})$ is the smallest $s$ such that $\rho_{1/s}(\mathcal{L}^*) = 1 + \varepsilon$. Recall that if $\mathcal{L}$ is a lattice and $\boldsymbol{v} \in \mathcal{L}$, then $\rho_s(\mathcal{L} + \boldsymbol{v}) = \rho_s(\mathcal{L})$ for all $s$. The *smoothing parameter* has the following well-known property.

LEMMA 2.7 (see [70, Claim 3.8]). *For any lattice $\mathcal{L} \subset \mathbb{R}^n$, $\boldsymbol{c} \in \mathbb{R}^n$, $\varepsilon > 0$, and $s \geq \eta_\varepsilon(\mathcal{L})$,*

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

COROLLARY 2.8. *Let $\mathcal{L} \subset \mathbb{R}^n$ be a lattice, let $q$ be a positive integer, and let $s \geq \eta_\varepsilon(q\mathcal{L})$. Let $C$ be a random coset in $\mathcal{L}/q\mathcal{L}$ sampled such that for any $\vec{c} \in \mathcal{L}/q\mathcal{L}$, $\Pr[C = q\mathcal{L} + \boldsymbol{c}] = \frac{\rho_s(q\mathcal{L}+\boldsymbol{c})}{\rho_s(\mathcal{L})}$. Also, let $U$ be a coset in $\mathcal{L}/q\mathcal{L}$ sampled uniformly at random. Then*

$$\mathrm{d}_{SD}(C,U) \leq 2\varepsilon.$$

*Proof.* By Lemma 2.7, we have that

$$\rho_s(q\mathcal{L}) \geq \rho_s(q\mathcal{L}+\boldsymbol{c}) \geq \frac{1-\varepsilon}{1+\varepsilon}\rho_s(q\mathcal{L})$$

for any $\boldsymbol{c} \in \mathcal{L}/q\mathcal{L}$, and hence

$$q^n \rho_s(q\mathcal{L}) \geq \sum_{\boldsymbol{c}\in\mathcal{L}/q\mathcal{L}} \rho_s(q\mathcal{L}+\boldsymbol{c}) = \rho_s(\mathcal{L})$$

Therefore,

$$\frac{\rho_s(q\mathcal{L}+\boldsymbol{c})}{\rho_s(\mathcal{L})} \geq \frac{1-\varepsilon}{1+\varepsilon} \cdot \frac{\rho_s(q\mathcal{L})}{\rho_s(\mathcal{L})} \geq \frac{1-\varepsilon}{1+\varepsilon} \cdot \frac{1}{q^n}.$$

We conclude that

$$\begin{aligned}
\mathrm{d}_{\mathrm{SD}}(C,U) &= \sum_{\boldsymbol{c}\in\mathcal{L}/q\mathcal{L}\,:\,\Pr[\boldsymbol{C}=\boldsymbol{c}]<\Pr[U=\boldsymbol{c}]} (\Pr[U=\boldsymbol{c}] - \Pr[C=\boldsymbol{c}]) \\
&\leq \sum_{\boldsymbol{c}\in\mathcal{L}/q\mathcal{L}\,:\,\Pr[C=\boldsymbol{c}]<\Pr[U=\boldsymbol{c}]} \Pr[U=\boldsymbol{c}]\left(1 - \frac{1-\varepsilon}{1+\varepsilon}\right) \\
&\leq \sum_{\boldsymbol{c}\in\mathcal{L}/q\mathcal{L}} \Pr[U=\boldsymbol{c}]\frac{2\varepsilon}{1+\varepsilon} \\
&\leq \frac{2\varepsilon}{1+\varepsilon}
\end{aligned}$$

as needed. ∎

We will need the following lemma, which was initially proved in [14].

LEMMA 2.9 (see [27, Lemma 2.13]). *For any lattice $\mathcal{L} \subset \mathbb{R}^n$, $s > 0$ and any $t \geq 1$,*

$$\Pr_{\boldsymbol{y}\sim\mathcal{D}_{\mathcal{L},s}}\left[\|\boldsymbol{y}\| \geq t\sqrt{\frac{n}{2\pi}} \cdot s\right] \leq e^{-\frac{n}{2}(t-1)^2}.$$

The following lemma gives a bound on the smoothing parameter.

LEMMA 2.10 (see [3, Lemma 2.7]). *For any lattice $\mathcal{L} \subset \mathbb{R}^n, \varepsilon \in (0,1)$, and $k > 1$, we have $k\eta_\varepsilon(\mathcal{L}) > \eta_{\varepsilon^{k^2}}(\mathcal{L})$.*

Micciancio and Peikert [59] showed the following result about the resulting distribution from the sum of many Gaussian samples.

THEOREM 2.11 (see [59, Theorem 3.3]). *Let $\mathcal{L}$ be an $n$ dimensional lattice, let $\boldsymbol{z} \in \mathbb{Z}^m$ be a nonzero integer vector, let $s_i \geq \sqrt{2}\|\boldsymbol{z}\|_\infty \cdot \eta_\varepsilon(\mathcal{L})$, and let $\mathcal{L} + \boldsymbol{c_i}$ be arbitrary cosets of $\mathcal{L}$ for $i = 1, \ldots, m$. Let $\boldsymbol{y_i}$ be independent vectors with distributions $D_{\mathcal{L}+\boldsymbol{c_i}, s_i}$, respectively. Then the distribution of $\boldsymbol{y} = \sum_i z_i \boldsymbol{y_i}$ is $\varepsilon m$ close to $D_{Y,s}$, where $Y = gcd(\boldsymbol{z})\mathcal{L} + \sum_i z_i \boldsymbol{c_i}$, and $s = \sqrt{\sum (z_i s_i)^2}$.*

We need to recall the definition of the *honest* discrete Gaussian sampling problem, introduced in [3].

DEFINITION 2.12 (see [3, Definition 5.1]). *For $\varepsilon \geqslant 0$, $\sigma$ a function that maps lattices to nonnegative real numbers, and $m \in \mathbb{N}$, the* honest *discrete Gaussian sampling problem $\varepsilon$-hDGS$_\sigma^m$ is defined as follows: the input is a basis $B$ for a lattice $\mathcal{L} \subset \mathbb{R}^n$ and a parameter $s > 0$. The goal is for the output distribution to be $\varepsilon$-close to $D_{\mathcal{L},s}^{m'}$ for some independent random variable $m' \geqslant 0$. If $s > \sigma(\mathcal{L})$, then $m'$ must be equal to $m$.*

THEOREM 2.13 (see [3, Theorem 5.11]). *Let $\sigma$ be the function that maps a lattice $\mathcal{L}$ to $\sqrt{2}\eta_{1/2}(\mathcal{L})$. Then there is an algorithm that solves $\exp(-\Omega(\kappa))$-hDGS$_\sigma^{2^{n/2}}$ in time $2^{n/2+\mathrm{polylog}(\kappa)+o(n)}$ for an $\kappa \geqslant \Omega(n)$.*

We will need the following reduction from $\alpha$-BDD to DGS that was shown in [27].

THEOREM 2.14 (see [27, Theorem 3.1]). *For any $\varepsilon \in (0, 1/200)$, lattice $\mathcal{L} \subset \mathbb{R}^n$, and $\alpha \leq \frac{\sqrt{\ln(1/\varepsilon)/\pi - o(1)}}{2\eta_\varepsilon(\mathcal{L}^*)\lambda_1\mathcal{L}}$. There exists an algorithm that solves $\alpha$-BDDP using $m \cdot poly(n)$ arithmetic operations where $m = O(\frac{n\log(1/\varepsilon)}{\sqrt{\varepsilon}})$. Moreover, the preprocessing advice only consists of $m$ vectors sampled from $D_{\mathcal{L}^*, \eta_\varepsilon(\mathcal{L}^*)}$.*

*Remark* 2.15. We are going to use this reduction in the superpolynomial regime: typically, $m$ will be exponential in $n$. This leaves unclear the space complexity of the reduction. The reduction works by evaluating a polynomial number of times functions of the form $\sum_{i=1}^m f_i(\boldsymbol{x})$ where each $f_i$ is a polynomial time computable function that depends on the $i$th DGS sample. Furthermore, all the complexities above are in terms of arithmetic operations, not bit complexity. If we assume that all the DGS samples have poly$(n)$ bit-size, then the reduction has time complexity $m \cdot \mathrm{poly}(n)$ and space complexity $O(\mathrm{poly}(n) + \log m)$, *excluding the storage space of the $m$ vectors provided by DGS.* Finally, as noted in the proof of the theorem in [27], only the preprocessing is probabilistic, and with probability at least $1 - 2^{-\Omega(n)}$ over the choice of the samples, the algorithm will deterministically solve all $\alpha$-BDD instances.

We need the following relation between the first minimum of lattice and the smoothing parameter of dual lattice. We will use this to compute the decoding distance of the BDD oracle.

LEMMA 2.16 (variant of [3, Lemma 6.1]). *For any lattice $\mathcal{L} \subset \mathbb{R}^n$ and $\varepsilon \in (0, 1)$,*

$$\sqrt{\frac{\ln(1/\varepsilon)}{\pi}} < \lambda_1(\mathcal{L})\eta_\varepsilon(\mathcal{L}^*) < \sqrt{\frac{\beta(\mathcal{L})^2 n}{2\pi e}} \cdot \varepsilon^{-1/n} \cdot (1 + o(1)), \tag{4}$$

*and if $\varepsilon \leq (e/\beta(\mathcal{L})^2 + o(1))^{-\frac{n}{2}}$, we also have*

$$\sqrt{\frac{\ln(1/\varepsilon)}{\pi}} < \lambda_1(\mathcal{L})\eta_\varepsilon(\mathcal{L}^*) < \sqrt{\frac{\ln(1/\varepsilon) + n\ln\beta(\mathcal{L}) + o(n)}{\pi}} \tag{5}$$

*as $n$ tends to infinity.*

*Remark* 2.17. As noted in [3] below Lemma 6.1, the inequality (4) actually holds for all $\varepsilon \in (0, 1)$, so we dropped the condition in the first case.

*Remark* 2.18. In Lemma 6.1 of [3], $\beta$ comes from Lemma 4.2 of the same paper and only needs to satisfy the equation $|\mathcal{L} \cap T_r| \leqslant \beta^{n+o(n)} r^n$ for all $r$, where $T_r = \{x \in \mathbb{R}^n : r \leqslant \|x\| \leqslant (1+\frac{1}{n})r\}$, assuming the lattice is normalized so that $\lambda_1(\mathcal{L}) = 1$. A trivial upper bound on $|\mathcal{L} \cap T_r|$ is $N(\mathcal{L}, r')$, the number of points in $\mathcal{L}$ of radius at most $r' = (1+\frac{1}{n})r$. By our definitions (1) and (3), this is bounded by $\gamma(\mathcal{L})\big((1+\frac{1}{n})r\big)^n \leqslant \beta(\mathcal{L})^n e r^n$ and therefore we can replace $\beta$ by our $\beta(\mathcal{L})$.

The following theorem, proved in [24], is required to solve SVP by an exponential number of calls to $\alpha$-BDD oracle.

THEOREM 2.19 (see [24, Theorem 8]). *Given a basis matrix $\mathbf{B} \in \mathbb{R}^{n \times n}$ for lattice $\mathcal{L}(\mathbf{B}) \subset \mathbb{R}^n$, a target vector $\boldsymbol{t} \in \mathbb{R}^n$, an $\alpha$-BDD oracle $\mathsf{BDD}_\alpha$ with $\alpha < 0.5$, and an integer scalar $p > 0$, let $f_p^\alpha : \mathbb{Z}_p^n \to \mathbb{R}^n$ be $f_p^\alpha(\boldsymbol{s}) = -p \cdot \mathsf{BDD}_\alpha(\mathcal{L}, (\mathbf{B}\boldsymbol{s} - \boldsymbol{t})/p) + \mathbf{B}\boldsymbol{s}$; then the list $m = \{f_p^\alpha(\boldsymbol{s}) \mid \boldsymbol{s} \in \mathbb{Z}_p^n\}$ contains all lattice points within distance $p\alpha\lambda_1(\mathcal{L})$ to $\boldsymbol{t}$.*

*Remark* 2.20. In the original Theorem 8 from [24], there is an extra condition on the target vector $\boldsymbol{t}$; i.e., $\boldsymbol{t}$ needs to satisfy $\mathsf{dist}(\mathcal{L}, \boldsymbol{t}) \leq \alpha\lambda_1(\mathcal{L})$. However, a thorough inspection of the proof reveals that this condition is unnecessary, and essentially the same proof follows without it.

We will need the following theorems to sample the DGS vectors with a large width.

THEOREM 2.21 (see [3, Proposition 2.17]). *For any $\varepsilon \leq 0.99$, there is an algorithm that takes as input a lattice $\mathcal{L} \in \mathbb{R}^n$, $M \in \mathbb{Z}_{>0}$ (the desired number of output vectors), and $s > 2^{n \log\log n / \log n} \cdot \eta_\varepsilon(\mathcal{L})$ and outputs $M$ independent samples from $D_{\mathcal{L},s}$ in time $M \cdot poly(n)$.*

LEMMA 2.22 (see [3, Lemma 5.12]). *There is a probabilistic polynomial-time algorithm that takes as input a lattice $\mathcal{L} \subset \mathbb{R}^n$ of rank $n$ and an integer $a$ with $n/2 \leq a < n$ and returns a superlattice $\mathcal{L}' \supset \mathcal{L}$ of index $2^a$ with $\mathcal{L}' \subseteq \mathcal{L}/2$ such that for any $\varepsilon \in (0,1)$, we have $\eta_{\varepsilon'}(\mathcal{L}') \leq \eta_\varepsilon(\mathcal{L})/\sqrt{2}$ with probability at least $1/2$, where $\varepsilon' := 2\varepsilon^2 + 2^{(n/2)+1-a}(1+\varepsilon)$.*

**2.2. Quantum computation.** In this paper, we use the Dirac bra-ket notation. A qubit is a unit vector in $\mathbb{C}^2$ with two (ordered) basis vectors $\{|0\rangle, |1\rangle\}$. The following gates form a universal set of gates:

$$H = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix},\ S = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix},\ T = e^{i\pi/8}\begin{bmatrix} e^{-i\pi/8} & 0 \\ 0 & e^{i\pi/8} \end{bmatrix},$$
$$CNOT = |0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes X.$$

The Toffoli gate, a three-qubit gate, is defined by

$$\text{Toffoli}|a\rangle|b\rangle|c\rangle = \begin{cases} |a\rangle|b\rangle|1 \oplus c\rangle & \text{if } a = b = 1, \\ |a\rangle|b\rangle|c\rangle & \text{otherwise} \end{cases}$$

for $a, b, c \in \{0, 1\}$. The Toffoli gate can be efficiently decomposed into $CNOT, H, S$, and $T$ gates [66], and hence it is considered as an elementary quantum gate in this paper. It is easy to see that a NAND gate can be implemented by a Toffoli gate: $\text{Toffoli}\,|a\rangle\,|b\rangle\,|1\rangle = |a\rangle\,|b\rangle\,|\text{NAND}(a,b)\rangle$, where $\text{NAND}(a,b) = 0$ if $(a,b) = (1,1)$ and $\text{NAND}(a,b) = 1$ otherwise. In particular, the Toffoli gate together with ancilla preparation is universal for classical computation; that is, for any classical function, we can implement it as (controlled) quantum one although at a nonnegligible cost due to the reversibility of quantum circuits.

THEOREM 2.23 (see [16, 56]). *Given any $\varepsilon > 0$ and any classical computation with running time $T$ and space complexity $S$, there exists an equivalent reversible classical computation with running time $O(T^{1+\varepsilon}/S^{\varepsilon})$ and space complexity $O(S(1+\ln(T/S)))$.*

COROLLARY 2.24. *Given any $\varepsilon > 0$ and any classical computation with running time $T$ and space complexity $S$, there exists an equivalent quantum circuit of size $O(T^{1+\varepsilon}/S^{\varepsilon})$ using $O(S(1+\ln(T/S)))$ qubits.*

**2.2.1. Search problem.** One of the most well known quantum algorithms is Grover's unstructured search algorithm [38]. Suppose we have a set of objects named $\{0, 1, \ldots, N-1\}$, of which some are *targets*. We say that an oracle $\mathcal{O}$ *identifies the targets* if, in the classical (resp., quantum) setting, $\mathcal{O}(i) = 1$ (resp., $\mathcal{O}|i\rangle = -|i\rangle$) when $i$ is a target and $\mathcal{O}(i) = 0$ (resp., $\mathcal{O}|i\rangle = |i\rangle$) otherwise. Given such an oracle $\mathcal{O}$, the goal is to find a target $j \in \{0, 1, \ldots, N-1\}$ by making queries to the oracle $\mathcal{O}$. In the search problem, one will try to minimize the number of queries to the oracle. In the classical setting, one needs $\mathcal{O}(N)$ queries to solve this problem. Grover, on the other hand, provided a quantum algorithm that solves the search problem with only $\mathcal{O}(\sqrt{N})$ queries []. When the number of targets is unknown, Brassard et al. provided a modified Grover algorithm that solves the search problem with $\mathcal{O}(\sqrt{N})$ queries [19], which is of the same order as the query complexity of the Grover search. Moreover, Dürr and Høyer showed, given an unsorted table of $N$ values, that there exists a quantum algorithm that finds the index of minimum with only $\mathcal{O}(\sqrt{N})$ queries [32], with constant probability of error.

THEOREM 2.25 (see [32, Theorem 1). *Let $T[1, 2, ..., N]$ be an unsorted table of $N$ items, each holding a value from an ordered set. Suppose that we have a quantum oracle $\mathcal{O}_T$ such that $\mathcal{O}_T|i\rangle|0\rangle = |i\rangle|T[i]\rangle$. Then there exists a quantum algorithm that finds the index $y$ such that $T[y]$ is the minimum with probability at least $1/2$ and $\mathcal{O}(\sqrt{N})$ queries to $\mathcal{O}_T$.*

**2.3. Probability.** We need the following lemma on the distribution of vector inner product which directly follows from the leftover hash lemma [45].

LEMMA 2.26. *Let $\mathbb{G}$ be a finite abelian group, and let $f$ be a positive integer. Let $\mathcal{Y} \subseteq \{0,1\}^f$. Define the inner product $\langle \cdot, \cdot \rangle : \mathbb{G}^f \times \mathcal{Y} \to \mathbb{G}$ by $\langle x, y \rangle = \sum_i x_i y_i$ for all $x \in \mathbb{G}^f, y \in \mathcal{Y}$. Let $X, Y$ be independent and uniformly random variables on $\mathbb{G}^f, \mathcal{Y}$, respectively. Then*

$$\mathrm{d}_{SD}((\langle X, Y \rangle, X), (U_{\mathbb{G}}, X)) \leq \frac{1}{2} \cdot \sqrt{\frac{|\mathbb{G}|}{|\mathcal{Y}|}},$$

*where $U_{\mathbb{G}}$ is uniform in $\mathbb{G}$ and independent of $X$.*

We will also need the Chernoff–Hoeffding bound [44].

LEMMA 2.27. *Let $X_1, \ldots, X_M$ be the independent and identically distributed random boolean variables of expectation $p$. Then, for $\varepsilon > 0$,*

$$\Pr\left[\frac{1}{M}\sum_{i=1}^{M} X_i \leq p(1-\delta)\right] \leq \left(\frac{e^{-\delta}}{(1-\delta)^{1-\delta}}\right)^{pM}.$$

**3. Algorithms with a time-memory tradeoff for lattice problems.** In this section, we present a new algorithm for discrete Gaussian sampling above the smoothing parameter.

**3.1. Algorithm for discrete Gaussian sampling.** We now present the main result of this section.

THEOREM 3.1. *Let $n \in \mathbb{N}, q \geq 2, d \in [1, n]$ be positive integers, and let $\varepsilon > 0$. Let $C$ be any positive integer. Let $\mathcal{L}$ be a lattice of rank $n$, and let $s \geq 2\sqrt{d}q\eta_\varepsilon(\mathcal{L})$. There is an algorithm that, given $N = 160d^2 \cdot C \cdot q^{n/d}$ independent samples from $D_{\mathcal{L},s}$, outputs a list of vectors that is $(4\varepsilon^{2d}N + 11Cq^{-5n/2})$-close to $Cq^{n/d}$ independent vectors from $D_{\mathcal{L},\frac{\sqrt{8d+1}}{q}s}$. The algorithm runs in time $C \cdot (10e \cdot d)^{8d} \cdot q^{8n+n/d+o(n)}$ and requires memory $poly(d) \cdot q^{n/d}$.*

*Proof.* We prove the result for $C = 1$, and the general result follows by repeating the algorithm. Let $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ be the $N$ input vectors, and let $\{\boldsymbol{c}_1, \ldots, \boldsymbol{c}_N\}$ be the corresponding cosets in $\mathcal{L}/q\mathcal{L}$. We will analyze Algorithm 1; note that it produces samples in a streaming fashion. The time complexity of the algorithm is

$$\frac{N}{2} \cdot \binom{N/2}{8d} \leq \frac{N}{2} \left(\frac{eN}{16d}\right)^{8d} \leq (10e \cdot d)^{8d} \cdot q^{8n+n/d+o(n)},$$

and memory requirement of the algorithm is immediate. We now show correctness: we will make repeated use of the data processing inequality (see section 2.0.5) and accumulate the error terms until the end of the proof. Let $\varepsilon' = \varepsilon^{2d}$ so that $s \geq \sqrt{2}\eta_{\varepsilon'}(q\mathcal{L})$ by Lemma 2.10. First, we can assume that the vectors $\boldsymbol{x}_i$ for $i \in [N]$ are obtained by first sampling $\boldsymbol{c}_i \in \mathcal{L}/q\mathcal{L}$ such that $\Pr[\boldsymbol{c}_i = \boldsymbol{c}] = \Pr[D_{\mathcal{L},s} \in q\mathcal{L} + \boldsymbol{c}]$ and then sampling the vector $\boldsymbol{x}_i$ according to $D_{q\mathcal{L}+\boldsymbol{c}_i,s}$. Indeed, let $X \sim D_{\mathcal{L},s}$, and let $C$ be a random coset in $\mathcal{L}/q\mathcal{L}$ sampled such that $\Pr[C = q\mathcal{L} + \boldsymbol{c}] = \frac{\rho_s(q\mathcal{L}+\boldsymbol{c})}{\rho_s(\mathcal{L})}$; then, for any $\boldsymbol{x} \in \mathcal{L}$,

$$\Pr[X = \boldsymbol{x}] = \frac{\rho_s(\boldsymbol{x})}{\rho_s(\mathcal{L})} = \frac{\rho_s(q\mathcal{L}+\boldsymbol{c})}{\rho_s(\mathcal{L})} \frac{\rho_s(\boldsymbol{x})}{\rho_s(q\mathcal{L}+\boldsymbol{c})} = \Pr[C = \boldsymbol{c}] \Pr_{Y \sim D_{q\mathcal{L}+\boldsymbol{c},s}}[Y = \boldsymbol{x}],$$

where $\boldsymbol{c} = q\mathcal{L} + \boldsymbol{x}$. Moreover, by Corollary 2.8, this distribution is $2\varepsilon' N$-close to sampling $\boldsymbol{c}_i$ for $i \in [N]$, independently and uniformly from $\mathcal{L}/q\mathcal{L}$, and then sampling

**Algorithm 1.** TradeOffSieve($L$).

**Require:** $L = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ is a list of $N$ vectors in $\mathcal{L}$
1: $L_1 \leftarrow \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{\frac{N}{2}}\}$
2: $L_2 \leftarrow \{\boldsymbol{x}_{\frac{N}{2}+1}, \ldots, \boldsymbol{x}_N\}$
3: $Q \leftarrow 0$
4: **while** $Q < q^{n/d}$ and $L_1$ is not empty **do**
5: $\quad \boldsymbol{v} \leftarrow$ is the first vector in $L_1$.
6: $\quad$ Find $8d$ vectors (by trying all $8d$-tuples) $\boldsymbol{x}_{i_1}, \ldots, \boldsymbol{x}_{i_{8d}}$ from $L_2$ s.t. $\boldsymbol{c}_{i_1} + \cdots + \boldsymbol{c}_{i_{8d}} - \boldsymbol{v} \in q\mathcal{L}$.
7: $\quad$ **if** such vectors exist **then**
8: $\quad\quad$ Output the vector $\frac{\boldsymbol{x}_{i_1}+\cdots+\boldsymbol{x}_{i_{8d}}-\boldsymbol{v}}{q} \in \mathcal{L}$
9: $\quad\quad Q \leftarrow Q + 1$.
10: $\quad\quad$ Remove vectors $\boldsymbol{x}_{i_1}, \ldots, \boldsymbol{x}_{i_{8d}}$ from $L_2$
11: $\quad$ **end if**
12: $\quad$ Remove vector $\vec{v}$ from $L_1$
13: **end while**

the vectors $\boldsymbol{x}_i$ according to $D_{q\mathcal{L}+\boldsymbol{c}_i,s}$. We now assume that the input is sampled from this distribution. Second, we can assume that the algorithm initially gets only the corresponding cosets as input, and the vectors $\boldsymbol{x}_{i_j} \in q\mathcal{L}+\boldsymbol{c}_{i_j}$ for $j \in [8d]$ and $\boldsymbol{v} \in q\mathcal{L}+\boldsymbol{c}$ are sampled from $D_{q\mathcal{L}+\boldsymbol{c}_{i_j},s}$ and $D_{q\mathcal{L}+\boldsymbol{c},s}$ only before such a tuple is needed in step 8 of the algorithm. Indeed, notice that the test at line 6 does not actually depend on the particular values of the $\boldsymbol{x}_{i_j}$ and $\boldsymbol{v}$ but only on their cosets. Since any input vector is used only once in step 8, these samples are independent of all prior steps. This implies, by Theorem 2.11, that the vector obtained in step 8 of the algorithm is $2\varepsilon'$-close to being distributed as $D_{\mathcal{L},s\frac{\sqrt{8d+1}}{q}}$. It remains to show that our algorithm finds $q^{n/d}$ vectors (with high probability). Let $N' = \frac{N}{2}$ be an integer, $X$ be a random variable uniform over $(\mathcal{L}/q\mathcal{L})^{N'}$, and $Y$ be a random variable independent of $X$ and uniform over vectors in $\{0,1\}^{N'}$ with Hamming weight $8d$. The number of such vectors is

$$\binom{N'}{8d} \geq \left(\frac{N'}{8d}\right)^{8d} \geq q^{8n}. \tag{6}$$

Let $U$ be a uniformly random coset of $\mathcal{L}/q\mathcal{L}$. By Lemma 2.26 and (6), we have

$$\mathrm{d}_{\mathrm{SD}}((\langle X,Y\rangle, X),(U,X)) \leq \frac{1}{2}\cdot\sqrt{\frac{q^n}{q^{8n}}} = \frac{1}{2}q^{-7n/2}.$$

By the Markov inequality we have

$$\begin{aligned}\Pr_{x\leftarrow X}\left[\mathrm{d}_{\mathrm{SD}}(\langle x,Y\rangle,U) \geqslant \frac{q^{-n}}{10}\right] &\leqslant \frac{10}{q^{-n}}\mathbb{E}_{x\leftarrow X}\left[\mathrm{d}_{\mathrm{SD}}(\langle x,Y\rangle,U)\right] \\ &= \frac{20}{q^{-n}}\,\mathrm{d}_{\mathrm{SD}}((\langle X,Y\rangle,X),(U,X)) \\ &\leqslant \frac{20}{q^{-n}}\cdot\frac{1}{2}q^{-7n/2} = 10q^{-5n/2}.\end{aligned}$$

Hence, with probability at least $1-10q^{-5n/2}$ over the choice of $x \leftarrow X$, we have that $\mathrm{d}_{\mathrm{SD}}(\langle x,Y\rangle,U) \leqslant \frac{q^{-n}}{10}$, and thus for any $\boldsymbol{v} \in \mathcal{L}/q\mathcal{L}$,

$$q^{-n} + \frac{q^{-n}}{10} > \Pr[\langle x,Y\rangle = \boldsymbol{v} \bmod q\mathcal{L}] > q^{-n} - \frac{q^{-n}}{10}. \tag{7}$$

It follows that, by introducing a statistical distance of at most $10q^{-5n/2}$ on the input, we can assume that the input vectors in list $L_2$ satisfy (7). Notice that after the algorithm found $i$ vectors for any $i < q^{n/d}$, it has removed $8id$ vectors from $L_2$. We will show that for each vector from $L_1$ (which is uniformly sampled from $\mathcal{L}/q\mathcal{L}$) with constant probability we will find $8d$-vectors in step 6. After $i < q^{n/d}$ output vectors have been found, there are $M = N' - 8id$ vectors remaining in the list $L_2$. There are $\binom{M}{8d}$ different $8d$-combinations possible with vectors remaining in $L_2$:

$$\begin{aligned}\binom{N'}{8d}\Big/\binom{M}{8d} &= \frac{N'\cdots(N'-8d+1)}{M\cdots(M-8d+1)} \\ &< \left(\frac{N'-8d}{N'-8d(i+1)}\right)^{8d}\end{aligned} \tag{8}$$

$$
\begin{aligned}
&< \left(1 + \frac{8dq^{n/d}}{N' - 8dq^{n/d} - 8d}\right)^{8d} \\
&= \left(1 + \frac{1}{10d - 1 - \frac{1}{q^{n/d}}}\right)^{8d} \qquad \text{since } N' = 80d^2q^{n/d} \text{ for } C = 1 \\
&\le \left(1 + \frac{1}{10d - 3/2}\right)^{8d} < \frac{5}{2}.
\end{aligned}
$$

At the beginning of the algorithm, there are $\binom{N'}{8d}$ combinations, and hence by (7), each of the $q^n$ cosets appears at least $0.9q^{-n}\binom{N'}{8d}$ times. After $i < q^{n/d}$ output vectors have been found, there are only $\binom{M}{8d}$ combinations left, and $\binom{N'}{8d} - \binom{M}{8d}$ possible combinations have been removed. We say that a coset $\boldsymbol{c}$ disappears if there is no set of $8d$ vectors in $L_2$ that add to $\boldsymbol{c}$. In order for a coset to disappear, all of the at least $0.9q^{-n}\binom{N'}{8d}$ combinations from the initial list must be removed. Hence, the number of cosets that disappear is at most $\frac{\binom{N'}{8d} - \binom{M}{8d}}{0.9q^{-n}\binom{N'}{8d}} < \frac{3/5}{0.9}q^n = \frac{2}{3}q^n$ distinct cosets by (8). Hence, with probability at least $1/3$, we find $8d$ vectors $\boldsymbol{x}_{i_1}, \ldots, \boldsymbol{x}_{i_{8d}}$ from $L_2$ such that $\boldsymbol{x}_{i_1} + \cdots + \boldsymbol{x}_{i_{8d}} - \boldsymbol{v} \in q\mathcal{L}$. By Chernoff–Hoeffding bound with probability greater than $1 - e^{-d^2q^{n/d}}$, the algorithm finds at least $q^{n/d}$ vectors. In total, the statistical distance from the desired distribution is

$$2\varepsilon' N + 2\varepsilon' q^{n/d} + 10 \cdot q^{-5n/2} + e^{-d^2q^{n/d}} \le 4\varepsilon' N + 11 \cdot q^{-5n/2}. \qquad \square$$

COROLLARY 3.2. *Let $n \in \mathbb{N}$, $q \in [4, \sqrt{n}]$ be an integer, and let $\varepsilon = q^{-32n/q^2}$. Let $\mathcal{L}$ be a lattice of rank $n$, and let $s \ge \eta_\varepsilon(\mathcal{L})$. There is an algorithm that outputs a list of vectors that is $q^{-\Omega(n)}$-close to $q^{16n/q^2}$ independent vectors from $D_{\mathcal{L},s}$. The algorithm runs in time $q^{13n+o(n)}$ and requires memory poly$(n) \cdot q^{16n/q^2}$.*

*Proof.* Choose $d$ so that $16d - 16 < q^2 \leqslant 16d$, which is possible when $q \geqslant 4$, and let $\alpha = q/\sqrt{8d+1}$—this is the ratio by which we decrease the Gaussian width in Theorem 3.1—and note that $\alpha \ge 1.2$. Let $p = \lceil 2\sqrt{d}q \rceil < q^2$, and let $k$ be the smallest integer such that $\alpha^k \cdot p \ge 2^{n \log\log n / \log n}$. Thus, $k = O(n \log\log n / \log n)$. Let $g = \alpha^k ps \ge 2^{n \log\log n/\log n} \cdot \eta_\varepsilon(\mathcal{L})$. By Theorem 2.21, in time $N_0 \cdot \text{poly}(n)$, we get $N_0 = (160d^2)^k q^{n/d}$ samples from $D_{\mathcal{L},g}$. We now iterate $k$ times the algorithm from Theorem 3.1. Initially we have $N_0$ vectors. At the beginning of the $i$th iteration for $i \le k-1$, we have $N_i := N_0 \cdot (160d^2)^{-i}$ vectors that are $\Delta_i$-close to being independently distributed from $D_{\mathcal{L},\alpha^{-i}g}$, where $\alpha^{-i}g \geqslant \alpha p \cdot \eta_\varepsilon(\mathcal{L})$. Hence, we can apply Theorem 3.1 and get $N_{i+1} = N_i/160d^2$ vectors that are $\Delta_{i+1}$-close to being independently distributed from $D_{\mathcal{L},\alpha^{-(i+1)}g}$, where $\Delta_{i+1} \leqslant \Delta_i + 4\varepsilon^{2d}N_i + 11(160d^2)^{k-i}q^{-5n/2}$. At each iteration we had $N_i \ge 160d^2q^{n/d}$ vectors, a necessary condition to apply Theorem 3.1. Therefore, after $k$ iterations, we have at least $N_k = N_0/(160d^2)^k = q^{n/d}$ samples that are $\Delta_k$-close to being independently distributed from $D_{\mathcal{L},\alpha^{-k}g}$, where

$$
\begin{aligned}
\Delta_k &\leqslant 11q^{-5n/2} \sum_{i=1}^{k} (160d^2)^{k-i} + \sum_{i=0}^{k-1} 10d\varepsilon^{2d} N_i \\
&\le 11(160d^2)^k q^{-5n/2} + 10dq^{-4n} q^{n/d} \sum_{i=0}^{k-1} (160d^2)^{k-i} \qquad \text{since } 16d \geqslant q^2 \\
&\le \left(11q^{-5n/2} + 10dq^{-4n+n/d}\right)(160d^2)^{k+1} = q^{-5n/2+o(n)} \quad \text{since } (160d^2)^{k+1} = q^{o(n)}.
\end{aligned}
$$

Any vector distributed as $D_{\mathcal{L},ps}$ is in $p\mathcal{L}$ with probability at least $p^{-n}$. We repeat the algorithm $2p^n = O(q^{2n})$ times to obtain $p^n \cdot 2 \cdot q^{n/d}$ vectors that are $2p^n q^{-5n/2+o(n)} = q^{-n/2+o(n)}$ close to $2p^n \cdot q^{n/d}$ independent samples from $D_{\mathcal{L},ps}$. Of these samples obtained, we only keep vectors that fall in $p\mathcal{L}$ and divide them by $p$. Let $M = p^n \cdot 2 \cdot q^{n/d}$. By Chernoff–Hoeffding (Lemma 2.27) with $P = p^{-n}$, and $\delta = \frac{1}{2}$, the probability of obtaining less than $(1-\delta)PM = q^{n/d}$ samples is at most $(\frac{e^{-\delta}}{(1-\delta)^{1-\delta}})^{PM} \leqslant e^{-\frac{1}{10}q^{n/d}}$. Furthermore, $d \leqslant \frac{q^2+16}{16}$ and $q \mapsto \frac{\ln q}{16+q^2}$ is decreasing for $q \geqslant 4$, and hence for $q \leqslant \sqrt{n}$,

$$q^{n/d} \geqslant e^{16n\frac{\ln q}{16+q^2}} \geqslant e^{16n\frac{\ln\sqrt{n}}{16+n}} \geqslant e^{16\ln\sqrt{n}-o(1)} = \Omega(n^8).$$

Hence, with probability greater than $1 - e^{-\frac{1}{10}q^{n/d}} = 1 - q^{-\Omega(n^8)}$, we get $q^{n/d}$ vectors from the distribution $D_{\mathcal{L},s}$. The statistical distance from the desired distribution is $q^{-\Omega(n^8)} + q^{-n/2+o(n)} \leq q^{-n/2+o(n)}$. We repeat this for $\frac{q^{16n/q^2}}{q^{n/d}}$ times to get $q^{16n/q^2}$ vectors. The total statistical distance from the desired distribution is $\frac{q^{16n/q^2}}{q^{n/d}} \cdot q^{-n/2+o(n)} \leq q^{-\Omega(n)}$. The total running time is bounded by

$$q^{2n}\left(\frac{q^{16n/q^2}}{q^{n/d}}\right)\left(\mathrm{poly}(n)\cdot N_0 + \sum_{i=0}^{k-1}(10ed)^{8d}\cdot(160d^2)^{k-i}q^{8n+n/d+o(n)}\right) \leqslant q^{13n+o(n)}.$$

The memory usage is slightly more involved: we can think of the $k$ iterations as a pipeline with $k$ intermediate lists, and we observe that as soon as a list (at any level) has more than $160d^2q^{16n/q^2}$ elements, we can apply Theorem 3.1 to produce $q^{16n/q^2}$ vectors at the next level. Hence, we can ensure that at any time, each level contains at most $160d^2q^{16n/q^2}$ vectors, so in total we only need to store at most $k \cdot 160d^2q^{16n/q^2} = \mathrm{poly}(n)q^{16n/q^2}$ vectors, to which we add the memory usage of the algorithm of Theorem 3.1 which is bounded by $\mathrm{poly}(n) \cdot q^{n/d} \leqslant \mathrm{poly}(n) \cdot q^{16n/q^2}$. Finally, we run the filter ($p\mathcal{L}$) on the fly at the end of the $k$ iterations to avoid storing useless samples. ∎

This tradeoff works for any $q \geq 4$, and the running time can be bounded by $c_1^{n+o(n)} \cdot q^{c_2 n}$ for some constants $c_1$ and $c_2$ that we have not tried to optimize.

### 3.2. Algorithms for BDD and SVP.

THEOREM 3.3. *Let $n \in \mathbb{N}, q \in [4, \sqrt{n}]$ be a positive integer. Let $\mathcal{L}$ be a lattice of rank $n$. There is a randomized algorithm that solves $0.1/q$-BDD in time $q^{13n+o(n)}$ and requires memory poly$(n) \cdot q^{16n/q^2}$.*

*Proof.* Let $\varepsilon = q^{\frac{-32n}{q^2}}$ and $s \geqslant \eta_\varepsilon(\mathcal{L}^*)$. From Corollary 3.2, there exists an algorithm that outputs $q^{16n/q^2}$ vectors whose distribution is statistically close to $D_{\mathcal{L}^*,s}$ in time $q^{13n+o(n)}$ and space $\mathrm{poly}(n) \cdot q^{16n/q^2}$. By repeating this algorithm $\mathrm{poly}(n)$ times, we can therefore build a $\mathsf{DGS}_{\eta_\varepsilon}^{\mathrm{poly}(n)q^{16n}/q^2}$ oracle such that each call takes time $q^{13n+o(n)}$ and space $\mathrm{poly}(n) \cdot q^{16n/q^2}$. By Theorem 5.2, we can construct a $\alpha$-BDD such that each call takes time $m \cdot \mathrm{poly}(n)$ and space $\mathrm{poly}(n)$, where $\alpha = \phi(\mathcal{L})/\lambda_1(\mathcal{L}) = \frac{\sqrt{\ln(1/\varepsilon)/\pi - o(1)}}{2\eta_\varepsilon(\mathcal{L}^*)\lambda_1(\mathcal{L})}$ and $m = O(\frac{n\log(1/\varepsilon)}{\sqrt{\varepsilon}}) = O(\frac{n^2}{q^2}q^{16n/q^2}\log q) \leqslant \mathrm{poly}(n)q^{16n/q^2}$. The preprocessing consists of $\mathrm{poly}(n)$ calls to the $\mathsf{DGS}_{\eta_\varepsilon}^m$ sampler described above and requires space $m \cdot \mathrm{poly}(n)$. Hence, the total complexity is $\mathrm{poly}(n)q^{13n+o(n)} = q^{13n+o(n)}$ in time

and $m \cdot \mathrm{poly}(n) = \mathrm{poly}(n) q^{16n/q^2}$ in space. By using inequality (4), in Lemma 2.16, we have that

$$\lambda_1(\mathcal{L})\eta_\varepsilon(\mathcal{L}^*) < \sqrt{\frac{\beta(\mathcal{L})^2 n}{2\pi e}} \cdot \varepsilon^{-1/n}(1 + o(1)).$$

Hence, we can guarantee that

$$\alpha(\mathcal{L}) \geq \sqrt{\frac{\ln(1/\varepsilon)}{2n(\beta^2/e)\varepsilon^{-2/n}}} \cdot (1 - o(1)) \geq \frac{1}{q}\sqrt{\frac{32 \cdot e \cdot \ln q}{2\beta^2 q^{64/q^2}}} \cdot (1 - o(1)) \geq (10q)^{-1}. \qquad \square$$

THEOREM 3.4. *Let $n \in \mathbb{N}, q \in [4, \sqrt{n}]$ be a positive integer. Let $\mathcal{L}$ be a lattice of rank $n$. There is a randomized algorithm that solves* SVP *in time $q^{13n+o(n)}$ and in space poly$(n) \cdot q^{\frac{16n}{q^2}}$.*

*Proof.* By Theorem 3.3, we can construct a $\frac{0.1}{q}$-BDD oracle in time $q^{13n+o(n)}$ and in space $\mathrm{poly}(n) \cdot q^{\frac{16n}{q^2}}$. Each execution of the BBD oracle now takes $m = \mathcal{O}(\frac{n^2}{q^2} q^{16n/q^2})$ time. By Theorem 2.19, with $(10q)^n$ queries to a $\frac{0.1}{q}$-BDD oracle, we can find the shortest vector. The total time complexity is $q^{13n+o(n)} + \frac{n^2}{q^2} q^{16n/q^2} \cdot (10q)^n = q^{13n+o(n)}$. $\square$

*Remark* 3.5. If we take $q = \sqrt{n}$, Theorem 3.4 gives an SVP algorithm that takes $n^{\mathcal{O}(n)}$ time and $\mathrm{poly}(n)$ space. The constant in the exponent of time complexity is worse than the best enumeration algorithms. When $q$ is a large enough constant, for any constant $\varepsilon > 0$, there exists a constant $C = C(\varepsilon) > 2$, such that there is a $2^{Cn}$ time and $2^{\varepsilon n}$ space algorithm for DGS and SVP. In particular, the time complexity of the algorithm in this regime is worse than the best sieving algorithms.

*Remark* 3.6. In [1], the authors gave $2^{\varepsilon n}$ time reductions between the lattice problems in different $\ell_p$ norms. Their reduction increases the approximation factor by a constant that depends on $\varepsilon$. Combining their results with the above theorem gives an algorithm with a time-memory tradeoff for the constant-factor approximation of all lattice problems.

**4. Quantum speedup for BDDP using QRAM.** The goal of this section is to obtain a quantum speedup of Theorem 2.14. We improve the time complexity of the algorithm by almost the square root factor, but we also require the advice string to be stored in QRAM. Before presenting our contribution, we will give an overview of the known algorithms for BDDP.

Most of the known BDDP algorithms (including the one in [27]) are based on the algorithm for decision-CVPP by Aharonov and Regev [7]. We first revisit the algorithm for decision-CVPP. In this work, the authors introduced the *periodic Gaussian function* $f : \mathcal{R}^n \to \mathcal{R}^+$,

$$f(\boldsymbol{t}) := \frac{\rho(\boldsymbol{t} + \mathcal{L})}{\rho(\mathcal{L})},$$

towards giving an algorithm for $\mathcal{O}(\sqrt{n/\log n})$ approximation of decision-CVPP. They observed that by the Poisson summation formula, we get the identity

$$f(\boldsymbol{t}) = \mathop{\mathbb{E}}_{\boldsymbol{w} \sim \mathcal{D}_{\mathcal{L}^*}} [\cos(2\pi \langle \boldsymbol{w}, \boldsymbol{t} \rangle)]. \tag{9}$$

They also showed that when distance of the target vector $\boldsymbol{t}$ from lattice $\mathcal{L}$ is at least $\sqrt{n}$, then $f(\boldsymbol{t})$ is negligible, and when distance between $\boldsymbol{t}$ and lattice $\mathcal{L}$ is at most

$\sqrt{\log n}$, then $f(\boldsymbol{t})$ is nonnegligible. This function $f$ evaluated on any vector $\boldsymbol{t}$ is an infinite sum and is not easy to evaluate efficiently. Their algorithm crucially relied on the observation in Equation (9) that shows that the function $f$ can be estimated by using a polynomial size advice string with at most $1/\mathrm{poly}(n)$ error. They gave the estimator

$$f_W(\boldsymbol{t}) = \frac{1}{N}\sum_{i=1}^{N}\cos(2\pi\langle \boldsymbol{w}_i, \boldsymbol{t}\rangle). \tag{10}$$

where $W = (\boldsymbol{w}_1, \ldots, \boldsymbol{w}_N) \in \mathcal{L}^*$ are i.i.d. samples from $\mathcal{D}_{\mathcal{L}^*}$ and showed that $f_W \approx f$ with at most $1/\mathrm{poly}(n)$ error when $N$ is a large enough number bounded by a polynomial in $n$.

Later, Liu, Lyubashevsky, and Micciancio [57] gave an algorithm for the approximation of search BDDP. The idea is to iteratively update the target vector $\boldsymbol{t}$ such that its distance from the closest lattice vector decreases and eventually it is easy to efficiently find the closest lattice vector. They are able to solve the $\alpha$-BDDP for $\alpha \le \mathcal{O}(\sqrt{\frac{\log n}{n}})$. Dadush, Regev, and Stephens-Davidowitz gave an improvement by a careful analysis of the function $f(\boldsymbol{t})$. They proposed that by iteratively updating $\boldsymbol{t}$ by an approximation of

$$\boldsymbol{t} + \frac{\nabla f(\boldsymbol{t})}{2\pi f(\boldsymbol{t})},$$

we can go near the closest lattice vector. Their algorithm solves $\alpha$-BDDP for $\alpha = \frac{\sqrt{\ln(1/\varepsilon)/\pi - o(1)}}{2\eta_\varepsilon(\mathcal{L}^*)\lambda_1(\mathcal{L})}$. The advice string consists of $N$ vectors from $\mathcal{D}_{\mathcal{L}^*,\eta_\varepsilon(\mathcal{L}^*)}$, and the algorithm performs $\mathcal{O}(N + \mathrm{poly}(n))$ arithmetic operations where $N = \mathcal{O}(\frac{n\log(1/\varepsilon)}{\sqrt{\varepsilon}})$. In this section, we will show that if the advice string is stored in QRAM, then we can achieve the same approximation of BDDP by using only $\mathcal{O}(\sqrt{N} + \mathrm{poly}(n))$ arithmetic operations.

We will start with listing some of the lemmas and theorems from [27] that we will directly use in our proof. After that, we will show the quantum improvement in the estimation of functions $f_W$ and $\nabla f_W$. In the last part of this section, we will present the main result of this section.

### 4.1. Results from [27].

LEMMA 4.1 (see [27, Lemma 2.14]). *Let $\mathcal{L} \subset \mathbb{R}^n$ be a lattice of rank $n$. Then, for all $\boldsymbol{t} \in \mathbb{R}^n$, $f(\boldsymbol{t}) \ge \rho(\boldsymbol{t})$.*

LEMMA 4.2 (see [27, Lemma 4.7]). *Let $\mathcal{L} \subset \mathbb{R}^n$ be a lattice with $\rho(\mathcal{L}) = 1 + \varepsilon$ for $\varepsilon \in (0, 1/400)$. Let $W = (\boldsymbol{w}_1, \ldots, \boldsymbol{w}_N)$ be sampled independently from $\mathcal{D}_{\mathcal{L}^*}$ with $N \ge \Omega(n/\sqrt{\varepsilon})$. Then*

$$\Pr[\exists \boldsymbol{t}, \|\boldsymbol{t}\| \le \varepsilon^{1/8}/(1000n) : \|\nabla f_W(\boldsymbol{t})/(2\pi f_W(\boldsymbol{t})) + \boldsymbol{t}\| > \varepsilon^{0.25}\|\boldsymbol{t}\|] \le 2^{-\Omega(n)}.$$

LEMMA 4.3 (see [27, Lemma 4.10]). *Let $\mathcal{L} \subset \mathbb{R}^n$ be a lattice with $\rho(\mathcal{L}) = 1 + \varepsilon$, with $\varepsilon \in (0, 1/400)$. Let $s_\varepsilon = (\frac{1}{\pi}\ln\frac{2(1+\varepsilon)}{\varepsilon})^{0.5}$. Let $W = (\boldsymbol{w}_1, \ldots, \boldsymbol{w}_N)$ be sampled independently from $\mathcal{D}_{\mathcal{L}^*}$. Then, for $\varepsilon^2 \le s \le 10$, if $N \ge \Omega(n\ln(1/\varepsilon)/s^2)$, then*

$$\Pr[\exists \boldsymbol{t} \in \mathbb{R}^n, \varepsilon^{1/8}/(1000n) \le \|\boldsymbol{t}\| \le s_\varepsilon : \|\nabla f_W(\boldsymbol{t}) - \nabla f(\boldsymbol{t})\| > s\|\boldsymbol{t}\|] \le 2^{-\Omega(N\cdot s^2)}.$$

LEMMA 4.4 (see [27, Lemma 4.12]). *Let $\mathcal{L} \subset \mathbb{R}^n$ be a lattice with $\rho(\mathcal{L}) = 1 + \varepsilon$, with $\varepsilon \in (0, 1/400)$. Let $s_\varepsilon = (\frac{1}{\pi}\ln\frac{2(1+\varepsilon)}{\varepsilon})^{0.5}$. Let $W = (\boldsymbol{w}_1, \ldots, \boldsymbol{w}_N)$ be sampled independently from $\mathcal{D}_{\mathcal{L}^*}$. Then, for $\varepsilon^2 \le s \le 10$, if $N \ge \Omega(n\ln(1/\varepsilon)/s^2)$, then*

$$\Pr[\exists \boldsymbol{t} \in \mathbb{R}^n, \|\boldsymbol{t}\| \le s_\varepsilon : \|f_W(\boldsymbol{t}) - f(\boldsymbol{t})\| > s] \le 2^{-\Omega(N\cdot s^2)}.$$

LEMMA 4.5 (see [27, part 1 of Lemma 4.5]). *Let $\mathcal{L} \subset \mathbb{R}^n$ be a lattice with $\rho(\mathcal{L}) = 1+\varepsilon$ for some $\varepsilon > 0$, and let $W = (\boldsymbol{w}_1, \ldots, \boldsymbol{w}_N)$ be sampled independently from $\mathcal{D}_{\mathcal{L}^*}$. Then, for $s \geq 0$, $N \min(s, s^2) \geq \Omega(n)$, and $\Delta_\varepsilon = \frac{4\pi\varepsilon}{1+\varepsilon}(\ln \frac{2+2\varepsilon}{\varepsilon} + 1)$, we have*

$$Pr[\|Hf_W(\mathbf{0}) + 2\pi \mathbf{I}_n\| > \Delta_\varepsilon + s] \leq 2^{-\Omega(N \min s, s^2)},$$

*where $Hf_W$ denotes the Hessian matrix of $f_W$.*

LEMMA 4.6 (see [27, part 1 of Lemma 4.7]). *Let $\mathcal{L} \subset \mathbb{R}^n$ be a lattice with $\rho(\mathcal{L}) = 1+\varepsilon$ for some $\varepsilon > 0$, and let $W = (\boldsymbol{w}_1, \ldots, \boldsymbol{w}_N)$ be sampled independently from $\mathcal{D}_{\mathcal{L}^*}$. Then we have*

$$Pr\left[\exists \boldsymbol{t} \in \mathbb{R}^n, \|\boldsymbol{t}\| \leq \frac{\varepsilon^{1/8}}{1000n} : \|\nabla f_W(\boldsymbol{t})\| > (2\pi + 4\varepsilon^{0.25})\|\boldsymbol{t}\|\right] \leq 2^{-\Omega(n)}$$

*and*

$$Pr\left[\exists \boldsymbol{t} \in \mathbb{R}^n, \|\boldsymbol{t}\| \leq \frac{\varepsilon^{1/8}}{1000n} : |f_W(\boldsymbol{t})| < 1 - \frac{\varepsilon^{0.25}}{100}\right] \leq 2^{-\Omega(n)}.$$

The following theorem tells us that if we can compute $f(\boldsymbol{t})$ and its gradient and suppose that $t$ is sufficiently close to the lattice, then we can simply apply the gradient ascent algorithm to find a vector $\boldsymbol{t}' = \frac{\nabla f(\boldsymbol{t})}{2\pi f(\boldsymbol{t})} + \boldsymbol{t}$ such that the new vector $\boldsymbol{t}'$ is even closer to the closest lattice vector of $\boldsymbol{t}$.

THEOREM 4.7 (see [27, Corollary 4.3]). *Let $\varepsilon \in (0, 1/400)$, and let $\mathcal{L} \subset \mathbb{R}^n$ be a lattice with $\rho(\mathcal{L}) = 1+\varepsilon$. Let $s_\varepsilon = (\frac{1}{\pi} \ln \frac{2(1+\varepsilon)}{\varepsilon})^{1/2}$. Then, for all $\boldsymbol{t} \in \mathbb{R}^n$ satisfying $\|\boldsymbol{t}\| \leq s_\varepsilon/2$,*

$$\left\| \frac{\nabla f(\boldsymbol{t})}{2\pi f(\boldsymbol{t})} + \boldsymbol{t} \right\| \leq 12(\varepsilon/2)^{1-2\delta(\boldsymbol{t})}\|\boldsymbol{t}\|,$$

*where $\delta(t) = \max(1/8, \|\boldsymbol{t}\|/s_\varepsilon)$. In particular, for $\delta(\boldsymbol{t}) \leq 1/2 - 2/(\pi s_\varepsilon^2)$,*

$$\left\| \frac{\nabla f(\boldsymbol{t})}{2\pi f(\boldsymbol{t})} + \boldsymbol{t} \right\| \leq \|\boldsymbol{t}\|/4.$$

### 4.2. Estimation of $f_W$ and $\nabla f_W$ in QRAM.

THEOREM 4.8 (amplitude estimation [22], follows from Theorem 12). *Let $\delta > 0$. Given natural number $M$ and access to an $(n+1)$-qubit unitary $U$ satisfying*

$$U|0^n\rangle|0\rangle = \sqrt{a}\,|\phi_1\rangle|1\rangle + \sqrt{1-a}\,|\phi_0\rangle|0\rangle,$$

*where $|\phi_1\rangle$ and $|\phi_0\rangle$ are arbitrary $n$-qubit states and $0 < a < 1$, there exists a quantum algorithm that uses $\mathcal{O}(M \cdot \log(1/\delta))$ applications of $U$ and $U^\dagger$ and $\tilde{\mathcal{O}}(M \cdot \log(1/\delta))$ time and outputs an estimate $\tilde{a}$ that with probability $\geq 1-\delta$ satisfies*[5]

$$|a - \tilde{a}| \leq \frac{1}{M}.$$

[5]The original Theorem 12 in [22] considered an estimator that output $\tilde{a}$ with additive error at most $1/M$ with probability $\geq 2/3$. One can repeat this process for $\mathcal{O}(\log(1/\delta))$ times and take the median number to increase the success probability to $\geq 1-\delta$.

THEOREM 4.9. *Let $N$ be a positive integer, let $\mathcal{L} \subset \mathbb{R}^n$ be a lattice, and let $W = \{\boldsymbol{w}_1, \ldots, \boldsymbol{w}_N\}$ be the set of vectors from $\mathcal{L}^*$. Let $O_W : |j\rangle |0\rangle \to |j\rangle |\boldsymbol{w}_j\rangle$. For any $\varepsilon, \delta > 0$, there exists a quantum algorithm that given target $\boldsymbol{t} \in \mathbb{R}^n$ outputs $\tilde{f}_W(\boldsymbol{t})$ which satisfy $|\tilde{f}_W(\boldsymbol{t}) - f_W(\boldsymbol{t})| \le \varepsilon$ with probability $1 - \delta$. The algorithm makes $\mathcal{O}(\varepsilon^{-1} \cdot \log \frac{1}{\delta})$ queries to $O_W$ and requires $\varepsilon^{-1} \cdot \log \frac{1}{\delta} \cdot poly(\log n)$ elementary quantum gates.*

*Proof.* We define the positive controlled rotation oracle as, for any $a \in [-1, 1]$

$$O_{CR^+} : |a\rangle|0\rangle \to \begin{cases} |a\rangle(\sqrt{a}|1\rangle + \sqrt{1-a}|0\rangle) & \text{if } a \ge 0, \\ |a\rangle|0\rangle & \text{otherwise,} \end{cases}$$

which can be implemented up to negligible error by $\mathrm{poly}(\log n)$ quantum elementary gates. Also, we define the cosine inner product oracle as for any $\boldsymbol{t}, \boldsymbol{w} \in \mathbb{R}^n$

$$O_{\cos} : |\boldsymbol{w}\rangle |\boldsymbol{t}\rangle |0\rangle \to |\boldsymbol{w}\rangle |\boldsymbol{t}\rangle |\cos(2\pi\langle \boldsymbol{w}, \boldsymbol{t}\rangle)\rangle ,$$

which can also be implemented by $\mathrm{poly}(\log n)$ quantum elementary gates. Preparing the state $\frac{1}{\sqrt{N}} \sum_{j=1}^{N} |j\rangle |\boldsymbol{0}\rangle |\boldsymbol{t}\rangle |0\rangle |0\rangle$, and then applying $O_W$ on the first and second registers, applying $O_{\cos}$ on the second, third, and fourth registers, and applying $O_{CR^+}$ on the fourth and fifth registers, we have

$$\begin{aligned} &\frac{1}{\sqrt{N}} \sum_{\substack{j \in [N] \text{ and} \\ \cos(2\pi\langle \boldsymbol{w}_j, \boldsymbol{t}\rangle) > 0}} |j\rangle |\boldsymbol{w}_j\rangle |\boldsymbol{t}\rangle |\cos(2\pi\langle \boldsymbol{w}_j, \boldsymbol{t}\rangle)\rangle \Big( \sqrt{\cos(2\pi\langle \boldsymbol{w}_j, \boldsymbol{t}\rangle)} |1\rangle \\ &\quad + \sqrt{1 - \cos(2\pi\langle \boldsymbol{w}_j, \boldsymbol{t}\rangle)} |0\rangle \Big) + \frac{1}{\sqrt{N}} \sum_{\substack{j \in [N] \text{ and} \\ \cos(2\pi\langle \boldsymbol{w}_j, \boldsymbol{t}\rangle) \le 0}} |j\rangle |\boldsymbol{w}_j\rangle |\boldsymbol{t}\rangle |\cos(2\pi\langle \boldsymbol{w}_j, \boldsymbol{t}\rangle)\rangle |0\rangle . \end{aligned}$$

By rearranging the equation, we have

$$\begin{aligned} &\frac{1}{\sqrt{N}} \sum_{\substack{j \in [N] \text{ and} \\ \cos(2\pi\langle \boldsymbol{w}_j, \boldsymbol{t}\rangle) > 0}} \sqrt{\cos(2\pi\langle \boldsymbol{w}_j, \boldsymbol{t}\rangle)} |j\rangle |\boldsymbol{w}_j\rangle |\boldsymbol{t}\rangle |\cos(2\pi\langle \boldsymbol{w}_j, \boldsymbol{t}\rangle)\rangle |1\rangle \\ &\quad + \frac{1}{\sqrt{N}} \Bigg( \sum_{\substack{j \in [N] \text{ and} \\ \cos(2\pi\langle \boldsymbol{w}_j, \boldsymbol{t}\rangle) > 0}} \sqrt{1 - \cos(2\pi\langle \boldsymbol{w}_j, \boldsymbol{t}\rangle)} |j\rangle |\boldsymbol{w}_j\rangle |\boldsymbol{t}\rangle |\cos(2\pi\langle \boldsymbol{w}_j, \boldsymbol{t}\rangle)\rangle \\ &\quad + \sum_{\substack{j \in [N] \text{ and} \\ \cos(2\pi\langle \boldsymbol{w}_j, \boldsymbol{t}\rangle \le 0}} |j\rangle |\boldsymbol{w}_j\rangle |\boldsymbol{t}\rangle |\cos(2\pi\langle \boldsymbol{w}_j, \boldsymbol{t}\rangle)\rangle \Bigg) |0\rangle \\ &= \sqrt{a^+} |\phi_1\rangle |1\rangle + \sqrt{1 - a^+} |\phi_0\rangle |0\rangle , \end{aligned}$$

where $a^+ = \sum_{\substack{j \in [N] \text{ and} \\ \cos(2\pi\langle \boldsymbol{w}_j, \boldsymbol{t}\rangle) > 0}} \frac{\cos(2\pi\langle \boldsymbol{w}_j, \boldsymbol{t}\rangle)}{N}$. By applying Theorem 4.8, we can estimate $a^+$ with additive error $\varepsilon/2$ by using $\mathcal{O}(\varepsilon^{-1})$ applications of $O_W$, $O_W^\dagger$, and $\varepsilon^{-1} \cdot \mathrm{poly}(\log n)$ elementary quantum gates. Following the same strategy, we can also estimate $a^- = \sum_{\substack{j \in [N] \text{ and} \\ \cos(2\pi\langle w_j, t\rangle) < 0}} \frac{\cos(2\pi\langle w_j, t\rangle)}{N}$ with the same additive error and by using the same number of queries and quantum elementary gates. Therefore, we can estimate $a^+ + a^- = \sum_{j \in [N]} \frac{\cos(2\pi\langle w_j, t\rangle)}{N}$ with additive error $\varepsilon$. By repeating the procedure $\Theta(\log \frac{1}{\delta})$ times and taking the median among them, we finish the proof. ∎

THEOREM 4.10. *Let $N$ be a positive integer, let $\mathcal{L} \subset \mathbb{R}^n$ be a lattice, and let$W = \{\boldsymbol{w}_1, \ldots, \boldsymbol{w}_N\}$ be the set of vectors in $\mathcal{L}^*$. Let $O_W : |j\rangle |0\rangle \to |j\rangle |\boldsymbol{w}_j\rangle$. For any $\varepsilon, \delta > 0$, there exists a quantum algorithm that given target $\boldsymbol{t} \in \mathbb{R}^n$ and $w_{max} \geq \max_{j\in[N]} \|\boldsymbol{w}_j\|$ outputs $\nabla_i \tilde{f}_W(\boldsymbol{t})$[6] for any $i \in [n]$ which satisfy $|\nabla_i \tilde{f}_W(\boldsymbol{t}) - \nabla_i f_W(\boldsymbol{t})| \leq 2\pi\varepsilon\|\boldsymbol{t}\| \cdot w_{max}^2$ with probability $1-\delta$. The algorithm makes $O(\varepsilon^{-1} \cdot \log \frac{1}{\delta})$ queries to $O_W$ and requires $\varepsilon^{-1} \cdot \log \frac{1}{\delta} \cdot poly(\log n)$ elementary quantum gates.*

*Proof.* From (10), we get $\nabla_i f_W(\boldsymbol{t}) = \frac{-1}{N} \sum_{j=1}^{N} \sin(2\pi\langle \boldsymbol{w}_j, \boldsymbol{t}\rangle) \cdot (w_j)_i$ for $i \in [n]$. We observe that for all $i \in [n]$ and $j \in [N]$,

$$|\sin(2\pi\langle \boldsymbol{w}_j, \boldsymbol{t}\rangle) \cdot (w_j)_i| \leq |2\pi\langle \boldsymbol{w}_j, \boldsymbol{t}\rangle \cdot w_{max}| \leq 2\pi\|\boldsymbol{t}\| \cdot w_{max}^2,$$

and hence $|\frac{\sin(2\pi\langle \boldsymbol{w}_j, \boldsymbol{t}\rangle)\cdot(w_j)_i}{2\pi\|\boldsymbol{t}\|\cdot w_{max}^2}| \leq 1$ for all $i \in [n]$ and $j \in [N]$. For simplicity, we define $g_i(\boldsymbol{w}, \boldsymbol{t}) = \frac{\sin(2\pi\langle \boldsymbol{w}, \boldsymbol{t}\rangle)\cdot(w)_i}{2\pi\|\boldsymbol{t}\|\cdot w_{max}^2}$. Then we define the sine inner product oracle as for any $\boldsymbol{t}, \boldsymbol{w} \in \mathbb{R}^n$ and $i \in [n]$

$$O_{\sin} : |\boldsymbol{w}\rangle |\boldsymbol{t}\rangle |i\rangle |0\rangle \to |\boldsymbol{w}\rangle |\boldsymbol{t}\rangle |i\rangle |g_i(\boldsymbol{w}, \boldsymbol{t})\rangle,$$

which can be implemented by $\mathrm{poly}(\log n)$ quantum elementary gates. Also, we define the positive controlled rotation oracle as for any $a \in \mathbb{R}$

$$O_{CR^+} : |a\rangle|0\rangle \to \begin{cases} |a\rangle(\sqrt{a}|1\rangle + \sqrt{1-a}|0\rangle) & \text{if } a \geq 0, \\ |a\rangle|0\rangle & \text{otherwise,} \end{cases}$$

which can be implemented by $\mathrm{poly}(\log n)$ quantum elementary gates.

Preparing the state $\frac{1}{\sqrt{N}} \sum_{j=1}^{N} |j\rangle |\boldsymbol{0}\rangle |\boldsymbol{t}\rangle |i\rangle |0\rangle |0\rangle$, and then applying $O_W$ on the first and second registers, applying $O_{\sin}$ on the second, third, fourth, and fifth registers, and applying $O_{CR^+}$ on the fifth and sixth registers, we have

$$\begin{aligned}
&\frac{1}{\sqrt{N}} \sum_{\substack{j\in[N] \text{ and} \\ g_i(w_j,t)>0}} |j\rangle |\boldsymbol{w}_j\rangle |\boldsymbol{t}\rangle |i\rangle |g_i(\boldsymbol{w}_j, \boldsymbol{t})\rangle \left(\sqrt{g_i(\boldsymbol{w}_j, \boldsymbol{t})}\,|1\rangle + \sqrt{1 - g_i(\boldsymbol{w}_j, \boldsymbol{t})}\,|0\rangle\right) \\
&\quad + \frac{1}{\sqrt{N}} \sum_{\substack{j\in[N] \text{ and} \\ g_i(\boldsymbol{w}_j,\boldsymbol{t})\leq 0}} |j\rangle |\boldsymbol{w}_j\rangle |\boldsymbol{t}\rangle |i\rangle |g_i(\boldsymbol{w}_j, \boldsymbol{t})\rangle |0\rangle \\
&= \frac{1}{\sqrt{N}} \sum_{\substack{j\in[N] \text{ and} \\ g_i(\boldsymbol{w}_j,\boldsymbol{t})>0}} \sqrt{g_i(\boldsymbol{w}_j, \boldsymbol{t})}\,|j\rangle |\boldsymbol{w}_j\rangle |\boldsymbol{t}\rangle |i\rangle |g_i(\boldsymbol{w}_j, \boldsymbol{t})\rangle |1\rangle \\
&\quad + \frac{1}{\sqrt{N}} \Bigg( \sum_{\substack{j\in[N] \text{ and} \\ g_i(\boldsymbol{w}_j,\boldsymbol{t})>0}} \sqrt{1 - g_i(\boldsymbol{w}_j, \boldsymbol{t})}\,|j\rangle |\boldsymbol{w}_j\rangle |\boldsymbol{t}\rangle |i\rangle |g_i(\boldsymbol{w}_j, \boldsymbol{t})\rangle \\
&\qquad\quad + \sum_{\substack{j\in[N] \text{ and} \\ g_i(\boldsymbol{w}_j,\boldsymbol{t})\leq 0}} |j\rangle |\boldsymbol{w}_j\rangle |\boldsymbol{t}\rangle |i\rangle |g_i(\boldsymbol{w}_j, \boldsymbol{t})\rangle \Bigg) |0\rangle \\
&= \sqrt{a^+}\,|\phi_1\rangle |1\rangle + \sqrt{1 - a^+}\,|\phi_0\rangle |0\rangle,
\end{aligned}$$

[6] We are abusing notation here: $\nabla_i \tilde{f}_W(\boldsymbol{t})$ just represent an estimated value of $\nabla_i f_W(\boldsymbol{t})$.

where $a^+ = \sum_{\substack{j\in[N] \text{ and} \\ g_i(\boldsymbol{w}_j,\boldsymbol{t})>0}} \frac{g_i(\boldsymbol{w}_j,\boldsymbol{t})}{N}$. By applying Theorem 4.8, we can estimate $a^+$ with additive error $\varepsilon/2$ by using $\mathcal{O}(\varepsilon^{-1})$ applications of $O_W$, $O_W^\dagger$, and $\varepsilon^{-1}\cdot\mathrm{poly}(\log n)$ elementary quantum gates. Following the same strategy, we can also estimate $a^- = \sum_{\substack{j\in[N] \text{ and} \\ g_i(\boldsymbol{w}_j,\boldsymbol{t})<0}} \frac{g_i(\boldsymbol{w}_j,\boldsymbol{t})}{N}$ with the same additive error and by using the same number of queries and quantum elementary gates. Therefore, we can estimate $a^+ + a^- = \sum_{j\in[N]} \frac{g_i(\boldsymbol{w}_j,\boldsymbol{t})}{N} = \frac{-\nabla_i f_W(\boldsymbol{t})}{2\pi\|\boldsymbol{t}\|\cdot w_{max}^2}$ with additive error $\varepsilon$. By repeating the procedure $\Theta(\log\frac{1}{\delta})$ times and taking the median among them, we finish the proof. ∎

We get the following corollary.

COROLLARY 4.11. *Let $\varepsilon \in (0,/100)$, let $N = \lceil (n^8/\sqrt{\varepsilon})\rceil$ be a positive integer, let $\mathcal{L}\subset\mathbb{R}^n$ be a lattice, and let $W = \{\boldsymbol{w}_1,\dots,\boldsymbol{w}_N\}$ be the set of vectors from $\mathcal{L}^*$ such that $\forall i\in[N], \|\boldsymbol{w}_i\| \le 2n+1$. Let $O_W : |j\rangle|0\rangle \to |j\rangle|\boldsymbol{w}_j\rangle$. There exists a quantum algorithm that given target $\boldsymbol{t}\in\mathbb{R}^n$ outputs $\tilde{f}_W(\boldsymbol{t})$ and $\nabla\tilde{f}_W(\boldsymbol{t})$ which satisfy*

$$|\tilde{f}_W(\boldsymbol{t}) - f_W(\boldsymbol{t})| \le \frac{\varepsilon^{1/4}}{100} \text{ and } \|\nabla\tilde{f}_W(\boldsymbol{t}) - \nabla f_W(\boldsymbol{t})\| \le \frac{\varepsilon^{1/4}}{100}\|\boldsymbol{t}\|$$

*with probability $1-2^{-\Omega(n)}$. The algorithm make $\mathcal{O}(\varepsilon^{-1/4}\cdot n^4)$ queries to $O_W$ and require $\varepsilon^{-1/4}\cdot n^4\cdot poly(\log n)$ elementary quantum gates.*

**4.3. BDDP in QRAM.** From the previous subsection, given $\boldsymbol{t}$, we can estimate $f_W(\boldsymbol{t})$ with small additive error. In this subsection, we will replace $f_W(\boldsymbol{t})$ with the approximation function $\tilde{f}_W(\boldsymbol{t})$, and show that doing gradient ascent on the approximation function $\tilde{f}_W(\boldsymbol{t})$ still helps us to find the closest vector, and hence we can use $\tilde{f}_W$ to solve BDDP.

THEOREM 4.12. *Let $\mathcal{L}\subset\mathbb{R}^n$ be a lattice with $\rho(\mathcal{L}) = 1+\varepsilon$, with $\varepsilon\in(0,1/400)$, and let $N$ be a positive integer. Let $s_\varepsilon = (\frac{1}{\pi}\ln\frac{2(1+\varepsilon)}{\varepsilon})^{0.5}$, let $\delta_{\max} = 0.5 - \frac{2}{\pi s_\varepsilon^2}$, and let $W = (\boldsymbol{w}_1,\dots,\boldsymbol{w}_N)$ be a set of vectors from $\mathcal{L}^*$. Suppose that for some $\gamma>0$, $\boldsymbol{t}\in\mathbb{R}^n$, we can compute $\tilde{f}_W(\boldsymbol{t})$ and $\nabla\tilde{f}_W(\boldsymbol{t})$, and the following hold:*

1. $\|\boldsymbol{t}\| \le \min\{\delta_{\max}s_\varepsilon, \sqrt{\ln(1/(4\gamma))/\pi}\}$,
2. $\|\nabla f_W(\boldsymbol{t}) - \nabla f(\boldsymbol{t})\| \le \frac{\pi}{2}\gamma\|\boldsymbol{t}\|$,
3. $\|\nabla f_W(\boldsymbol{t}) - \nabla\tilde{f}_W(\boldsymbol{t})\| \le \frac{\pi}{2}\gamma\|\boldsymbol{t}\|$,
4. $|f_W(\boldsymbol{t}) - f(\boldsymbol{t})| \le \gamma$,
5. $|f_W(\boldsymbol{t}) - \tilde{f}_W(\boldsymbol{t})| \le \gamma$.

*Then*

$$\left\| \frac{\nabla\tilde{f}_W(\boldsymbol{t})}{2\pi\tilde{f}_W(\boldsymbol{t})} - \frac{\nabla f(\boldsymbol{t})}{2\pi f(\boldsymbol{t})} \right\| \le \frac{6\gamma}{\rho(\boldsymbol{t})}\|\boldsymbol{t}\|.$$

*Proof.* It's easy to see that the above conditions imply

$$\|\nabla\tilde{f}_W(\boldsymbol{t}) - \nabla f(\boldsymbol{t})\| \le \pi\gamma\|\boldsymbol{t}\| \tag{11}$$

and

$$|\tilde{f}_W(\boldsymbol{t}) - f(\boldsymbol{t})| \le 2\gamma. \tag{12}$$

From Lemma 4.1 and the condition on length of $\boldsymbol{t}$, we get $f(\boldsymbol{t}) \ge \rho(\boldsymbol{t}) \ge 4\gamma$. From the triangle inequality, we get

$$\left\|\frac{\nabla \tilde{f}_W(\boldsymbol{t})}{2\pi \tilde{f}_W(\boldsymbol{t})} - \frac{\nabla f(\boldsymbol{t})}{2\pi f(\boldsymbol{t})}\right\| = \left\|\frac{\nabla \tilde{f}_W(\boldsymbol{t}) - \nabla f(\boldsymbol{t})}{2\pi f(\boldsymbol{t})}\frac{f(\boldsymbol{t})}{\tilde{f}_W(\boldsymbol{t})} + \frac{\nabla f(\boldsymbol{t})}{2\pi f(\boldsymbol{t})}\left(\frac{f(\boldsymbol{t})}{\tilde{f}_W(\boldsymbol{t}-1)}\right)\right\| \tag{13}$$
$$\leq \left\|\frac{\nabla \tilde{f}_W(\boldsymbol{t}) - \nabla f(\boldsymbol{t})}{2\pi f(\boldsymbol{t})}\right\| \frac{f(\boldsymbol{t})}{\tilde{f}_W(\boldsymbol{t})} + \left\|\frac{\nabla f(\boldsymbol{t})}{2\pi f(\boldsymbol{t})}\right\| \left|\frac{f(\boldsymbol{t})}{\tilde{f}_W(\boldsymbol{t})} - 1\right|.$$

For the first term in (13), by using (11) and (12), we get

$$\left\|\frac{\nabla \tilde{f}_W(\boldsymbol{t}) - \nabla f(\boldsymbol{t})}{2\pi f(\boldsymbol{t})}\right\| \frac{f(\boldsymbol{t})}{\tilde{f}_W(\boldsymbol{t})} \leq \frac{\pi\gamma\|\boldsymbol{t}\|}{2\pi f(\boldsymbol{t})}\frac{f(\boldsymbol{t})}{f(\boldsymbol{t})-2\gamma} = \frac{\gamma\|\boldsymbol{t}\|}{2(f(\boldsymbol{t})-2\gamma)}. \tag{14}$$

For the second term in (13), by using Theorem 4.7 and (12), we get

$$\left\|\frac{\nabla f(\boldsymbol{t})}{2\pi f(\boldsymbol{t})}\right\| \left|\frac{f(\boldsymbol{t})}{\tilde{f}_W(\boldsymbol{t})} - 1\right| \leq \frac{5}{4}\|\boldsymbol{t}\|\left(\frac{f(\boldsymbol{t})}{f(\boldsymbol{t})-2\gamma} - 1\right) = \frac{10\gamma}{4(f(\boldsymbol{t})-2\gamma)}\|\boldsymbol{t}\|. \tag{15}$$

From (13), (14), and (15), we get

$$\left\|\frac{\nabla \tilde{f}_W(\boldsymbol{t})}{2\pi \tilde{f}_W(\boldsymbol{t})} - \frac{\nabla f(\boldsymbol{t})}{2\pi f(\boldsymbol{t})}\right\| \leq \frac{3\gamma\cdot\|\boldsymbol{t}\|}{f(\boldsymbol{t})-2\gamma} \leq \frac{6\gamma}{\rho(\boldsymbol{t})}\|\boldsymbol{t}\|. \qquad \square$$

LEMMA 4.13. *Let $\mathcal{L} \subset \mathbb{R}^n$ be a lattice with $\rho(\mathcal{L}) = 1+\varepsilon$ for $\varepsilon \in (0, 1/400)$. Let $W = (\boldsymbol{w}_1, \ldots, \boldsymbol{w}_N)$ be sampled independently from $D_{\mathcal{L}^*}$, with $N \geq n^8/\sqrt{\varepsilon}$. Suppose that for some $\boldsymbol{t} \in \mathbb{R}^n$ such that $\|\boldsymbol{t}\| \leq \varepsilon^{1/8}/(1000n)$, we can compute $\tilde{f}_W(\boldsymbol{t})$ and $\nabla\tilde{f}_W(\boldsymbol{t})$ which satisfy the following:*

1. $|\tilde{f}_W(\boldsymbol{t}) - f_W(\boldsymbol{t})| \leq \frac{\varepsilon^{1/4}}{100}$,
2. $\|\nabla\tilde{f}_W(\boldsymbol{t}) - \nabla f_W(\boldsymbol{t})\| \leq \frac{\varepsilon^{1/4}}{100}\|\boldsymbol{t}\|$.

*Then, with probability $1-2^{-\Omega(n)}$, $\|\frac{\nabla\tilde{f}_W(\boldsymbol{t})}{2\pi\tilde{f}_W(\boldsymbol{t})} + \boldsymbol{t}\| \leq 3\varepsilon^{1/4}\|\boldsymbol{t}\|$ holds.*

*Proof.* By Lemma 4.6, with at least $1-2^{-\Omega(n)}$ probability, $\|\nabla f_W(\boldsymbol{t})\| \leq (2\pi + 4\varepsilon^{1/4})\|\boldsymbol{t}\|$ and $|f_W(\boldsymbol{t})| > 1 - \frac{\varepsilon^{1/4}}{100}$. By the triangle inequality and both of the assumptions, with probability at least $1-2^{-\Omega(n)}$, we have

$$\|\nabla\tilde{f}_W(\boldsymbol{t})\| \leq (2\pi + 5\varepsilon^{1/4})\|\boldsymbol{t}\| \text{ and } |\tilde{f}_W(\boldsymbol{t})| > 1 - \frac{\varepsilon^{1/4}}{50}. \tag{16}$$

It implies that

$$\begin{aligned}\left\|\frac{\nabla\tilde{f}_W(\boldsymbol{t})}{2\pi\tilde{f}_W(\boldsymbol{t})} - \frac{\nabla f_W(\boldsymbol{t})}{2\pi f_W(\boldsymbol{t})}\right\| &\leq \left\|\frac{\nabla\tilde{f}_W(\boldsymbol{t}) - \nabla f_W(\boldsymbol{t})}{2\pi\tilde{f}_W(\boldsymbol{t})}\right\| + \frac{\|\nabla f_W(\boldsymbol{t})\|}{2\pi}\left|\frac{1}{\tilde{f}_W(\boldsymbol{t})} - \frac{1}{f_W(\boldsymbol{t})}\right| \\ &\leq \frac{\varepsilon^{1/4}\|\boldsymbol{t}\|}{2\pi(1-\varepsilon^{1/4}/50)} + \frac{(2\pi+5\varepsilon^{1/4})\|\boldsymbol{t}\|}{2\pi} \\ &\quad \cdot \frac{\varepsilon^{1/4}/100}{(1-\varepsilon^{1/4}/100)(1-\varepsilon^{1/4}/50)} \\ &\leq 2\varepsilon^{1/4}\|\boldsymbol{t}\|.\end{aligned}$$

By Lemma 4.2, we know that, with at least $1-2^{-\Omega(n)}$ probability, we have $\|\frac{\nabla f_W(\boldsymbol{t})}{2\pi f_W(\boldsymbol{t})} + \boldsymbol{t}\| \leq \varepsilon^{1/4}\|\boldsymbol{t}\|$. Hence, by the triangle inequality and union bound, we have $\|\frac{\nabla\tilde{f}_W(\boldsymbol{t})}{2\pi\tilde{f}_W(\boldsymbol{t})} + \boldsymbol{t}\| \leq 3\varepsilon^{1/4}\|\boldsymbol{t}\|$ with probability greater than $1-2^{-\Omega(n)}$. $\square$

THEOREM 4.14. *Let $\mathcal{L} \subset \mathbb{R}^n$ be a lattice with $\rho(\mathcal{L}) = 1 + \varepsilon$, with $\varepsilon \in (0, 1/400)$. Let $s_\varepsilon = (\frac{1}{\pi} \ln \frac{2(1+\varepsilon)}{\varepsilon})^{0.5}$, $\delta_{\max} = 0.5 - \frac{2}{\pi s_\varepsilon^2}$, $\delta(\boldsymbol{t}) = \max\{1/8, \|\boldsymbol{t}\|/s_\varepsilon\}$, and $W = (\boldsymbol{w}_1, \ldots, \boldsymbol{w}_N)$, and let be sampled independently from $\mathcal{D}_{\mathcal{L}^*}$, where $N \geq n^8 \ln(1/\varepsilon)/\sqrt{\varepsilon}$. Suppose that for some $\boldsymbol{t} \in \mathbb{R}^n$ such that $\|\boldsymbol{t}\| \leq \delta_{\max} s_\varepsilon$, we can compute $\tilde{f}_W(\boldsymbol{t})$ and $\nabla \tilde{f}_W(\boldsymbol{t})$ which satisfy the following:*

1. $|\tilde{f}_W(\boldsymbol{t}) - f_W(\boldsymbol{t})| \leq \frac{\varepsilon^{1/4}}{100}$,
2. $\|\nabla \tilde{f}_W(\boldsymbol{t}) - \nabla f_W(\boldsymbol{t})\| \leq \frac{\|\boldsymbol{t}\| \cdot \varepsilon^{1/4}}{100}$.

*Then, with probability at least $1 - 2^{-\Omega(n)}$, $\|\frac{\nabla \tilde{f}_W(\boldsymbol{t})}{2\pi \tilde{f}_W(\boldsymbol{t})} + \boldsymbol{t}\| \leq \varepsilon^{(1-2\delta(\boldsymbol{t}))/4} \|\boldsymbol{t}\|$.*

*Proof.* Lemma 4.13 shows that the theorem is satisfied for all $\boldsymbol{t}$ with $\|\boldsymbol{t}\| \leq \varepsilon^{1/8}/(1000n)$. So we consider the case when $\varepsilon^{1/8}/(1000n) < \|\boldsymbol{t}\| \leq \delta_{\max} s_\varepsilon$. By Lemma 4.1, for such $\boldsymbol{t}$,

$$f(\boldsymbol{t}) \geq \rho(\boldsymbol{t}) \geq e^{-\pi \delta_{\max}^2 s_\varepsilon^2} > \varepsilon^{\delta_{\max}^2}/2 \geq \varepsilon^{1/4}/2.$$

Also we know the following:

- By Lemma 4.3, $\|\nabla f_W(\boldsymbol{t}) - \nabla f(\boldsymbol{t})\| \leq \varepsilon^{1/4}\|\boldsymbol{t}\|/100$ holds with probability $\geq 1 - 2^{-\Omega(\varepsilon^{1/2}N/100^2)} = 1 - 2^{-\Omega(n)}$.
- By Lemma 4.4, $\|f(\boldsymbol{t}) - f(\boldsymbol{t})\| \leq \varepsilon^{1/4}/100$ holds with probability $\geq 1 - 2^{-\Omega(\varepsilon^{1/2}N/100^2)} = 1 - 2^{-\Omega(n)}$.

Therefore, by using Theorem 4.12 with $\gamma = \frac{\varepsilon^{1/4}}{50\pi}$, we have that with probability $\geq 1 - 2^{-\Omega(n)}$,

$$\begin{aligned}
\left\| \frac{\nabla \tilde{f}_W(\boldsymbol{t})}{2\pi \tilde{f}_W(\boldsymbol{t})} + \boldsymbol{t} \right\| &\leq \frac{6\gamma}{\rho(\boldsymbol{t})} \|\boldsymbol{t}\| + \left\| \frac{\nabla f(\boldsymbol{t})}{2\pi f(\boldsymbol{t})} + \boldsymbol{t} \right\| \\
&\leq \frac{\varepsilon^{1/4}}{25} \cdot e^{\pi \|\boldsymbol{t}\|^2} \|\boldsymbol{t}\| + 12(\varepsilon/2)^{1-2\delta(\boldsymbol{t})} \|\boldsymbol{t}\| \qquad \text{(by Theorem 4.7)} \\
&\leq \frac{3\varepsilon^{1/4}}{25} \left( \frac{2(1+\varepsilon)}{\varepsilon} \right)^{\delta(\boldsymbol{t})^2} \|\boldsymbol{t}\| + 12(\varepsilon/2)^{1-2\delta(\boldsymbol{t})} \|\boldsymbol{t}\| \\
&\leq \frac{\varepsilon^{0.25-\delta(\boldsymbol{t})^2}}{6} \cdot \|\boldsymbol{t}\| + \frac{4}{5} (\varepsilon)^{(1-2\delta(\boldsymbol{t}))/4} \|\boldsymbol{t}\| \\
&\leq \varepsilon^{(1-2\delta(\boldsymbol{t}))/4} \|\boldsymbol{t}\|.
\end{aligned}$$

∎

THEOREM 4.15. *For any integer $n > 0$, $\varepsilon \in (e^{-n^2}, 1/400)$ and lattice $\mathcal{L} \subset \mathcal{R}^n$, let $\phi(\mathcal{L}) = \frac{\sqrt{\ln(1/\varepsilon)/\pi - o(1)}}{2\eta_\varepsilon(\mathcal{L}^*)}$ and $N = \frac{n^8 \ln(1/\varepsilon)}{\sqrt{\varepsilon}}$. Given $N$ vectors sampled from $\mathcal{D}_{\mathcal{L}^*, \eta_\varepsilon(\mathcal{L}^*)}$ stored in QRAM, there exists an algorithm that solves $\frac{\phi(\mathcal{L})}{\lambda_1(\mathcal{L})}$-BDD with probability greater than $1 - 2^{-\Omega(n)}$ using $\sqrt{N} \cdot poly(n)$ arithmetic operations and requires $\mathcal{O}(poly(n) + \log N)$ classical space, $poly(n)$ qubits, and QRAM of size $N \cdot poly(n)$.*

*Proof.* Let $s_\varepsilon = (\frac{1}{\pi} \log(\frac{2(1+\varepsilon)}{\varepsilon}))^{1/2}$ and $\delta_{max} = \frac{1}{2} - \frac{2}{\pi s_\varepsilon^2}$. Let $W = (\boldsymbol{w}_1, \ldots, \boldsymbol{w}_N)$ be the given set of vectors sampled from $\mathcal{D}_{\mathcal{L}^*, \eta_\varepsilon(\mathcal{L}^*)}$.

The algorithm takes target $\boldsymbol{t} \in \mathcal{R}^n$ and set of vectors $W$. It then iteratively updates $\boldsymbol{t} \leftarrow \boldsymbol{t} + \frac{\nabla \tilde{f}_W(\boldsymbol{t})}{2\pi \tilde{f}_W(\boldsymbol{t})}$ for $1 + \lceil 8 \log(\sqrt{n} s_\varepsilon)/\log(1/\varepsilon) \rceil$ times. It then scans the first $\sqrt{N}$ vectors from $W$ and takes the first $n$ linearly independent vectors of length bounded by $\sqrt{n} \cdot \eta_\varepsilon(\mathcal{L}^*)$ as set $V^* = (\boldsymbol{v}_1^*, \ldots, \boldsymbol{v}_n^*) \subset W$; if no such set exists, then abort. Compute $V = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n)$ such that $\boldsymbol{v}_i \cdot \boldsymbol{v}_j^* = \delta_{i,j}$, and return $\sum c_i \boldsymbol{v}_i$, where $c_i = \lfloor \boldsymbol{v}_i^* \cdot \boldsymbol{t} \rceil$.

By scaling the lattice appropriately, we can assume that $\rho(\mathcal{L}) = 1 + \varepsilon$, so that $\eta_\varepsilon(\mathcal{L}^*) = 1$. Let $\boldsymbol{t}' = \boldsymbol{t} - \boldsymbol{w}$ such that $\|\boldsymbol{t}'\| \leq \delta_{max} s_\varepsilon$ for some $\boldsymbol{w} \in \mathcal{L}$. There exists

such a vector $\boldsymbol{t}'$ because of the promise of the $\phi(\mathcal{L})/\lambda_1(\mathcal{L})$-BDD and the fact that $\phi(\mathcal{L}) \leq \delta_{max} s_\varepsilon$. From Lemma 2.9, we get that with probability $1-N\cdot e^{-2n^2} \geq 1-2^{\Omega(n)}$, $\forall i \in [N], \|\boldsymbol{w}_i\| \leq 2n+1$. Hence, by using Corollary 4.11, we can estimate $\tilde{f}_W(\boldsymbol{t})$ and $\nabla \tilde{f}_W(\boldsymbol{t})$ which satisfy

$$|\tilde{f}_W(\boldsymbol{t}) - f_W(\boldsymbol{t})| \leq \frac{\varepsilon^{1/4}}{100} \text{ and } \|\nabla \tilde{f}_W(\boldsymbol{t}) - \nabla f_W(\boldsymbol{t})\| \leq \frac{\varepsilon^{1/4}}{100}\|\boldsymbol{t}\|$$

with probability $1-2^{-\Omega(n)}$. The algorithm makes $\mathcal{O}(\sqrt{N})$ arithmetic operations. Corollary 4.11 also says that the distribution of $\tilde{f}_W(\boldsymbol{t})$ and $\nabla \tilde{f}_W(\boldsymbol{t})$ is periodic over the lattice. Therefore, by using Theorem 4.14 we can say that by every update of $\boldsymbol{t}$ by $\boldsymbol{t} \leftarrow \boldsymbol{t} + \frac{\nabla \tilde{f}_W(\boldsymbol{t})}{2\pi \tilde{f}_W(\boldsymbol{t})}$, the output vector is of the form $\boldsymbol{w} + \boldsymbol{t}^*$, where $\|\boldsymbol{t}^*\|$ shrinks by a factor of at least $\varepsilon^{(1-2(1/4))/4} = \varepsilon^{1/8}$ with probability $1-2^{-\Omega(n)}$. Hence, by $1 + \lceil 8\log(\sqrt{n}s_\varepsilon)/\log(1/\varepsilon)\rceil$ updates, we get $\boldsymbol{t} = \boldsymbol{w} + \boldsymbol{t}^*$ such that $\|\boldsymbol{t}^*\| \leq 1/(2\sqrt{n})$. For the correctness of the proof, it is sufficient to show that $\boldsymbol{w} = \sum_i c_i \boldsymbol{v}_i$. Note that $\lfloor \langle \boldsymbol{v}_i^*, \boldsymbol{t} \rangle \rceil = \lfloor \langle \boldsymbol{v}_i^*, \boldsymbol{t}^* \rangle \rceil + \langle \boldsymbol{v}_i^*, \boldsymbol{w} \rangle$. By Cauchy–Schwarz, we get $\lfloor \langle \boldsymbol{v}_i^*, \boldsymbol{t}^* \rangle \rceil = 0$. Hence, we get the vector $\boldsymbol{w}$ as the output with probability greater than $1-2^{-\Omega(n)}$.

Now we will show that there exists a set $V^* \subset \{\boldsymbol{w}_1, \ldots, \boldsymbol{w}_{\boldsymbol{n^2}}\}$ containing $n$ linearly independent vectors of length at most $\sqrt{n}$ and an algorithm will abort with probability at most $2^{-\Omega(n^2)}$ over the given set $W$. Let $W' = (\boldsymbol{w}_1, \ldots, \boldsymbol{w}_{n^2})$. By Lemma 2.9, with at least $1-n^2\cdot e^{-n}$ probability all the vectors in set $W'$ have length at most $\sqrt{n}\cdot\eta_\varepsilon(\mathcal{L}^*)$. From Lemma 4.5, we know that

$$\|Hf_{W'}(\mathbf{0}) + 2\pi \mathbf{I}_n\| \leq \frac{4\pi\varepsilon}{1+\varepsilon}\left(\log\left(\frac{2(1+\varepsilon)}{\varepsilon}+1\right)+1\right) < 2\pi$$

with probability at least $1-2^{-\Omega(n^2)}$. Hence, we have that $Hf_{W'}(\mathbf{0})$ is invertible and $Hf_{W'}(\mathbf{0}) = \frac{-4\pi^2}{m}\sum_{i=1}^m \boldsymbol{w}_i \boldsymbol{w}_i^T$ (from (10)). It implies that $W'$ spans $\mathbb{R}^n$ and completes the proof. ∎

**5. Improved algorithms for BDD.** We obtain a BDD oracle with decoding distance $\alpha$ by using the same reduction as above but making each call cheaper. This is achieved by building a sampler that directly samples at the smoothing parameter, hence avoiding the $\sqrt{2}$ factor, allowing us to take a bigger $\varepsilon$. In [3], it was shown how to construct a dense lattice $\mathcal{L}'$ whose smoothing parameter $\eta(\mathcal{L}')$ is $\sqrt{2}$ times smaller than the original lattice and that contains all lattice points of the original lattice. Suppose that we first use such a dense lattice to construct a corresponding discrete Gaussian sampler with standard deviation equal to $s = \sqrt{2}\eta(\mathcal{L}')$. We then do the rejection sampling on the condition that the output is in the original lattice $\mathcal{L}$. We thus have constructed a discrete Gaussian sampler of $\mathcal{L}$ whose standard deviation is $\sqrt{2}\eta(\mathcal{L}') = \eta(\mathcal{L})$. Nevertheless, $|\mathcal{L}'/\mathcal{L}|$ will be at least $2^{0.5n}$, which implies that this procedure needs at least $2^{0.5n}$ input vectors to produce an output vector. The complexity of our BDD algorithms heavily depends on the quantity $\beta(\mathcal{L})$ which is related to the kissing number of the lattice (see section 2). For this reason, we first provide complexity bounds that depend on $\beta(\mathcal{L})$ and then obtain complexity bounds in the worst case ($\beta(\mathcal{L}) \leqslant 2^{0.402}$) as corollaries. We first show how to efficiently sample a discrete Gaussian at the smoothing parameter.

LEMMA 5.1. *There is a probabilistic algorithm that, given a lattice $\mathcal{L} \subset \mathbb{R}^n$, $m \in \mathbb{Z}_+$, and $s \geq \eta_{1/3}(\mathcal{L})$ as input, outputs $m$ samples from a distribution $(m\cdot 2^{-\Omega(n^2)})$-close to $D_{\mathcal{L},s}$ in expected time $m\cdot 2^{n/2+o(n)}$ and space $(m+2^{n/2})\cdot 2^{o(n)}$. Furthermore, all samples have poly$(n)$ bit-size.*

*Proof.* Let $a = \frac{n}{2} + 4$. We repeat the following until we output $m$ vectors. We use the algorithm in Lemma 2.22 to obtain a lattice $\mathcal{L}' \supset \mathcal{L}$ of index $2^a$. We then run the algorithm from Theorem 2.13 with input $(\mathcal{L}', s)$ to obtain a list of vectors from $\mathcal{L}'$. We output the vectors in this list that belong to $\mathcal{L}$. The correctness of the algorithm, assuming it outputs anything, is clear as long as the samples obtained from Theorem 2.13 are (sufficiently) independent, which we will prove below. By Theorem 2.13, we obtain, in time and space $2^{(n/2)+o(n)}$, $M \leqslant 2^{n/2}$ vectors that are $2^{-\Omega(n^2)}$-close to $M$ vectors independently sampled from $D_{\mathcal{L}',s}$. The theorem guarantees that $M = 2^{n/2}$ if $s \geqslant \sqrt{2}\eta_{1/2}(\mathcal{L}')$. Also, by Lemma 2.22, with probability at least $1/2$, we have $s \geq \eta_{1/3}(\mathcal{L}) \geq \sqrt{2}\eta_{1/2}(\mathcal{L}')$. Note that when $s < \sqrt{2}\eta_{1/2}(\mathcal{L}')$, the samples obtained from Theorem 2.13 are still $2^{-\Omega(n^2)}$-close to $M$ vectors independently sampled from $D_{\mathcal{L}',s}$ but $M$ could be much lower than $2^{n/2}$ or even 0. On the other hand, if $s \geqslant \sqrt{2}\eta_{1/2}(\mathcal{L}')$, then $M = 2^{n/2}$. Assume that $s \geqslant \sqrt{2}\eta_{1/2}(\mathcal{L}')$, which happens with probability at least $1/2$. From these $M = 2^{n/2}$ vectors, we will reject the vectors which are not in lattice $\mathcal{L}$. It is easy to see that the probability that a vector sampled from the distribution $D_{\mathcal{L}',s}$ is in $\mathcal{L}$ is at least $\rho_s(\mathcal{L})/\rho_s(\mathcal{L}') \geq \frac{1}{2^a}$ using Lemma 2.7. Thus, the probability that we obtain at least one vector from $\mathcal{L}$ (which is distributed as $D_{\mathcal{L},s}$) is at least

$$\frac{1}{2}\left(1-(1-1/2^a)^{2^{n/2}}\right) = \frac{1}{2}\left(1-(1-1/2^{n/2+4})^{2^{n/2}}\right) \geq \frac{1}{2}\cdot\left(1-e^{-2^{n/2}/2^{n/2+4}}\right)$$
$$= \frac{1}{2}(1-e^{-1/16}).$$

It implies that after rejection of vectors, with constant probability we will get at least one vector from $D_{\mathcal{L},s}$. Thus, the expected number of times we need to repeat the algorithm is $O(m)$ until we obtain vectors $\boldsymbol{y_1}, \ldots, \boldsymbol{y_m}$ whose distribution is statistically close to being independently distributed from $D_{\mathcal{L},s}$. The time and space complexity is clear from the algorithm. We can ensure that all samples have poly$(n)$ bit-size by first generating more samples (say twice the amount) and throwing away all samples of norm larger than $\exp(\Omega(n^2))$. Since the vectors are sampled from a Gaussian with width at most[7] $\exp(O(n))$, the error induced by throwing away the tail of the distribution is smaller than $2^{-\Omega(n^2)}$. ∎

**5.1. Reduction from BDD to DGS.** In [27], the authors gave an algorithm for BDDP which requires a sampler from the discrete Gaussian distribution *exactly at the smoothing parameter* $\eta_\varepsilon(\mathcal{L})$ which is generally not known. As BDDP allows preprocessing on the input lattice for arbitrary time, they are able to assume that preprocessing advice strings contains vectors sampled exactly at smoothing parameter. In this section, we present a modification of their algorithm, which gives a reduction from BDD to DGS. In this paper, our goal is to solve BDD, so we will be very precise with the runtime and not assume any preprocessing.

THEOREM 5.2. *For any $\alpha > 0$, any $e^{-n^2} \leqslant \varepsilon \leqslant \min(e^{-n^\alpha}, 1/200)$. There exists an algorithm that, on input $\mathcal{L}$ and with constant probability, constructs a classical and a quantum (with QRAM) $\frac{\phi(\mathcal{L})}{\lambda_1(\mathcal{L})}$-BDD oracle for $\mathcal{L}$ by doing poly$(n)$ calls to a 0.5-$\mathrm{hDGS}^m_{\eta_\varepsilon}$ sampler on the lattice $\mathcal{L}^*$ and requires storage space $m \cdot$ poly$(n)$, where $m = O(\frac{n\ln(1/\varepsilon)}{\sqrt{\varepsilon}})$ and $\phi(\mathcal{L}) \equiv \frac{\sqrt{\ln(1/\varepsilon)/\pi - o(1)}}{2\eta_\varepsilon(\mathcal{L}^*)}$. Each call to the classical oracle takes time $m\cdot$poly$(n)$ and space $O($poly$(n)+\ln m)$, excluding the storage space of the preprocessing.*

[7] Here we are using our assumption that the basis vectors have size at most $2^{o(n)}$.

*Every call to the quantum oracle with QRAM takes time $\sqrt{m}\cdot poly(n)$, classical space $O(poly(n)+\ln m)$, and $poly(n)$ qubits and requires a QRAM of size $m\cdot poly(n)$ that contains the preprocessed data.*[8]

The proof follows from Theorems 2.14 and 4.15. For completeness, a detailed proof is given in Appendix A.

**5.2. BDD when $\varepsilon$ is small.** In order to go further, we will make heavy use of Theorem 5.2 and Lemma 2.16 to relate the smoothing parameter to other parameters of the lattice. A small difficulty when applying Lemma 2.16 is the case distinction on $\varepsilon$. We will start by using inequality (5), which will require us to take very small values of $\varepsilon$ when sampling the discrete Gaussian.

LEMMA 5.3. *For any sufficiently large $n$, any lattice $\mathcal{L}\subset\mathbb{R}^n$, and any $A$ such that $\frac{1}{2\ln 2}-b+o(1)\leqslant A\leqslant 1$, there exists a randomized algorithm that creates a classical and a quantum (with QRAM) $\alpha$-BDD oracle in time $2^{(A+1)n/2+o(n)}$ and space $2^{0.5n+o(n)}$, where $\alpha=\frac{1}{2}\sqrt{\frac{A}{A+b}}$ and $b=\log_2\beta(\mathcal{L})$. Every call to the classical oracle takes time $2^{An/2+o(n)}$ and space poly$(n)$, excluding the space of the preprocessed data. Every call to the quantum oracle with QRAM takes time $2^{An/4+o(n)}$, classical space poly$(n)$, and poly$(n)$ qubits and requires QRAM of size $2^{An/2+o(n)}$ that contains the preprocessed data.*

*Proof.* Let $\varepsilon=2^{-An}$, $A\leqslant 1$ to be fixed later. We know that $\eta_\varepsilon(\mathcal{L}^*)>\eta_{1/3}(\mathcal{L}^*)$ for any sufficiently large $n$ ($n>\frac{1}{A}\log_2 3$) by the monotonicity of the smoothing parameter function. Hence, the $\mathsf{DGS}^m_{\eta_{1/3}}$ sampler from Lemma 5.1 can be used as a $\mathsf{DGS}^m_{\eta_\varepsilon}$ sampler for any $m\in\mathbb{N}$. By Theorem 5.2, we can construct a $\alpha$-BDD such that each call takes time $m\cdot\mathrm{poly}(n)=2^{An/2+o(n)}$ and space poly$(n)$, where $\alpha=\phi(\mathcal{L})/\lambda_1(\mathcal{L})=\frac{\sqrt{\ln(1/\varepsilon)/\pi-o(1)}}{2\eta_\varepsilon(\mathcal{L}^*)\lambda_1(\mathcal{L})}$ and $m=O(\frac{n\log(1/\varepsilon)}{\sqrt{\varepsilon}})=2^{An/2+o(n)}$. The preprocessing consists of $\mathrm{poly}(n)$ calls to the $\mathsf{DGS}^m_{\eta_\varepsilon}$ sampler described above and requires space $m\cdot\mathrm{poly}(n)$. Hence, the total complexity is $\mathrm{poly}(n)\cdot m\cdot 2^{n/2+o(n)}=2^{(A+1)n/2+o(n)}$ in time and $2^{An/2+o(n)}\leqslant 2^{n/2+o(n)}$ in space. By using Lemma 2.16, inequality (5), only valid when $\varepsilon\leq(e/\beta(\mathcal{L})^2+o(1))^{-\frac{n}{2}}$, we have that

$$\lambda_1(\mathcal{L})\eta_\varepsilon(\mathcal{L}^*)<\sqrt{\frac{\ln(1/\varepsilon)+n\ln\beta(\mathcal{L})+o(n)}{\pi}}.$$

Hence, we can guarantee that

$$\alpha=\frac{\sqrt{\ln(1/\varepsilon)/\pi-o(1)}}{2\eta_\varepsilon(\mathcal{L}^*)\lambda_1}>\frac{\sqrt{\ln(1/\varepsilon)/\pi-o(1)}}{2\sqrt{\frac{\ln(1/\varepsilon)+n\ln\beta(\mathcal{L})+o(n)}{\pi}}}=\frac{1}{2}\sqrt{\frac{\ln(1/\varepsilon)+o(1)}{\ln(1/\varepsilon)+n\ln\beta(\mathcal{L})}}$$
$$=\frac{1}{2}\sqrt{\frac{A}{A+b}}+o(1),$$

where $b=\log_2\beta(\mathcal{L})$. Furthermore, as noted above, this inequality is only valid when $\varepsilon\leq(e/\beta^2+o(1))^{-\frac{n}{2}}$, that is, $A\geqslant\frac{1}{2\ln 2}-b+o(1)$. Finally, note that since $b\leqslant 0.402$, we must have $A\geqslant 0.32$ and the inequality holds as soon as $n\geqslant 5\geqslant\frac{1}{A}\log_2 3$. Finally, note that Theorem 5.2 requires $\varepsilon\leqslant 1/200$, which holds as soon as $n\geqslant 17\geqslant\frac{1}{A}\ln 200$. The quantum algorithm is exactly the same but using the quantum oracle of Theorem 5.2, which then has running time $\sqrt{m}\cdot\mathrm{poly}(n)=2^{An/4+o(n)}$ and requires a QRAM of size $m\cdot\mathrm{poly}(n)=2^{An/2+o(n)}$. ∎

[8]Here we are assuming that the Gaussian samples appear in the streaming fashion.

We can reformulate the previous lemma by expressing the complexity in terms of $\alpha$ instead of some arbitrary constant $A$.

COROLLARY 5.4. *For any $n \geqslant 5$, any lattice $\mathcal{L} \subset \mathbb{R}^n$, and any $\alpha$ such that $\frac{1}{2}\sqrt{1-2b\ln 2}+o(1) \leqslant \alpha < \frac{1}{2}\sqrt{\frac{1}{1+b}}$, there exists a randomized algorithm that creates a classical and a quantum (with QRAM) $\alpha$-BDD oracle in time $2^{(A+1)n/2+o(n)}$ and space $2^{0.5n+o(n)}$. Every call to the classical oracle takes time $2^{An/2+o(n)}$ and space poly$(n)$, excluding the space of the preprocessed data; and every call to the quantum oracle with QRAM takes time $2^{An/4+o(n)}$, classical space poly$(n)$, and poly$(n)$ qubits and requires a QRAM of size $2^{An/2+o(n)}$ that contains the preprocessed data, where $A = \frac{4b\alpha^2}{1-4\alpha^2}$ and $b = \log_2 \beta(\mathcal{L})$.*

*Proof.* Apply Lemma 5.3 for some $A$ to be fixed later. Observe that $\alpha = \frac{1}{2}\sqrt{\frac{A}{A+b}}$ so $A = \frac{4b\alpha^2}{1-4\alpha^2}$. Now the constraints $\frac{1}{2\ln 2} - b + o(1) \leqslant A \leqslant 1$ become

$$\begin{aligned}\frac{1}{2\ln 2} - b + o(1) \leqslant \frac{4b\alpha^2}{1-4\alpha^2} &\Leftrightarrow \left(\frac{1}{2\ln 2} - b + o(1)\right)(1-4\alpha^2) \leqslant 4b\alpha^2 \\ &\Leftrightarrow \frac{1}{4\ln 2} - \frac{b}{2} + o(1) \leqslant \frac{\alpha^2}{\ln 2} \\ &\Leftrightarrow \frac{1}{2}\sqrt{1-2b\ln 2} + o(1) \leqslant \alpha\end{aligned}$$

and

$$\frac{4b\alpha^2}{1-4\alpha^2} \leqslant 1 \Leftrightarrow 4(1+b)\alpha^2 \leqslant 1 \Leftrightarrow \alpha \leqslant \frac{1}{2}\sqrt{\frac{1}{1+b}}. \qquad \square$$

**5.3. BDD when $\varepsilon$ is large.** The inequality (5) in Lemma 2.16 tells us that if we take an extremely small $\varepsilon$ to compute the BDD oracle, we can find a BDD oracle with $\alpha(\mathcal{L})$ almost $1/2$. However, the time complexity for each call of the oracle will be very costly. On the other hand, if we use the inequality (4) in Lemma 2.16 with a larger $\varepsilon$, each call of the oracle will take much less time, but the constraint on the decoding coefficient $\alpha$ will be different. It is therefore important to study this second regime as well. Note that inequality (4) actually applies to all $\varepsilon \in (0,1)$ but is mostly useful when $\varepsilon$ is large.

LEMMA 5.5. *For any sufficiently large $n$, any lattice $\mathcal{L} \subset \mathbb{R}^n$, and any $\frac{1}{n}\log_2 3 \leqslant A \leqslant 1$, there exists a randomized algorithm that creates a classical and a quantum (with QRAM) $\alpha$-BDD oracle in time $2^{(A+1)n/2+o(n)}$ and space $2^{0.5n+o(n)}$, where $\alpha = \frac{2^{-A}\sqrt{A}\sqrt{2e\ln 2}}{2\beta(\mathcal{L})} - o(1)$. Every call to this oracle takes time $2^{An/2+o(n)}$ and space poly$(n)$, excluding the space of the preprocessed data. Every call to the quantum oracle with QRAM takes time $2^{An/4+o(n)}$, classical space poly$(n)$, and poly$(n)$ qubits and requires a QRAM of size $2^{An/2+o(n)}$ that contains the preprocessed data.*

*Proof.* Let $\varepsilon = 2^{-An}$, $A \leqslant 1$ to be fixed later. We know that $\eta_\varepsilon(\mathcal{L}^*) > \eta_{1/3}(\mathcal{L}^*)$ for any sufficiently large $n$ ($n > \frac{1}{A}\log_2 3$) by the monotonicity of the smoothing parameter function. Hence, the $\mathsf{DGS}^m_{\eta_{1/3}}$ sampler from Lemma 5.1 can be used as a $\mathsf{DGS}^m_{\eta_\varepsilon}$ sampler for any $m \in \mathbb{N}$. By Theorem 5.2, we can construct an $\alpha$-BDD such that each call takes time $m \cdot \mathrm{poly}(n) = 2^{An/2+o(n)}$ and space poly$(n)$, where $\alpha = \phi(\mathcal{L})/\lambda_1(\mathcal{L}) = \frac{\sqrt{\ln(1/\varepsilon)/\pi - o(1)}}{2\eta_\varepsilon(\mathcal{L}^*)\lambda_1(\mathcal{L})}$ and $m = O(\frac{n\log(1/\varepsilon)}{\sqrt{\varepsilon}}) = 2^{An/2+o(n)}$. The preprocessing consists of poly$(n)$ calls to the $\mathsf{DGS}^m_{\eta_\varepsilon}$ sampler described above and requires space $m \cdot \mathrm{poly}(n)$.

Hence, the total complexity is $\mathrm{poly}(n)\cdot m\cdot 2^{n/2+o(n)} = 2^{(A+1)n/2+o(n)}$ in time and $2^{An/2+o(n)} \leqslant 2^{n/2+o(n)}$ in space. By using inequality (4) in Lemma 2.16, we have that

$$\lambda_1(\mathcal{L})\eta_\varepsilon(\mathcal{L}^*) < \sqrt{\frac{\beta(\mathcal{L})^2 n}{2\pi e}}\cdot \varepsilon^{-1/n}(1+o(1)).$$

Hence, we can guarantee that

$$\begin{aligned}\alpha &= \frac{\sqrt{\ln(1/\varepsilon)/\pi - o(1)}}{2\eta_\varepsilon(\mathcal{L}^*)\lambda_1(\mathcal{L})} > \frac{1}{2}\sqrt{\frac{2e\ln\frac{1}{\varepsilon} - o(1)}{n}}\cdot \beta(\mathcal{L})^{-1}\varepsilon^{\frac{1}{n}}\cdot(1-o(1))\\ &= \frac{2^{-A}\sqrt{A}\cdot\sqrt{2e\ln 2}}{2\beta(\mathcal{L})} - o(1).\end{aligned}$$

The quantum algorithm is exactly the same but using the quantum oracle of Theorem 5.2, which then has running time $\sqrt{m}\cdot\mathrm{poly}(n) = 2^{An/4+o(n)}$ and requires a QRAM of size $m\cdot\mathrm{poly}(n) = 2^{An/2+o(n)}$. ∎

COROLLARY 5.6. *For any $n \geqslant 2$, any integer $m > 0$, any lattice $\mathcal{L}\subset\mathbb{R}^n$, and any $\frac{\sqrt{e\ln 3}}{\sqrt{2n}\beta(\mathcal{L})} \leqslant \alpha \leqslant \frac{1}{2\beta(\mathcal{L})}$, where $b = \log_2\beta(\mathcal{L})$, there exists a randomized algorithm that creates a classical and a quantum (with QRAM) $(\alpha+o(1))$-BDD oracle in time $2^{(A+1)n/2+o(n)}$ and space $2^{n/2+o(n)}$. Every call to the classical oracle takes time $2^{An/2+o(n)}$ and space poly$(n)$, excluding the space of the preprocessed data; and every call to the quantum oracle with QRAM takes time $2^{An/4+o(n)}$, classical space poly$(n)$, and poly$(n)$ qubits and requires a QRAM of size $2^{An/2+o(n)}$ that contains the preprocessed data, where*

$$A = -\frac{1}{2\ln 2}W\left(-\frac{4\alpha^2\beta(\mathcal{L})^2}{e}\right),$$

*where $W$ is the principal branch of the Lambert W function. Furthermore, the above expression of $A$ is a continuous and increasing function of $\beta(\mathcal{L})$.*

*Proof.* By Lemma 5.5, we can build an oracle for any $\frac{1}{n}\log_2 3 \leqslant A \leqslant 1$ such that the decoding radius is $\alpha = \frac{2^{-A}\sqrt{A}\sqrt{2e\ln 2}}{2\beta(\mathcal{L})} - o(1)$. Hence, we want to find $A$ such that

$$\frac{2^{-A}\sqrt{A}\sqrt{2e\ln 2}}{2\beta(\mathcal{L})} = \alpha \qquad \text{and} \qquad \frac{1}{n}\log_2 3 \leqslant A \leqslant 1.$$

Let $f: A\mapsto 2^{-A}\sqrt{A}$ so that the first condition is equivalent to

$$f(A) = \frac{2\alpha\beta(\mathcal{L})}{\sqrt{2e\ln 2}}. \tag{17}$$

Now assume that (17) holds and let $y = -2A\ln(2)$; then it is equivalent to

$$e^y y = -2\ln(2)\frac{2\alpha^2\beta(\mathcal{L})^2}{e\ln 2},$$

that is,

$$e^y y = -\frac{4\alpha^2\beta(\mathcal{L})^2}{e}. \tag{18}$$

This equation admits a solution if and only if

$$-\frac{4\alpha^2\beta(\mathcal{L})^2}{e} \geqslant -\frac{1}{e} \quad \Leftrightarrow \quad \alpha \leqslant \frac{1}{2\beta(\mathcal{L})}. \tag{19}$$

Assuming this is the case, (18) can admit up to two solutions. However, since the complexity increases with $A$, we want the solution that minimizes $A$, i.e., that maximizes $y$. The largest of the (up to) two solutions of (18) is always given by the principal branch $W$ of the Lambert W function

$$y = W\left(-\frac{4\alpha^2\beta(\mathcal{L})^2}{e}\right) \quad \text{that is} \quad A = -\frac{1}{2\ln 2} W\left(-\frac{4\alpha^2\beta(\mathcal{L})^2}{e}\right) \tag{20}$$

and always satisfies $y \geqslant -1$. In particular, we always have $A \leqslant \frac{1}{2\ln 2}$. Now check that $f$ is strictly increasing over $[0, \frac{1}{2\ln 2}]$. Hence, the condition $\frac{1}{n}\log_2 3 \leqslant A$ is equivalent to

$$\begin{aligned} & f\left(\frac{1}{n}\log_2 3\right) \leqslant f(A) \\ & \Leftrightarrow f\left(\frac{1}{n}\log_2 3\right)^2 \leqslant \left(\frac{2\alpha\beta(\mathcal{L})}{\sqrt{2e\ln 2}}\right)^2 \qquad \text{by (17)} \\ & \Leftrightarrow 2^{-\frac{2}{n}\log_2 3}\frac{1}{n}\log_2 3 \leqslant \frac{2\alpha^2\beta(\mathcal{L})^2}{e\ln 2} \\ & \Leftrightarrow \frac{e\ln 3}{2n\beta(\mathcal{L})^2} 9^{-\frac{1}{n}} \leqslant \alpha^2 \\ & \Leftarrow \frac{e\ln 3}{2n\beta(\mathcal{L})^2} \leqslant \alpha^2. \end{aligned} \tag{21}$$

In summary, we can always take $A$ as in (20), assuming (19) and (21) hold. ∎

**5.4. Putting everything together.** We have analyzed the construction of $\alpha$-BDDs in two regimes, based on Lemma 5.1. It is not a priori clear which construction is better, and in fact we will see that it depends in a nontrivial way on the relation between $\alpha$ and $\beta(\mathcal{L})$.

THEOREM 5.7. *For any sufficiently large $n$, any $m > 0$, any $\frac{\sqrt{e\ln 3}}{\sqrt{2n}\beta(\mathcal{L})} \leqslant \alpha < \frac{1}{2}\sqrt{\frac{1}{1+b}}$, and any lattice $\mathcal{L} \subset \mathbb{R}^n$, there exists a randomized algorithm that creates a classical and a quantum (with QRAM) $(\alpha + o(1))$-BDD oracle in time $2^{(A+1)n/2+o(n)}$ and space $2^{n/2}$. Every call to the classical oracle takes time $2^{An/2+o(n)}$ and space poly$(n)$, excluding the space of the preprocessed data; and every call to the quantum oracle with QRAM takes time $2^{An/4+o(n)}$, classical space poly$(n)$, and poly$(n)$ qubits and requires a QRAM of size $2^{An/2+o(n)}$ that contains the preprocessed data, where*

$$A = \begin{cases} -\dfrac{1}{2\ln 2} W\left(-\dfrac{4\alpha^2\beta(\mathcal{L})^2}{e}\right) & \text{when } b < \frac{1-4\alpha^2}{2\ln 2}, \\ \dfrac{4\alpha^2}{1-4\alpha^2} b & \text{when } b \geqslant \frac{1-4\alpha^2}{2\ln 2}, \end{cases}$$

*where $W$ is the principal branch of the Lambert W function and $b = \log_2 \beta(\mathcal{L})$. Furthermore, the above expression of $A$ is a continuous and increasing function of $b$.*

*Proof.* Let $\frac{\sqrt{e\ln 3}}{\sqrt{2n}\beta(\mathcal{L})} \leqslant \alpha < \frac{1}{2}\sqrt{\frac{1}{1+b}}$ and $b = \log_2 \beta(\mathcal{L})$. By Corollary 5.4, we can build an $\alpha$-BDD if $\frac{1}{2}\sqrt{1-2b\ln 2} \leqslant \alpha$, in which case the complexity will depend on $A = A_1(\alpha, b) := \frac{4b\alpha^2}{1-4\alpha^2}$. By Corollary 5.6, we can build an $\alpha$-BDD if $\alpha < \frac{1}{2}\sqrt{1-2b\ln 2}$, in which case the complexity will depend on $A = A_2(\alpha, b) := -\frac{1}{2\ln 2} W(-\frac{4\alpha^2\beta(\mathcal{L})^2}{e})$. In both cases, the BDD oracle can be created in time $2^{(A+1)n/2+o(n)}$ and space $2^{0.5n+o(n)}$, and each call takes time $2^{An/2+o(n)}$. Now observe that

$$\alpha < \frac{1}{2}\sqrt{1-2b\ln 2} \Leftrightarrow 4\alpha^2 < 1-2b\ln 2 \Leftrightarrow b < \frac{1-4\alpha^2}{2\ln 2}.$$

Let $b^* := \frac{1-4\alpha^2}{2\ln 2}$; then there are two cases:

- If $b \geqslant b^*$, then $\frac{1}{2}\sqrt{1-2b\ln 2} \leqslant \alpha$ so only Corollary 5.4 applies and we can build an $\alpha$-BDD. In this case, the complexity exponent is $A_1(\alpha, b) = \frac{4\alpha^2}{1-4\alpha^2} b$.
- If $b < b^*$, then $\alpha < \frac{1}{2}\sqrt{1-2b\ln 2}$ so Corollary 5.6 applies but Corollary 5.4 does not for this particular value of $\alpha$. However, we can apply Corollary 5.4 to build an $\alpha'$-BDD oracle with $\alpha' \geqslant \alpha_1^{min}(b) := \frac{1}{2}\sqrt{1-2b\ln 2} > \alpha$. We will show that the $\alpha$-BDD of Corollary 5.6 is always more efficient than the $\alpha'$-BDD of Corollary 5.4 in this case and the complexity exponent will thus be $A_2(\alpha, b) = -\frac{1}{2\ln 2} W(-\frac{4\alpha^2\beta(\mathcal{L})^2}{e})$.

Assuming that $b < b^*$, we claim that $A_1(\alpha, b) \geqslant A_2(\alpha', b)$ for any $\alpha' \geqslant \alpha_1^{min}(b)$. Indeed, on the one hand, $A_2$ is an increasing function of $b$ so

$$\begin{aligned} A_2(\alpha, b) < A_2\left(\alpha, \frac{5}{18\ln 2}\right) &= -\frac{1}{2\ln 2} W(-4\alpha^2 e^{-4\alpha^2}) \\ &= \frac{4\alpha^2}{2\ln 2} = \frac{2\alpha^2}{\ln 2} \qquad \text{since } W(xe^x) = x. \end{aligned}$$

On the other hand, $A_1$ is an increasing function of $\alpha$ so

$$A_1(\alpha, b) \geqslant A_1(\alpha_1^{min}(b), b) = \frac{1-2b\ln 2}{2\ln 2},$$

which is a decreasing function of $b$, and therefore

$$A_1(\alpha, b) \geqslant A_1(\alpha_1^{min}(b^*), b^*) = \frac{1-2b^*\ln 2}{2\ln 2} = \frac{2\alpha^2}{\ln 2} > A_2(\alpha, b). \qquad \square$$

COROLLARY 5.8. *For any sufficiently large integer $n$, any integer $m > 0$, and any lattice $\mathcal{L} \subset \mathbb{R}^n$, there exists a randomized algorithm that creates a classical and a quantum (with QRAM) 1/3-BDD oracle in time $2^{0.6604n+o(n)}$ and space $2^{0.5n+o(n)}$. Every call to the classical oracle takes time $2^{0.1604n+o(n)}$ and space poly$(n)$, excluding the space of the preprocessed data. Every call to the quantum oracle with QRAM takes time $2^{0.0802n+o(n)}$, classical space poly$(n)$, and poly$(n)$ qubits and requires a QRAM of size $2^{0.1604n+o(n)}$ that contains the preprocessed data.*

*Proof.* By Theorem 5.7, the value of $A$ increases with $b = \log_2 \beta(\mathcal{L})$. Since $b \leqslant 0.402$ and $0.402 \geqslant \frac{5}{18\ln 2}$, we always have $A \leqslant \frac{4}{5}0.402 + o(1) \leqslant 0.3208 + o(1)$ and we obtain the result. $\square$

**6. Solving SVP by spherical caps on the sphere.** We now explain how to reduce the number of queries to the $\alpha$-BDD oracle. Consider a uniformly random target vector $\boldsymbol{t}$ such that $\alpha(1-\frac{1}{n})\lambda_1(\mathcal{L}) \leq \|\boldsymbol{t}\| < \alpha\lambda_1(\mathcal{L})$; it satisfies the condition of Theorem 2.19, i.e., $\mathrm{dist}(\mathcal{L}, \boldsymbol{t}) \leq \alpha\lambda_1(\mathcal{L})$. We enumerate all lattice vectors within

**Algorithm 2.** Solving SVP by spherical caps on the sphere

**Require:** basis $\boldsymbol{B}$ of a lattice $\mathcal{L} \subset \mathbb{R}^n$
**Require:** an $\alpha$-BDD oracle (for a well-chosen $\alpha$)
**Ensure:** a shortest nonzero vector of $\mathcal{L}$

Use LLL to get a number $d$ that satisfies $\lambda_1(\mathcal{L}) \le d \le 2^{n/2}\lambda_1(\mathcal{L})$.
$\boldsymbol{z} \leftarrow$ any basis vector
**for** $i = 1, \ldots, n^2$ **do**
  $d_i \leftarrow d(1 + \frac{1}{n})^{-i}$
  $N \leftarrow \frac{A_n(\lambda_1)}{V_{n-1}(2\alpha\lambda_1)}$
  **for** $j = 1, \ldots, N$ **do**
    $\boldsymbol{v}_{ij} \leftarrow$ random vector of norm $r(1 - \frac{1}{n})d_i$
    **for** $\boldsymbol{x} \in \{0,1\}^n$ **do**
      $\boldsymbol{y} \leftarrow f_{ij}(\boldsymbol{x})$
      **if** $\|\boldsymbol{y}\| < \|\boldsymbol{x}\|$ **then**
        $\boldsymbol{z} \leftarrow \boldsymbol{y}$
      **end if**
    **end for**
  **end for**
**end for**
**return** $\boldsymbol{z}$

distance $2\alpha\lambda_1(\mathcal{L})$ of $\boldsymbol{t}$ and keep only the shortest nonzero one. We show that for any $\alpha \geqslant \frac{1}{3}$, we will get the shortest nonzero vector of the lattice with probability at least $2^{-cn+o(n)}$ for some $c$ that depends on $\alpha$. By repeating this $2^{cn+o(n)}$ times, the algorithm will succeed with constant probability. The optimal choice of $\alpha$ is not obvious and is deferred to section 7.

THEOREM 6.1. *Assume we can create an $\alpha$-BDD oracle, with $\alpha \geqslant \frac{1}{3}$, in time $T_c$ and space $S_c$ such that each call takes time $T_o$. Then there is a randomized algorithm that solves, with constant probability, SVP in space $S_c$ and time*

$$T_c + \frac{2^{o(n)}T_o}{\alpha^n}.$$

*Furthermore, there is a quantum algorithm that solves SVP in classical space $S_c$ using a polynomial number of qubit and time*

$$T_c + \frac{2^{o(n)}T_o}{\alpha^{n/2}}.$$

*If we can create a quantum $\alpha$-BDD oracle with QRAM in classical time $T_c$ and space $S_c$ such that each call takes quantum time $T_o$, poly$(n)$ qubits and requires a QRAM of size $M_o$, then there is a quantum algorithm with QRAM that solves SVP in classical space $S_c$ using a polynomial number of qubit and time*

$$T_c + \frac{2^{o(n)}T_o}{\alpha^{n/2}}$$

*and requires a QRAM of size $M_o$.*

*Proof.* On input lattice $\mathcal{L}(\mathbf{B})$, use the LLL algorithm [55] to get a number $d$ (the norm of the first vector of the basis) that satisfies $\lambda_1(\mathcal{L}) \le d \le 2^{n/2}\lambda_1(\mathcal{L})$. For $i = 1, \dots, n^2$, let $d_i = d/(1+\frac{1}{n})^i$. There exists a $j$ such that $\lambda_1(\mathcal{L}) \le d_j \le (1+\frac{1}{n})\lambda_1(\mathcal{L})$. We repeat the following procedure for all $i = 1, \dots, n^2$: Fix $N \in \mathbb{N}$ to be fixed later. For $j = 1$ to $N$, pick a uniformly random vector $\boldsymbol{v_{ij}}$ on the surface of the ball of radius $r(1-\frac{1}{n})d_i$. By Theorem 2.19, we can enumerate $2^n$ lattice points using the function $f_{ij} : \mathbb{Z}_2^n \to \mathcal{L}$ defined by

$$f_{ij}(\boldsymbol{x}) = \mathbf{B}\boldsymbol{x} - 2 \cdot \mathsf{BDD}_\alpha(\mathcal{L}, (\mathbf{B}\boldsymbol{x} - \boldsymbol{v_{ij}})/2). \tag{22}$$

At each step we only store the shortest nonzero vector. At the end, we output the shortest among them. The running time of the algorithm is straightforward. We make $2^n$ queries to an $\alpha$-$\mathsf{BDD}$ oracle that takes time $T_o$, and we further repeat this $n^2 N$ times. Therefore, the algorithm takes time $T_c + 2^n \cdot n^2 N \cdot T_o$ and space $S_c$. This entire procedure is summarized in Algorithm 2. To prove the correctness of the algorithm, it suffices to show that there exists an $i \in [n^2]$ for which the algorithm finds the shortest vector with high probability. Recall that there exists an $i$ such that $\lambda_1(\mathcal{L}) \le d_i \le (1+\frac{1}{n})\lambda_1(\mathcal{L})$, and let that index be $k$. We will show that for a uniformly random vector $\boldsymbol{v}$ of length $r(1-\frac{1}{n})d_k$, if we enumerate $2^n$ vectors by the function $f : \mathbb{Z}_2^n \to \mathcal{L}$,

$$f(\boldsymbol{x}) = \mathbf{B}\boldsymbol{x} - 2 \cdot \mathrm{BDD}_\alpha(\mathcal{L}, (\mathbf{B}\boldsymbol{x} - \boldsymbol{v})/2), \tag{23}$$

then with probability $\delta$, whose expression is derived in the next paragraph, there exists $\boldsymbol{x} \in \mathbb{Z}_2^n$ such that $f(\boldsymbol{x})$ is the shortest nonzero lattice vector; we will then choose $N = 1/\delta$ so that repeating $N$ times this process finds the shortest vector with probability bounded from below by a constant. To that aim, we show that we can cover the sphere of radius $\lambda_1$ by $N$ balls of radius $2\alpha\lambda_1$ whose centers are at distance $r(1-\frac{1}{n})d_k \le r\lambda_1$ from the origin (see Figure 1, where we took $r = \alpha$). We have two concentric circles of radius $r(1-\frac{1}{n})d_k$ and $\lambda_1$, and we let $P$ be a uniformly random point on the surface of the ball of radius $r(1-\frac{1}{n})d_k$. A ball of radius $2\alpha\lambda_1$ at center $P$ will cover the spherical cap with angle $\phi$ of the ball of radius $\lambda_1$. For convenience, write $r = s\lambda_1$ for some $s$. We can calculate the optimal choice of $r$ by noting that if we take the center of the caps to be at distance $r$, then the angle $\phi$ satisfies $\cos\phi = \frac{\lambda_1^2 + r^2 - 4\alpha^2\lambda_1^2}{2r\lambda_1} = f(s)$

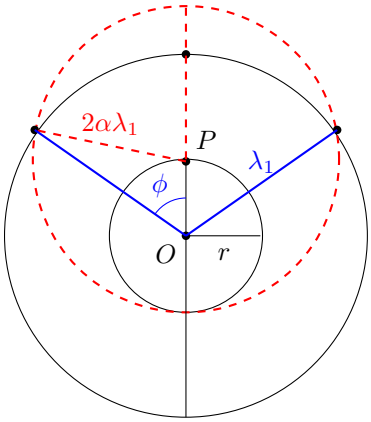


FIG. 1. *One can cover the sphere of radius $\lambda_1$ by balls of radius $2\alpha\lambda_1$, where $\frac{1}{3} \leqslant \alpha < \frac{1}{2}$, whose centers (here $P$) are at distance $r$ from the origin $O$. Each such ball covers a spherical cap of half-angle $\phi$.*

by the law of cosines, where $f(x) = \frac{1+x^2-4\alpha^2}{2x}$. We want to maximize the angle $\phi$, since the area we can cover increases with $\phi$. For minimizing $\cos(\phi)$, we minimize $f$. Check that $f(x)$ is decreasing until $\sqrt{1-4\alpha^2}$ and then increasing. We conclude that the optimal radius is when $s = \sqrt{1-4\alpha^2}$, and this gives an optimal angle $\phi$ such that $\cos\phi = \sqrt{1-4\alpha^2}$ and therefore $\sin\phi = 2\alpha$. Now observe that the surface area of any such cap is lower bounded by the surface area of the base of the cap, which is an $(n-1)$-dimensional sphere of radius $\lambda_1 \sin\phi$. Hence, the number of spherical caps required to cover the surface of sphere is in the order of $N := A_n(\lambda_1)/V_{n-1}(\lambda_1 \sin\phi)$, where $A_n$ (resp., $V_n$) is the surface area (resp., volume) of an $n$-dimensional sphere:

$$A_n(\lambda_1) = \frac{2\pi^{n/2}\lambda_1^{n-1}}{\Gamma(n/2)}, \qquad V_{n-1}(\lambda_1 \sin\phi) = \frac{\pi^{(n-1)/2}\lambda_1^{n-1}\sin^{n-1}\phi}{\Gamma((n+1)/2)}. \tag{24}$$

Thus, we have

$$N = \frac{A_n(\lambda_1)}{V_{n-1}(\lambda_1 \sin\phi)} = \frac{2^{o(n)}}{\sin^{n-1}\phi} = \frac{2^{-n+o(n)}}{\alpha^n}.$$

If we randomly choose the center $\boldsymbol{v}$ of the sphere, the corresponding spherical caps will cover the shortest vector with probability at least $1/N$. By Theorem 2.19, the list $\{f(\boldsymbol{x}) \mid \boldsymbol{x} \in \mathbb{Z}_2^n\}$ will contain all lattice points within radius $2\alpha d_k$ from $\boldsymbol{v}$. Hence, the list will contain a shortest vector with probability $1/N$. By repeating this process $N$ times, we can find the shortest vector with constant probability. Therefore, an upper bound of the total time complexity of our method can be expressed as

$$T_c + 2^n N \cdot T_o = T_c + \frac{2^{o(n)} T_o}{\alpha^n}.$$

In the quantum case, we can apply the quantum minimum finding algorithm in Theorem 2.25 to speed up search. Let $f$ be the function defined in (23). The algorithm works on three quantum registers, and our goal is to build a superposition of states of the form $|\boldsymbol{s}\rangle\,|f(\boldsymbol{s})\rangle\,|x\rangle$, where $x = \|f(\boldsymbol{s})\|$ most of the time (see the definition of $U$ below). Recall that we assumed that we can create a classical $\alpha$-BDD in time $T_c$ and space $S_c$ such that each call takes time $T_o$. Hence, we can first create the oracle in time $T_c$ and space $T_c$ and then hardcode the preprocessed data that we obtained into the oracle to obtain a new, self-contained oracle that still runs in time $T_o$ and polynomial space. Let $\varepsilon = o(\frac{n}{\ln T_o})$, and apply Corollary 2.24 to construct a quantum circuit $\mathcal{O}_{BDD}$ of size $T_o^{1+\varepsilon} = 2^{o(n)} T_o$ on poly$(n)$ qubits that satisfies $\mathcal{O}_{BDD}|\boldsymbol{s}\rangle|\boldsymbol{0}\rangle = |\boldsymbol{s}\rangle|f(\boldsymbol{s})\rangle$ for all $\boldsymbol{s} \in \mathbb{Z}_3^n$. We then construct another quantum circuit $U$ satisfying

$$U(|\boldsymbol{\omega}\rangle|0\rangle) = \begin{cases} |\boldsymbol{\omega}\rangle\,|\,\|\boldsymbol{\omega}\|\rangle & \text{if } \boldsymbol{\omega} \neq \boldsymbol{0}, \\ |\boldsymbol{\omega}\rangle\,|\,\|\boldsymbol{B}\boldsymbol{e_1}\| + 1\rangle & \text{if } \boldsymbol{\omega} = \boldsymbol{0}. \end{cases}$$

Here $\boldsymbol{e_1} \in \mathbb{Z}^n$ is a vector whose first coordinate is one and the rest are zero. We then consider the quantum circuit as per Figure 2 (we have not drawn ancilla qubits):

This circuit $\mathcal{O}$ has size $2^{o(n)} T_o^{1+\varepsilon}$, satisfies $\mathcal{O}\,|\boldsymbol{s}\rangle\,|\boldsymbol{0}\rangle\,|0\rangle = |\boldsymbol{s}\rangle\,|f(\boldsymbol{s})\rangle\,|\|f(\boldsymbol{s})\|\rangle$ if $f(\boldsymbol{s}) \neq 0$ and $\mathcal{O}\,|\boldsymbol{s}\rangle\,|\boldsymbol{0}\rangle\,|0\rangle = |\boldsymbol{s}\rangle\,|f(\boldsymbol{s})\rangle\,|\|\boldsymbol{B}\boldsymbol{e_1}\| + 1\rangle$, and uses poly$(n)$ qubits. We can now apply the quantum minimum finding algorithm from Theorem 2.25 on the first and third registers and the index $\boldsymbol{s'}$ of a shortest vector in this list. The output of the algorithm will be $f(\boldsymbol{s'})$. We can then build a circuit to generate the random vectors $\boldsymbol{v_{ij}}$ above and therefore build a circuit that associates to every $(i, j, \boldsymbol{s})$ the lattice vector $f_{ij}(\boldsymbol{s})$. By an argument similar to the above, using Corollary 2.24, we

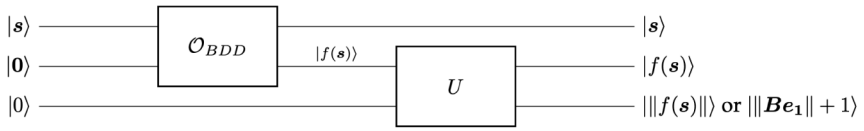


FIG. 2. *Quantum circuit.*

conclude that this quantum circuit has size $2^{o(n)}T_o$ and uses a polynomial number of qubits. Finally, we can apply the quantum minimum finding algorithm on the set of $(i,j,\boldsymbol{s}) \in [n^2] \times [N] \times [2^n]$ and obtain the shortest vector of that list by making $\sqrt{n^2 N 2^n} = 2^{n/2+o(n)}\sqrt{N}$ queries to the BDD oracle. The total running time (including preprocessing) is therefore

$$T_c + 2^{n/2+o(n)}\sqrt{N} \cdot 2^{o(n)}T_o = T_c + \frac{2^{o(n)}T_o}{\alpha^{n/2}}.$$

Last, we show that the quantum algorithm will output a shortest nonzero vector with constant probability. Since $\|\boldsymbol{B}\boldsymbol{e_1}\| + 1 > \lambda_1(\mathcal{L})$, with at least $1/2$ probability one will find the index $(i,j,\boldsymbol{s})$ such that $f_{ij}(\boldsymbol{s})$ is a shortest in the list by using the quantum minimum finding algorithm. Therefore, it suffices to show that there is an index $(i,j,\boldsymbol{s})$ such that $\|f_{ij}(\boldsymbol{s})\| = \lambda_1(\mathcal{L})$. By the analysis above, the list $\{f_{ij}(\boldsymbol{s}) | \boldsymbol{s} \in \mathbb{Z}_3^n\}$ contains the shortest nonzero vector with at least constant probability. Hence, the algorithm will find the shortest vector of the lattice with constant probability. When using a QRAM, the proof is exactly the same, except that we put the samples in a QRAM instead of hardcoding them. This QRAM will have size $M_o$, and everything else is the same. ∎

COROLLARY 6.2. *There is a randomized algorithm that solves* SVP *in time* $2^{1.669n+o(n)}$ *and in space* $2^{0.5n+o(n)}$ *with constant probability. There is a quantum algorithm that solves* SVP *in time* $2^{0.9497n+o(n)}$ *and classical-space* $2^{0.5n+o(n)}$ *with a polynomial number of qubits. There is a quantum algorithm with QRAM that solves the* SVP *in time* $2^{0.8345n+o(n)}$*, using a QRAM of size* $2^{0.293n+o(n)}$*, poly*$(n)$ *qubits and classical space* $2^{0.5n}$.

*Proof.* Apply Theorem 5.7 with[9] $\alpha = 0.3853$: since $0 \leqslant b = \log_2 \beta(\mathcal{L}) \leqslant 0.402$, we indeed have that $\alpha < \frac{1}{2}\sqrt{\frac{1}{1+b}}$ so we can create an $(\alpha + o(1))$-BDD oracle in time $T_c = 2^{(A+1)n/2+o(n)}$ and space $S_c = 2^{0.5n+o(n)}$ such that each call takes time $T_o = 2^{An/2+o(n)}$, where $A = A(b)$ is given by Theorem 5.7. The theorem also guarantees that $A(b)$ increases with $b$ so $A \leqslant A(0.402)$. But $0.402 \geqslant \frac{1-4\alpha^2}{2\ln 2} \approx 0.2929$ so $A(0.402) = \frac{4\alpha^2}{1-4\alpha^2} \cdot 0.402 \approx 0.5862$ by Theorem 5.7. Apply Theorem 6.1 to get a randomized algorithm that solves SVP in space $S_c$ and in time

$$T := T_c + \frac{2^{o(n)}T_o}{\alpha^n} = T = 2^{(A+1)n/2+o(n)} + 2^{An/2-n\log_2\alpha+o(n)} = 2^{1.6690n+o(n)}.$$

For the quantum case, we use $\alpha = 0.3473$ and the same reasoning gives $A \approx 0.3738$ and the total complexity

$$T' := T_c + \frac{2^{o(n)}T_o}{\alpha^n} = T = 2^{(A+1)n/2+o(n)} + 2^{An/2-n/2\log_2\alpha+o(n)} = 2^{0.9497n+o(n)}.$$

[9]The optimal value of $\alpha$ was found numerically; see section 7.

Finally, for the quantum algorithm with a QRAM, the running time of the quantum oracle is $T'_o = \sqrt{T_o}$ and it uses a QRAM of size $T_q = 2^{An/2+o(n)}$. We use $\alpha = 0.3853$ (again) to obtain $A \approx 0.5862$ and a total time complexity of

$$T'' := T_c + \frac{2^{o(n)}T'_o}{\alpha^n} = T = 2^{(A+1)n/2+o(n)} + 2^{An/4-n/2\log_2\alpha+o(n)} = 2^{0.8345n+o(n)}$$

with a QRAM of size $T_q = 2^{0.293n+o(n)}$. ∎

**7. Dependency of the SVP on a quantity related to the kissing number.** In the previous sections, we obtained several algorithms for SVP and bounded their complexity using the only known bound on the quantity $\beta(\mathcal{L})$, which is related to the lattice kissing number (see (2) and (3)): $\beta(\mathcal{L}) \leqslant 2^{0.402}$. The complexity of those algorithms is highly affected by this quantity, and since $\beta(\mathcal{L})$ can be anywhere between 1 and $2^{0.402}$ (see section 2), we will study the dependence of the time complexity in $\beta(\mathcal{L})$. Recall that $b = \log_2 \beta(\mathcal{L})$. In order to avoid doing the analysis three times, we introduce a factor $\nu$ that is 1 for classical algorithms and $\frac{1}{2}$ for quantum algorithms. We also introduce a factor $\xi$ that is 1 for classical and plain quantum algorithms and $\frac{1}{2}$ for quantum algorithms with QRAMs. We now can reformulate the time complexity in Theorem 6.1 as

$$T_c + \frac{2^{o(n)}T_o}{\alpha^{\nu n}}. \tag{25}$$

We instantiate the algorithm in Theorem 6.1 with the $\alpha$-BDD oracle provided by Theorem 5.7, which satisfies

$$T_c = 2^{(A+1)n/2+o(n)}, \qquad T_o = 2^{A\xi n/2+o(n)}, \quad \text{and} \quad S_c = 2^{0.5n+o(n)},$$

where

$$A = \begin{cases} -\dfrac{1}{2\ln 2} W\left(-\dfrac{4\alpha^2\beta(\mathcal{L})^2}{e}\right) & \text{when } b < \frac{1-4\alpha^2}{2\ln 2}, \\ \dfrac{4\alpha^2}{1-4\alpha^2} b & \text{when } b \geqslant \frac{1-4\alpha^2}{2\ln 2}. \end{cases}$$

The new expression of the time complexity is

$$2^{(A+1)n/2+o(n)} + 2^{(A\xi/2-\nu\log_2\alpha)n+o(n)}. \tag{26}$$

Note that for the classical and plain quantum algorithms, the preprocessing is always negligible compared to the cost of the queries, but this is not necessarily the case when using a QRAM. The optimal choice of $\alpha$ is not obvious: by increasing the decoding radius, we reduce the number of queries but increase the cost of each query. While we could, in principle, obtain closed-form expressions for the optimal value of $\alpha$, those are too complicated to be really helpful. Instead, we express it as an optimization program. Formally, we have $T = 2^{c(b,\nu,\xi)n+o(n)}$, where

$$c(b,\nu,\xi) = \min_{\alpha\in[\frac{1}{3},\frac{1}{2})} \max\left(\frac{A+1}{2}, \frac{A\xi}{2} - \nu\log_2\alpha)\right),$$

where $A$ and $\cos\phi$ are given by the expressions above that depend on $\alpha$. We numerically computed the graph of this function and plotted the result in Figure 3 for the classical, plain quantum and quantum with QRAM algorithms, respectively. As mentioned earlier, it is reasonable to conjecture that $\gamma(\mathcal{L}) = 2^{o(n)}$ for most lattices. We obtain the following result when $\gamma(\mathcal{L}) = \beta(\mathcal{L})^n$ is subexponential in $n$.

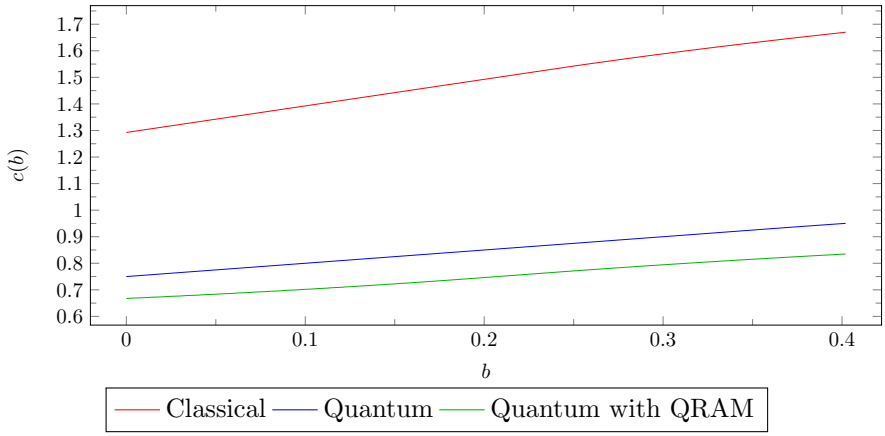


FIG. 3. *(Exponent $c(b)$ of the) time complexity of the spherical capping algorithm, plotted against $b = \log_2 \beta(\mathcal{L})$. The complexity of the algorithms is $2^{c(b)n+o(n)}$.*

THEOREM 7.1. *There is an algorithm that, given a full rank lattice $\mathcal{L}$ of rank $n$ with $\gamma(\mathcal{L}) = 2^{o(n)}$, finds a shortest vector in $\mathcal{L}$ and runs*

- *in classical time $2^{1.292n+o(n)}$ and space $2^{0.5n}$,*
- *in quantum time $2^{0.750n+o(n)}$, classical space $2^{0.5n}$, and poly$(n)$ qubits,*
- *in quantum time $2^{0.667n+o(n)}$, classical space $2^{0.5n}$, and poly$(n)$ qubits and using a QRAM of size $2^{0.167n+o(n)}$.*

**8. Direct application to the ZLIP problem.** The lattice isomorphism problem (LIP) consists of recovering an orthogonal linear transformation sending one lattice to another, assuming its existence. The case of the trivial lattice $\mathbb{Z}^n$ is of particular interest (ZLIP) for cryptographic purposes [17, 31]. In [30], the author proposed a reduction from ZLIP of dimension $n$ to SVP of dimension $n/2+1$.

THEOREM 8.1 (see [30]). *ZLIP can be solved by making polynomially many calls to a shortest vector problem (SVP) oracle in dimension at most $n/2+1$.*

Using Corollary 6.2, we directly obtain the following theorem.

THEOREM 8.2. *There exists a provable quantum algorithm that solves (ZLIP) in time $2^{0.417n+o(n)}$. The algorithm requires a QRAM of size $2^{0.147n+o(n)}$, poly$(n)$ qubits, and $2^{0.25n}$ classical space.*

**Appendix A. Reduction from CVP to DGS.** The goal of this section is to improve Theorems 2.14 and 4.14 to not require a sampler for the discrete Gaussian distribution *exactly at the smoothing parameter* $\eta_\varepsilon(\mathcal{L})$. Indeed, we usually do not know $\eta_\varepsilon(\mathcal{L})$, and it is a nontrivial problem to even estimate it [25]. It was stated in [3, Theorem 7.3] that the reduction still holds if we only provide a DGS oracle above the smoothing parameter. For completeness, we provide a self-contained proof in the form of Theorem 5.2. We first prove some technical lemmas on the discrete Gaussian distribution.

LEMMA A.1. *For any lattice $\mathcal{L} \subset \mathbb{R}^n$, $s > 0$, and $r \geqslant s\sqrt{n}/\lambda_1(\mathcal{L})$,*

$$\rho_s(\mathcal{L} \setminus B_n(r\lambda_1(\mathcal{L})) \leqslant r^n \beta(\mathcal{L})^{n+o(n)} \rho_s(\mathcal{L} \setminus \{0\})^{r^2}.$$

*Proof.* Let $t = 1 + 1/n$, $R = r\lambda_1(\mathcal{L})$, $r_i = Rt^i$, and $T_i = B_n(r_{i+1}) \setminus B_n(r_i)$ for all $i \in \mathbb{N}$. Then, by the definition of $\beta(\mathcal{L})$,

$$|\mathcal{L} \cap T_i| \leqslant |\mathcal{L} \cap B_n(r_{i+1})| \leqslant \beta(\mathcal{L})^{n+o(n)}(rt^{i+1})^n.$$

It follows that

$$\begin{aligned}\rho_s(\mathcal{L} \setminus B_n(R)) &= \sum_{i=0}^{\infty} \rho_s(\mathcal{L} \cap T_i) \\ &\leqslant \sum_{i=0}^{\infty} |\mathcal{L} \cap T_i| e^{-\pi \frac{r_i^2}{s^2}} \\ &\leqslant \beta(\mathcal{L})^{n+o(n)} \sum_{i=0}^{\infty} \underbrace{(rt^{i+1})^n e^{-\pi \frac{R^2}{s^2} t^{2i}}}_{=f(i)},\end{aligned}$$

where $f(i) = (rt^{i+1})^n e^{-\pi \frac{R^2}{s^2} t^{2i}}$. But check that for all $i \in \mathbb{N}$,

$$\frac{f(i+1)}{f(i)} = t^n e^{-\pi \frac{R^2}{s^2} t^{2i}(t^2-1)} \leqslant e^{1-2\pi \frac{R^2}{ns^2}} \leqslant e^{1-3\pi} < 1/2,$$

where we have used that $R \geqslant s\sqrt{n}$, $t^2 - 1 \leqslant \frac{3}{n}$, $t^n \leqslant e$ and $t^{2i} \geqslant 1$. It follows that

$$\begin{aligned}\rho_s(\mathcal{L} \setminus B_n(R)) &\leqslant \beta(\mathcal{L})^{n+o(n)} \cdot 2 \cdot f(0) \leqslant \beta(\mathcal{L})^{n+o(n)} \cdot 2(rt)^n e^{-\pi \frac{R^2}{s^2}} \\ &\leqslant r^n \beta(\mathcal{L})^{n+o(n)} \cdot e^{-\pi \frac{r^2 \lambda_1(\mathcal{L})}{s^2}}.\end{aligned}$$

On the other hand,

$$\rho_s(\mathcal{L} \setminus \{0\}) \geqslant e^{-\pi \frac{\lambda_1(\mathcal{L})^2}{s^2}},$$

so the result follows immediately. ∎

LEMMA A.2. *For any $c > 1$, for any lattice $\mathcal{L} \subset \mathbb{R}^n$, $\varepsilon \in (0, 1/e)$, we have $t\eta_\varepsilon(\mathcal{L}) \leqslant \eta_{\varepsilon^{t^2} e^{o(1)}}(\mathcal{L})$, where $t = 1 + \frac{1}{n^c}$.*

*Proof.* Let $s = \eta_\varepsilon(\mathcal{L})$, $M = n^c$, and $t = 1 + \frac{1}{M}$, with $c > 1$. Let $r = M$, and check that

$$\frac{\sqrt{n}}{s\lambda_1(\mathcal{L}^*)} \leqslant \sqrt{\frac{n\pi}{\ln(1/\varepsilon)}} \leqslant r \tag{27}$$

by Lemma 2.16 for large enough $n$ since $\varepsilon \leqslant 1/e$. Let $\Gamma = (\mathcal{L}^* \setminus \{0\}) \cap B_n(r\lambda_1(\mathcal{L}^*))$; then

$$|\Gamma| \leqslant \beta(\mathcal{L}^*)^{n+o(n)} r^n.$$

Now observe that

$$\begin{aligned}\rho_{1/ts}(\mathcal{L}^* \setminus \{0\}) &= \sum_{x \in \mathcal{L}^* \setminus \{0\}} \rho_{1/ts}(x) = \sum_{x \in \mathcal{L}^* \setminus \{0\}} \rho_{1/s}(x)^{t^2} \\ &\geqslant \sum_{x \in \Gamma} \rho_{1/s}(x)^{t^2} \geqslant |\Gamma| \left( \frac{1}{|\Gamma|} \sum_{x \in \Gamma} \rho_{1/s}(x) \right)^{t^2} \qquad \text{by Jensen's inequality} \\ &= |\Gamma|^{1-t^2} \rho_{1/s}(\Gamma)^{t^2} \\ &= |\Gamma|^{1-t^2} \left( \rho_{1/s}(\mathcal{L}^* \setminus \{0\}) - \rho_{1/s}(\mathcal{L}^* \setminus B_n(r\lambda_1(\mathcal{L}^*))) \right)^{t^2} \\ &\geqslant \left( \beta(\mathcal{L}^*)^{n+o(n)} r^n \right)^{1-t^2} \left( \varepsilon - \beta(\mathcal{L}^*)^{n+o(n)} r^n \varepsilon^{r^2} \right)^{t^2}\end{aligned}$$

by (27) and Lemma A.1 and since $\rho_{1/s}(\mathcal{L}^* \setminus \{0\}) = \varepsilon$. Now observe that $t^2 - 1 \leqslant 3/M$ and

$$\left(\beta(\mathcal{L}^*)^{n+o(n)} r^n\right)^{-3/M} = e^{-3n^{1-c}(\ln \beta(\mathcal{L}^*) + o(1) + c \ln n)} = e^{-3n^{1-c} O(\ln n)} = e^{-o(1)}$$

since $c > 1$. We also have that

$$\begin{aligned}\beta(\mathcal{L}^*)^{n+o(n)} r^n \varepsilon^{r^2} &= e^{n \ln \beta(\mathcal{L}^*) + nc \ln(n) + o(n) - (r^2-1) \ln \frac{1}{\varepsilon}} \varepsilon \\ &\leqslant e^{O(n \ln n) - n^{2c}} \varepsilon && \text{since } \varepsilon \leqslant 1/e \\ &\leqslant e^{o(1)} \varepsilon\end{aligned}$$

for large enough $n$. It follows that

$$\rho_{1/ts}(\mathcal{L}^* \setminus \{0\}) \geqslant e^{o(1)} (\varepsilon e^{o(1)})^{t^2} \geqslant \varepsilon^{t^2} e^{o(1)}$$

since $t \leqslant 2$. Therefore, we must have $\eta_{\varepsilon^{t^2} e^{o(1)}}(\mathcal{L}) \geqslant ts$. ∎

We are in a position to prove the theorem, which we restate below.

THEOREM 5.2. *For any $\alpha > 0$, any $e^{-n^2} \leqslant \varepsilon \leqslant \min(e^{-n^\alpha}, 1/200)$. There exists an algorithm that, on input $\mathcal{L}$ and with constant probability, constructs a classical and a quantum (with QRAM) $\frac{\phi(\mathcal{L})}{\lambda_1(\mathcal{L})}$-BDD oracle for $\mathcal{L}$ by doing poly$(n)$ calls to a* 0.5-hDGS$^m_{\eta_\varepsilon}$ *sampler on the lattice $\mathcal{L}^*$ and requires storage space $m \cdot$ poly$(n)$, where $m = O(\frac{n \ln(1/\varepsilon)}{\sqrt{\varepsilon}})$ and $\phi(\mathcal{L}) \equiv \frac{\sqrt{\ln(1/\varepsilon)/\pi - o(1)}}{2\eta_\varepsilon(\mathcal{L}^*)}$. Each call to the classical oracle takes time $m \cdot$poly$(n)$ and space $O($poly$(n) + \ln m)$, excluding the storage space of the preprocessing. Every call to the quantum oracle with QRAM takes time $\sqrt{m} \cdot$ poly$(n)$, classical space $O($poly$(n) + \ln m)$, and poly$(n)$ qubits and requires a QRAM of size $m \cdot$ poly$(n)$ that contains the preprocessed data.*[10]

*Proof.* First, we note that we can easily identify an interval $I = [a, b]$ such that $\eta_\varepsilon(\mathcal{L}^*) \in [a, b]$ and $\frac{b}{a} \leqslant 2^{n+o(n)}$. Indeed, by, e.g., [70, Lemma 2.11 and Claim 2.13], one has

$$\sqrt{\ln(1/\varepsilon)}\pi \leqslant \lambda_1(\mathcal{L}) \eta_\varepsilon(\mathcal{L}^*) \leqslant \sqrt{n}$$

so $\eta_\varepsilon(\mathcal{L}^*) \in \frac{1}{\lambda_1(\mathcal{L})}[c, d]$, where $\frac{d}{c} = \sqrt{\frac{n}{\ln(1/\varepsilon)}} = O(\sqrt{n})$ since $\varepsilon \leqslant 1/200$. Furthermore, by running the LLL algorithm on $\mathcal{L}$ and taking the length of the shortest basis vector, we obtain a length $\ell$ such that $2^{-n}\ell \leqslant \lambda_1(\mathcal{L}) \leqslant \ell$. It follows that $\eta_\varepsilon(\mathcal{L}^*) \in [a, b] := [\frac{c}{\ell}, \frac{2^n d}{\ell}]$ and $\frac{b}{a} = 2^n \frac{d}{c} = 2^{n+o(n)}$. Now let $c = 2 + \alpha$, $\delta = 1 + \frac{1}{n^c}$, and $N = \lceil \frac{\ln(b/a)}{\ln \delta} \rceil$. Check that $N = \text{poly}(n)$ since $\frac{b}{a} = 2^{n+o(n)}$. Furthermore, if we let $s_i = a\delta^i$ for $i = 1, \ldots, N$, then there must exist some $i_0$ such that

$$\frac{1}{\delta} s_{i_0} \leqslant \eta_\varepsilon(\mathcal{L}^*) \leqslant s_{i_0}. \tag{28}$$

The preprocessing stage of the algorithm consists in calling the hDGS$^m_\sigma$ sampler, where $\sigma(\mathcal{L}) = \eta_\varepsilon(\mathcal{L}^*)$, with parameter $s_i$ to obtain a list $L_i$ of $m$ vectors for each $i = 1, \ldots, N$ and storing all the lists. This requires $N = \text{poly}(n)$ calls, and we need to store $m \cdot \text{poly}(n)$ vectors. Note that the hDGS sampler is allowed to return less than $m$ samples if $s_i > \sigma(\mathcal{L})$: in this case, we do not care about the distribution

[10] Here we are assuming that the Gaussian samples appear in the streaming fashion.

of the vectors anyway, so we can add random vectors until we get $m$ samples when that happens. We now describe the oracle for $\mathsf{CVP}^{\phi}$. On input $\boldsymbol{t} \in \mathbb{R}^n$, for each $i = 1, \ldots, N$, the oracle calls the algorithm of Theorem 2.14 on input $\boldsymbol{t}$ and provides the list $L_i$ to the algorithm in place of the DGS samples. Hence, for each $i$, we either obtain a lattice vector $\boldsymbol{y}_i$ or the algorithm from Theorem 2.14 fails and we let $\boldsymbol{y}_i = \boldsymbol{0}$. Finally, the oracle returns the point closest to $\boldsymbol{t}$ in the list $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_N$. As noted in Remark 2.15, if we assume that all the DGS samples have $\mathrm{poly}(n)$ bit-size, then the reduction from Theorem 2.14 has time complexity $m \cdot \mathrm{poly}(n)$ and space complexity $O(\mathrm{poly}(n) + \ln m)$, *excluding the storage space of the $m$ vectors provided by the DGS.* Furthermore, when the basis vectors of $\mathcal{L}$ have bit-size $\mathrm{poly}(n)$ (which is the case in this paper by assumption) we can ensure that all DGS samples have $\mathrm{poly}(n)$ bit-size by first generating more samples (say twice the amount) and throwing away all samples of norm larger than $\exp(\Omega(n^2))$. Since the vectors are sampled from a Gaussian with width at most $\exp(O(n))$ (since the basis vectors have size at most $2^{o(n)}$, the error induced by throwing away the tail of the distribution is smaller than $2^{-\Omega(n^2)}$. In the quantum setting, we use Theorem 4.15 instead of Theorem 2.14, which gives exactly the same result except for the exponent in the complexity of the oracle. Therefore, running time is clear, so it remains to prove that the algorithm actually solves $\mathsf{CVP}^{\phi}$ on $\mathcal{L}$. We first note that when called on $s_{i_0}$, the $\mathrm{hDGS}^m_\sigma$ sampler will return $m$ vectors that are $0.5-$close to $m$ samples from $D^m_{\mathcal{L},s}$ since $s_{i_0} \geqslant \sigma(\mathcal{L}) = \eta_\varepsilon(\mathcal{L}^*)$ by (28). Furthermore, by Lemma A.2 we have

$$t\eta_\varepsilon(\mathcal{L}^*) \leqslant \eta_{\varepsilon^{t^2} e^{o(1)}}(\mathcal{L}^*),$$

where $t = 1 + \frac{1}{n^c} = \delta$. It follows by (28) that

$$\eta_\varepsilon(\mathcal{L}) \leqslant s_{i_0} \leqslant \delta\eta_\varepsilon(\mathcal{L}^*) \leqslant \eta_{\varepsilon^{\delta^2} e^{o(1)}}(\mathcal{L}^*).$$

But the map $\varepsilon \mapsto \eta_\varepsilon(\mathcal{L})$ is continuous and decreasing, so it follows that

$$s_{i_0} = \eta_{\varepsilon'}(\mathcal{L}^*) \quad \text{for some } \varepsilon^{\delta^2} e^{o(1)} \leqslant \varepsilon' \leqslant \varepsilon.$$

Therefore, by Theorem 2.14 and Remark 2.15, with constant probability over the choice of $L_{i_0}$, the (deterministic) algorithm of Theorem 2.14 solves $\mathsf{CVP}^{\psi}$ when given $L_{i_0}$, where

$$\psi(\mathcal{L}) = \frac{\sqrt{\ln(1/\varepsilon')/\pi - o(1)}}{2\eta_{\varepsilon'}(\mathcal{L}^*)},$$

assuming that $m = |L_{i_0}| \geqslant m' := O(\frac{n \ln(1/\varepsilon')}{\sqrt{\varepsilon'}})$, which holds because

$$\frac{n\ln(1/\varepsilon')}{\sqrt{\varepsilon'}} \leqslant \frac{n\delta^2 \ln(1/\varepsilon) + o(n)}{\sqrt{\varepsilon^{\delta^2} e^{o(1)}}} \leqslant \frac{n\ln(1/\varepsilon) + n^{1-c}\ln(1/\varepsilon) + o(n)}{\sqrt{\varepsilon}\varepsilon^{n^{-c}/2} e^{o(1)}} = O\left(\frac{n\ln(1/\varepsilon)}{\sqrt{\varepsilon}}\right)$$

since $n^{1-c}\ln\varepsilon = n^{1-c+\alpha} = o(1)$ and $\varepsilon^{n^{-c}/2} = e^{n^{\alpha-c}/2} = e^{o(1)}$. It follows that, with constant probability over the preprocessing, our oracle solves $\mathsf{CVP}^{\psi}$. Check that $\varepsilon^{\delta^2} e^{o(1)} \geqslant \varepsilon^{\delta^2 + o(1)}$ since $\varepsilon < \frac{1}{e}$, and thus by Lemma 2.10,

$$\eta_{\varepsilon^{\delta^2} e^{o(1)}}(\mathcal{L}^*) \leqslant \eta_{\varepsilon^{\delta^2 + o(1)}}(\mathcal{L}^*) \leqslant (\delta^2 + o(1))\eta_\varepsilon(\mathcal{L}^*) \leqslant (1 + o(1))\eta_\varepsilon(\mathcal{L}^*)$$

since $\delta = 1 + o(1)$. Hence, we have

$$\psi(\mathcal{L}) \geqslant \frac{\sqrt{\ln(1/\varepsilon)/\pi - o(1)}}{2(1 + o(1))\eta_\varepsilon(\mathcal{L}^*)} \equiv \phi(\mathcal{L}). \qquad \square$$

**Acknowledgments.** We would like to thank Pierre-Alain Fouque, Paul Kirchner, Amaury Pouly, and Noah Stephens-Davidowitz for useful comments and suggestions.

## REFERENCES

[1] D. Aggarwal, Y. Chen, R. Kumar, Z. Li, and N. Stephens-Davidowitz, *Dimension-preserving reductions between SVP and CVP in different p-norms*, in Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA), 2021, pp. 2444–2462, https://doi.org/10.1137/1.9781611976465.145.

[2] D. Aggarwal, Y. Chen, R. Kumar, and Y. Shen, *Improved (provable) algorithms for the shortest vector problem via bounded distance decoding*, in Proceedings of the 38th International Symposium on Theoretical Aspects of Computer Science (STACS), Schloss Dagstuhl-Leibniz-Zentrum für Informatik, Wadern, Germany, 2021.

[3] D. Aggarwal, D. Dadush, O. Regev, and N. Stephens-Davidowitz, *Solving the shortest vector problem in $2^n$ time using discrete Gaussian sampling: Extended abstract*, in Proceedings of the Forty-Seventh Annual ACM Symposium on Theory of Computing (STOC), 2015, pp. 733–742, https://doi.org/10.1145/2746539.2746606.

[4] D. Aggarwal, J. Li, P. Q. Nguyen, and N. Stephens-Davidowitz, *Slide reduction, revisited—Filling the gaps in SVP approximation*, in Advances in Cryptology - (CRYPTO) 2020 - 40th Annual International Cryptology Conference, Santa Barbara, CA, 2020, https://doi.org/10.1007/978-3-030-56880-1.

[5] D. Aggarwal and N. Stephens-Davidowitz, *(Gap/S)ETH Hardness of SVP*, in Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, 2018, pp. 228–238.

[6] D. Aggarwal and N. Stephens-Davidowitz, *Just take the average! An embarrassingly simple $2^{\hat{o}n}$-time algorithm for SVP (and CVP)*, in Proceedings of the1st Symposium on Simplicity in Algorithms, (SOSA), 2018, pp. 12:1–12:19, https://doi.org/10.4230/OASIcs.SOSA.2018.12.

[7] D. Aharonov and O. Regev, *Lattice problems in $NP \cap coNP$*, J. ACM, 52 (2005), pp. 749–765, https://doi.org/10.1145/1089023.1089025.

[8] M. Ajtai, *Generating hard instances of lattice problems (extended abstract)*, in Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing (STOC), 1996, pp. 99–108, https://doi.org/10.1145/237814.237838.

[9] M. Ajtai, R. Kumar, and D. Sivakumar, *A sieve algorithm for the shortest lattice vector problem*, in Proceedings of the Thirty-third Annual ACM Symposium on Theory of Computing (STOC), 2001, pp. 601–610, https://doi.org/10.1145/380752.

[10] M. R. Albrecht, S. Bai, P.-A. Fouque, P. Kirchner, D. Stehlé, and W. Wen, *Faster enumeration-based lattice reduction: Root hermite factor time*, in Proceedings of the Annual International Cryptology Conference, Springer, 2020, pp. 186–212.

[11] M. R. Albrecht, L. Ducas, G. Herold, E. Kirshanova, E. W. Postlethwaite, and M. Stevens, *The general sieve kernel and new records in lattice reduction*, in Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques, Springer, 2019, pp. 717–746.

[12] Y. Aono, P. Q. Nguyen, and Y. Shen, *Quantum lattice enumeration and tweaking discrete pruning*, in Advances in Cryptology—ASIACRYPT 2018, T. Peyrin and S. Galbraith, eds., Springer, Cham, 2018, pp. 405–434.

[13] S. Bai, T. Laarhoven, and D. Stehlé, *Tuple lattice sieving*, LMS J. Comput. Math., 19 (2016), pp. 146–162, https://doi.org/10.1112/S1461157016000292.

[14] W. Banaszczyk, *New bounds in some transference theorems in the geometry of numbers*, Math. Ann., 296 (1993), pp. 625–635, https://doi.org/10.1007/BF01445125.

[15] A. Becker, L. Ducas, N. Gama, and T. Laarhoven, *New directions in nearest neighbor searching with applications to lattice sieving*, in Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), 2016, pp. 10–24, https://doi.org/10.1137/1.9781611974331.ch2.

[16] C. H. Bennett, *Time/space trade-offs for reversible computation*, SIAM J. Comput., 18 (1989), pp. 766–776, https://doi.org/10.1137/0218053.

[17] H. Bennett, A. Ganju, P. Peetathawatchai, and N. Stephens-Davidowitz, *Just how hard are rotations of $\mathbb{z}^n$? algorithms and cryptography with the simplest lattice*, in Advances in Cryptology—EUROCRYPT 2023: 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Proceedings, Part V, Springer-Verlag, Berlin, Heidelberg, 2023, pp. 252–281, https://doi.org/10.1007/978-3-031-30589-4_9.

[18] H. Bennett, A. Golovnev, and N. Stephens-Davidowitz, *Difficulties Constructing Lattices with Exponential Kissing Number from Codes*, preprint, https://arxiv.org/abs/2410.16660, 2024.

[19] M. Boyer, G. Brassard, P. Høyer, and A. Tapp, *Tight bounds on quantum searching*, Fortschr. Phys., 46 (1998), pp. 493–505, https://doi.org/10.1002/(SICI)1521-3978(199806)46:4/5⟨493::AID-PROP493⟩3.0.CO;2-P.

[20] Z. Brakerski, A. Langlois, C. Peikert, O. Regev, and D. Stehlé, *Classical hardness of learning with errors*, in Proceedings of the Forty-Fifth Annual ACM Symposium on Theory of Computing (STOC), 2013, pp. 575–584.

[21] Z. Brakerski and V. Vaikuntanathan, *Lattice-based FHE as secure as PKE*, in Proceedings of the 5th Conference on Innovations in Theoretical Computer Science (ITCS), 2014, pp. 1–12, https://doi.org/10.1145/2554797.2554799.

[22] G. Brassard, P. Høyer, M. Mosca, and A. Tapp, *Quantum amplitude amplification and estimation*, in Quantum Computation and Information, Contemp. Math. 305, AMS, Providence, RI, 2002, pp. 53–74, https://doi.org/10.1090/conm/305/05215.

[23] E. F. Brickell, *Breaking iterated knapsacks*, in Advances in Cryptology, Proceedings of CRYPTO '84, 1984, pp. 342–358, https://doi.org/10.1007/3-540-39568-7.

[24] Y. Chen, K. Chung, and C. Lai, *Space-efficient classical and quantum algorithms for the shortest vector problem*, Quantum Inf. Comput., 18 (2018), pp. 285–306, https://doi.org/10.26421/QIC18.3-4-7.

[25] K.-M. Chung, D. Dadush, F.-H. Liu, and C. Peikert, *On the lattice smoothing parameter problem*, in Proceedings of the IEEE Conference on Computational Complexity, 2013, pp. 230–241, https://doi.org/10.1109/CCC.2013.31.

[26] R. M. Corless, G. H. Gonnet, D. E. G. Hare, D. J. Jeffrey, and D. E. Knuth, *On the Lambert*W *function*, Adv. Comput. Math., 5 (1996), pp. 329–359, https://doi.org/10.1007/BF02124750.

[27] D. Dadush, O. Regev, and N. Stephens-Davidowitz, *On the closest vector problem with a distance guarantee*, in Proceedings of the 29th IEEE Conference on Computational Complexity (CCC), 2014, pp. 98–109, https://doi.org/10.1109/CCC.2014.18.

[28] R. de Buda, *Some optimal codes have structure*, IEEE J. Sel. Area. Comm., 7 (1989), pp. 893–899, https://doi.org/10.1109/49.29612.

[29] L. Ducas, *Shortest vector from lattice sieving: A few dimensions for free*, in Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Proceedings, Part I, Lecture Notes in Comput. Sci. 10820, J. B. Nielsen and V. Rijmen, eds., Springer, 2018, pp. 125–145, https://doi.org/10.1007/978-3-319-78381-9.

[30] L. Ducas, *Provable lattice reduction of* $\mathbf{z}^n$ *with blocksize* $n/2$, Des. Codes Cryptogr., 92 (2024), pp. 909–916, https://doi.org/10.1007/s10623-023-01320-7.

[31] L. Ducas and W. van Woerden, *On the lattice isomorphism problem, quadratic forms, remarkable lattices, and cryptography*, in Advances in Cryptology – EUROCRYPT 2022: 41st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Proceedings, Part III, Springer-Verlag, Berlin, Heidelberg, 2022, pp. 643–673, https://doi.org/10.1007/978-3-031-07082-2_23.

[32] C. Dürr and P. Høyer, *A Quantum Algorithm for Finding the Minimum*, preprint, http://arxiv.org/abs/quant-ph/9607014, 1999.

[33] A. Frank and É. Tardos, *An application of simultaneous diophantine approximation in combinatorial optimization*, Combinatorica, 7 (1987), pp. 49–65, https://doi.org/10.1007/BF02579200.

[34] N. Gama and P. Q. Nguyen, *Finding short lattice vectors within Mordell's inequality*, in Proceedings of the 40th Annual ACM Symposium on Theory of Computing (STOC), 2008, pp. 207–216, https://doi.org/10.1145/1374376.1374408.

[35] N. Gama, P. Q. Nguyen, and O. Regev, *Lattice enumeration using extreme pruning*, in Advances in Cryptology – EUROCRYPT, H. Gilbert, ed., Springer, Berlin, Heidelberg, 2010, pp. 257–278.

[36] C. Gentry, *Fully homomorphic encryption using ideal lattices*, in Proceedings of the 41st Annual ACM Symposium on Theory of Computing (STOC), 2009, pp. 169–178, https://doi.org/10.1145/1536414.1536440.

[37] C. Gentry, C. Peikert, and V. Vaikuntanathan, *Trapdoors for hard lattices and new cryptographic constructions*, in Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing (STOC), pp. 197–206.

[38] L. K. Grover, *A fast quantum mechanical algorithm for database search*, in Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing (STOC), 1996, pp. 212–219, http://doi.acm.org/10.1145/237814.237866.

[39] G. Hanrot, X. Pujol, and D. Stehlé, *Analyzing blockwise lattice algorithms using dynamical systems*, in Advances in Cryptology – CRYPTO 2011, P. Rogaway, ed., Springer, Berlin, Heidelberg, 2011, pp. 447–464.

[40] G. Hanrot and D. Stehlé, *Improved analysis of Kannan's shortest lattice vector algorithm*, in Advances in Cryptology - CRYPTO 2007, 27th Annual International Cryptology Conference, Proceedings, 2007, pp. 170–186, https://doi.org/10.1007/978-3-540-74143-5.

[41] I. Haviv and O. Regev, *Tensor-based hardness of the shortest vector problem to within almost polynomial factors*, in Proceedings of the Thirty-Ninth Annual ACM Symposium on Theory of Computing (STOC), 2007, pp. 469–477.

[42] B. Helfrich, *Algorithms to construct Minkowski reduced and Hermite reduced lattice bases*, Theoret. Comput. Sci., 41 (1985), pp. 125–139, https://doi.org/10.1016/0304-3975(85)90067-2.

[43] G. Herold and E. Kirshanova, *Improved algorithms for the approximate k-list problem in euclidean norm*, in Public-Key Cryptography – PKC 2017, S. Fehr, ed., Springer, Berlin, Heidelberg, 2017, pp. 16–40.

[44] W. Hoeffding, *Probability inequalities for sums of bounded random variables*, J. Amer. Statist. Assoc., 58 (1963), pp. 13–30, https://doi.org/10.1080/01621459.1963.10500830.

[45] R. Impagliazzo, L. A. Levin, and M. Luby, *Pseudo-random generation from one-way functions*, in Proceedings of the Twenty-First Annual ACM Symposium on Theory of Computing (STOC), 1989, pp. 12–24.

[46] H. W. Lenstra, Jr., *Integer programming with a fixed number of variables*, Math. Oper. Res., 8 (1983), pp. 538–548, https://doi.org/10.1287/moor.8.4.538.

[47] G. A. Kabatjyanskiĭ and V. I. Levenšhteĭn, *On bounds for packings on a sphere and in space*, Problemy Peredachi Informatsii, 14 (1978), pp. 3–25, https://www.mathnet.ru/php/archive.phtml?wshow=paper&jrnid=ppi&paperid=1518&option_lang=eng.

[48] R. Kannan, *Minkowski's convex body theorem and integer programming*, Math. Oper. Res., 12 (1987), pp. 415–440, https://doi.org/10.1287/moor.12.3.415.

[49] S. Khot, *Hardness of approximating the shortest vector problem in lattices*, J. ACM, 52 (2005), pp. 789–808, https://doi.org/10.1145/1089023.1089027.

[50] P. Kirchner and P.-A. Fouque, *Time-Memory Trade-Off for Lattice Enumeration in a Ball*, Cryptology ePrint Archive, Report 2016/222, 2016, https://eprint.iacr.org/2016/222.

[51] E. Kirshanova, E. Mårtensson, E. W. Postlethwaite, and S. R. Moulik, *Quantum algorithms for the approximate k-list problem and their application to lattice sieving*, in Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security, Springer, 2019, pp. 521–551.

[52] P. Klein, *Finding the closest lattice vector when it's unusually close*, in Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), SIAM, Philadelphia, 2000, pp. 937–941.

[53] T. Laarhoven, M. Mosca, and J. Van De Pol, *Finding shortest lattice vectors faster using quantum search*, Des. Codes Cryptogr., 77 (2015), pp. 375–400, https://doi.org/10.1007/s10623-015-0067-5.

[54] J. C. Lagarias and A. M. Odlyzko, *Solving low-density subset sum problems*, J. ACM, 32 (1985), pp. 229–246, https://doi.org/10.1145/2455.2461.

[55] A. Lenstra, H. Lenstra, and L. Lovász, *Factoring polynomials with rational coefficients*, Math. Ann., 261 (1982), pp. 515–534, https://doi.org/10.1007/BF01457454.

[56] R. Y. Levine and A. T. Sherman, *A note on Bennett's time-space tradeoff for reversible computation*, SIAM J. Comput., 19 (1990), pp. 673–677, https://doi.org/10.1137/0219046.

[57] Y.-K. Liu, V. Lyubashevsky, and D. Micciancio, *On bounded distance decoding for general lattices*, in Approximation, Randomization, and Combinatorial Optimization, Algorithms and Techniques, Springer, 2006, pp. 450–461.

[58] D. Micciancio, *The shortest vector in a lattice is hard to approximate to within some constant*, in Proceedings of the 39th Annual IEEE Symposium on Foundations of Computer Science (FOCS), 1998.

[59] D. Micciancio and C. Peikert, *Hardness of SIS and LWE with small parameters*, in Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Proceedings, Part I, Springer, 2013, pp. 21–39, https://doi.org/10.1007/978-3-642-40041-4_2.

[60] D. Micciancio and O. Regev, *Worst-case to average-case reductions based on Gaussian measures*, in Proceedings of the 45th Symposium on Foundations of Computer Science (FOCS), 2004, pp. 372–381, https://doi.org/10.1109/FOCS.2004.72.

[61] D. Micciancio and O. Regev, *Lattice-Based Cryptography*, manuscript, 2008, https://link.springer.com/chapter/10.1007/978-3-540-88702-7_5.
[62] D. Micciancio and P. Voulgaris, *Faster exponential time algorithms for the shortest vector problem*, in Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), 2010, pp. 1468–1480, https://doi.org/10.1137/1.9781611973075.119.
[63] D. Micciancio and P. Voulgaris, *A deterministic single exponential time algorithm for most lattice problems based on Voronoi cell computations*, SIAM J. Comput., 42 (2013), pp. 1364–1391, https://doi.org/10.1137/100811970.
[64] D. Micciancio and M. Walter, *Fast lattice point enumeration with minimal overhead*, in Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), 2015, pp. 276–294, https://doi.org/10.1137/1.9781611973730.21.
[65] P. Q. Nguyen and T. Vidick, *Sieve algorithms for the shortest vector problem are practical*, J. Math. Cryptol., 2 (2008), pp. 181–207, https://doi.org/10.1515/JMC.2008.009.
[66] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information,* 10th Anniversary ed., Cambridge University Press, 2016, https://www.cambridge.org/de/academic/subjects/physics/quantum-physics-quantum-information-and-quantum-computation/quantum-computation-and-quantum-information-10th-anniversary-edition?format=HB.
[67] X. Pujol and D. Stehlé, *Solving the Shortest Lattice Vector Problem in Time* $2^{2.465n}$, IACR Cryptol. ePrint Arch., Paper 2009/605, http://eprint.iacr.org/2009/605.
[68] O. Regev, *Lattices in computer science, lecture* 8*, Fall*, 2004.
[69] O. Regev, *Lattice-based cryptography*, in Advances in Cryptology - CRYPTO 2006, 26th Annual International Cryptology Conference, Proceedings, 2006, pp. 131–141, https://doi.org/10.1007/11818175_8.
[70] O. Regev, *On lattices, learning with errors, random linear codes, and cryptography*, J. ACM, 56 (2009), pp. 34:1–34:40, https://doi.org/10.1145/1568318.1568324.
[71] C. Schnorr and M. Euchner, *Lattice basis reduction: Improved practical algorithms and solving subset sum problems*, Math. Program., 66 (1994), pp. 181–199, https://doi.org/10.1007/BF01581144.
[72] C. P. Schnorr, *A hierarchy of polynomial time lattice basis reduction algorithms*, Theoret. Comput. Sci., 53 (1987), pp. 201–224, https://doi.org/10.1016/0304-3975(87)90064-8.
[73] A. Shamir, *A polynomial-time algorithm for breaking the basic Merkle-Hellman cryptosystem*, IEEE Trans. Inform. Theory, 30 (1984), pp. 699–704, https://doi.org/10.1109/TIT.1984.1056964.
[74] SVP Challenge, https://www.latticechallenge.org/svp-challenge/.
[75] S. Vlăduţ, *Lattices with exponentially large kissing numbers*, Mosc. J. Comb. Number Theory, 8 (2019), pp. 163–177, https://doi.org/10.2140/moscow.2019.8.163.