Improved Algorithms for the Shortest Vector Problem and the Closest Vector Problem in the Infinity Norm

Divesh Aggarwal[1]
Centre for Quantum Technologies and School of Computing, National University of Singapore
dcsdiva@nus.edu.sg

Priyanka Mukhopadhyay[2]
Centre for Quantum Technologies, National University of Singapore
mukhopadhyay.priyanka@gmail.com

Abstract

Ajtai, Kumar and Sivakumar [5] gave the first $2^{O(n)}$ algorithm for solving the Shortest Vector Problem (SVP) on n-dimensional Euclidean lattices. The algorithm starts with $N \in 2^{O(n)}$ randomly chosen vectors in the lattice and employs a sieving procedure to iteratively obtain shorter vectors in the lattice, and eventually obtaining the shortest non-zero vector. The running time of the sieving procedure is quadratic in N. Subsequent works [7, 11] generalized the algorithm to other norms.

We study this problem for the special but important case of the ℓ_∞ norm. We give a new sieving procedure that runs in time linear in N, thereby improving the running time of the algorithm for SVP in the ℓ_∞ norm. As in [6, 11], we also extend this algorithm to obtain significantly faster algorithms for approximate versions of the shortest vector problem and the closest vector problem (CVP) in the ℓ_∞ norm.

We also show that the heuristic sieving algorithms of Nguyen and Vidick [23] and Wang et al. [27] can also be analyzed in the ℓ_∞ norm. The main technical contribution in this part is to calculate the expected volume of intersection of a unit ball centred at origin and another ball of a different radius centred at a uniformly random point on the boundary of the unit ball. This might be of independent interest.

2012 ACM Subject Classification Theory of computation → Randomness, geometry and discrete structures

Keywords and phrases Lattice, Shortest Vector Problem, Closest Vector Problem, ℓ_∞ norm

Digital Object Identifier 10.4230/LIPIcs.ISAAC.2018.35

Related Version A full version of the paper is available at [3], `https://arxiv.org/abs/1801.02358`.

Acknowledgements We thank the anonymous referees who helped improve the draft of this paper.

[1] This research was partially funded by the Singapore Ministry of Education and the National Research Foundation, also through the Tier 3 Grant "Random numbers from quantum processes",MOE2012-T3-1-009.

[2] This research was funded by the National Research Foundation,Prime Minister's Office, Singapore and the Ministry of Education, Singapore.

1 Introduction

A lattice $\mathcal{L}$ is the set of all integer combinations of linearly independent vectors $\mathbf{b}_1, \ldots, \mathbf{b}_n \in \mathbb{R}^d$, $\mathcal{L} = \mathcal{L}(\mathbf{b}_1, \ldots, \mathbf{b}_n) := \{\sum_{i=1}^{n} z_i \mathbf{b}_i : z_i \in \mathbb{Z}\}$.

We call n the rank of the lattice, and d the dimension of the lattice. The matrix $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ is called a basis of $\mathcal{L}$, and we write $\mathcal{L}(\mathbf{B})$ for the lattice generated by $\mathbf{B}$. A lattice is said to be full-rank if $n = d$. In this work, we will only consider full-rank lattices unless otherwise stated.

The two most important computational problems on lattices are the Shortest Vector Problem (SVP) and the Closest Vector Problem (CVP). Given a basis for a lattice $\mathcal{L} \subseteq \mathbb{R}^d$, SVP asks us to compute a non-zero vector in $\mathcal{L}$ of minimal length, and CVP asks us to compute a lattice vector at a minimum distance to a target vector $\mathbf{t}$. Typically the length/distance is defined in terms of the ℓ_p norm for some $p \in [1, \infty]$, such that
$\|\mathbf{x}\|_p := (|x_1|^p + |x_2|^p + \cdots + |x_d|^p)^{1/p}$ for $1 \leq p < \infty$, and $\|\mathbf{x}\|_\infty := \max_{1 \leq i \leq d} |x_i|$.

The most popular of these, and the most well studied is the Euclidean norm, which corresponds to $p = 2$. Starting with the seminal work of [18], algorithms for solving these problems either exactly or approximately have been studied intensely. Some classic applications of these algorithms are in factoring polynomials over rationals [18], integer programming [19], cryptanalysis [22], checking the solvability by radicals [17], and solving low-density subset-sum problems [12]. More recently, many powerful cryptographic primitives have been constructed whose security is based on the *worst-case* hardness of these or related lattice problems(see for example [24] and the references therein).

One recent application that is based on the hardness of SVP in the ℓ_∞ norm is a recent signature scheme by Ducas et al. [13]. For the security of their signature scheme, the authors choose parameters under the assumption that SVP in the ℓ_∞ norm for an appropriate dimension is infeasible. Due to lack of sufficient work on the complexity analysis of SVP in the ℓ_∞ norm, they choose parameters based on the best known algorithms for SVP in the ℓ_2 norm (which are variants of the algorithm from [23]). The rationale for this is that SVP in ℓ_∞ norm is likely harder than in the ℓ_2 norm. Our results in this paper show that this assumption by Ducas et al. [13] is correct, and perhaps too generous. In particular, we show that the space and time complexity of the ℓ_∞ version of [23] is at most $(4/3)^n$ and $(4/3)^{2n}$ respectively, which is significantly larger than the best known algorithms for SVP in the ℓ_2 norm.

The closest vector problem in the ℓ_∞ norm is particularly important since it is equivalent to the integer programming problem [14]. The focus of this work is to study the complexity of the closest vector problem and the shortest vector problem in the ℓ_∞ norm.

1.1 Prior Work

1.1.1 Algorithms in the Euclidean Norm

The fastest known algorithms for solving these problems run in time 2^{cn}, where n is the rank of the lattice and c is some constant. The first algorithm to solve SVP in time exponential in the dimension of the lattice was given by Ajtai, Kumar, and Sivakumar [5] who devised a method based on "randomized sieving," whereby exponentially many randomly generated lattice vectors are iteratively combined to create shorter and shorter vectors, eventually resulting in the shortest vector in the lattice. Subsequent work has resulted in improvement of their sieving technique thereby improving the constant c in the exponent, and the current

fastest provable algorithm for exact SVP runs in time $2^{n+o(n)}$ [1, 4], and the fastest algorithm that gives a constant approximation runs in time $2^{0.802n+o(n)}$ [20]. The fastest heuristic algorithm that is conjectured to solve SVP in practice runs in time $(3/2)^{n/2}$ [9].

The CVP is considered a harder problem than SVP since there is a simple dimension and approximation-factor preserving reduction from SVP to CVP [15]. Based on a technique due to Kannan [16], Ajtai, Kumar, and Sivakumar [6] gave a sieving based algorithm that gives a $1+\alpha$ approximation of CVP in time $(2+1/\alpha)^{O(n)}$. Later exact exponential time algorithms for CVP were discovered [21, 2]. The current fastest algorithm for CVP runs in time $2^{n+o(n)}$ and is due to [2].

1.1.2 Algorithms in Other ℓ_p Norms

Blomer and Naewe [11], and then Arvind and Joglekar [7] generalised the AKS algorithm [5] to give exact algorithms for SVP that run in time $2^{O(n)}$. Additionally, [11] gave a $1+\varepsilon$ approximation algorithm for CVP for all ℓ_p norms that runs in time $(2+1/\varepsilon)^{O(n)}$. For the special case when $p=\infty$, Eisenbrand et al. [14] gave a $2^{O(n)}\cdot(\log(1/\varepsilon))^n$ algorithm for $(1+\varepsilon)$-approx CVP.

1.2 Our contribution

1.2.1 Provable Algorithms

We modify the sieving algorithm by [5, 6] for SVP and approximate CVP for the ℓ_∞ norm that results in substantial improvement over prior results. Before describing our idea, we give an informal description of the sieving procedure of [5, 6]. The algorithm starts by randomly generating a set S of $N\in 2^{O(n)}$ lattice vectors of length at most $R\in 2^{O(n)}$. It then runs a sieving procedure a polynomial number of times. In the i^{th} iteration the algorithm starts with a list S of lattice vectors of length at most $R_{i-1}\approx\gamma^{i-1}R$, for some parameter $\gamma\in(0,1)$. The algorithm maintains and updates a list of "centres" C, which is initialised to be the empty set. Then for each lattice vector $\mathbf{y}$ in the list, the algorithm checks whether there is a centre $\mathbf{c}$ at distance at most $\gamma\cdot R_{i-1}$ from this vector. If there exists such a centre pair, then the vector $\mathbf{y}$ is replaced in the list by $\mathbf{y}-\mathbf{c}$, and otherwise it is deleted from S and added to C. This results in $N_{i-1}-|C|$ lattice vectors which are of length at most $R_i\approx\gamma R_{i-1}$, where N_{i-1} is the number of lattice vectors at the end of $i-1$ sieving iterations. We mention here that this description hides many details and in particular, in order to show that this algorithm eventually obtains the shortest vector, we need to add a little perturbation to the lattice vectors to start with. The details can be found in Section 3.

A crucial step in this algorithm is to find a vector $\mathbf{c}$ from the list of centers that is close to $\mathbf{y}$. This problem is called the nearest neighbor search (NNS) problem and has been well studied especially in the context of heuristic algorithms for SVP (see [9] and the references therein). A trivial bound on the running time for this is $|S|\cdot|C|$, but the aforementioned heuristic algorithms have spent considerable effort trying to improve this bound under reasonable heuristic assumptions. Since they require heuristic assumptions, such improved algorithms for the NNS have not been used to improve the provable algorithms for SVP.

We make a simple but powerful observation that for the special case of the ℓ_∞ norm, if we partition the ambient space $[-R,R]^n$ into $([-R,-R+\gamma\cdot R),[-R+\gamma\cdot R,-R+2\gamma\cdot R),\ldots[-R+\lfloor\frac{2}{\gamma}\rfloor\cdot\gamma\cdot R,R])^n$, then it is easy to see that each such partition will contain at most one centre. Thus, to find a centre at ℓ_∞ distance $\gamma\cdot R$ from a given vector $\mathbf{y}$, we only need to find the partition in which $\mathbf{y}$ belongs, and then check whether this partition contains a centre. This can be easily done by checking the interval in which each co-ordinate

of $\mathbf{y}$ belongs. This drastically improves the running time for the sieving procedure in the SVP algorithm from $|S| \cdot |C|$ to $|S| \cdot n$. Notice that we cannot expect to improve the time complexity beyond $O(|S|)$.

This same idea can also be used to obtain significantly faster approximation algorithms for both SVP and CVP. It must be noted here that the prior provable algorithms using AKS sieve lacked an explicit value of the constant in the exponent for both space and time complexity and they used a quadratic sieve. Our modified sieving procedure is linear in the size of the input list and thus yields a faster algorithm compared to the prior algorithms. In order to get the best possible running time, we optimize several steps specialized to the case of ℓ_∞ norm in the analysis of the algorithms. See Theorems 15, 17, and 18 for explicit running times and a detailed description.

Just to emphasise that our results are nearly the best possible using these techniques, notice that for a large enough constant τ, we obtain a running time (and space) close to 3^n for τ-approximate SVP. To put things in context, the best algorithm [28] for a constant approximate SVP in the ℓ_2 norm runs in time $2^{0.802n}$ and space $2^{0.401n}$. Their algorithm crucially uses the fact that $2^{0.401n}$ is the best known upper bound for the kissing number of the lattice (which is the number of shortest vectors in the lattice) in ℓ_2 norm. However, for the ℓ_∞ norm, the kissing number is 3^n for $\mathbb{Z}^n$. So, if we would analyze the algorithm from [28] for the ℓ_∞ norm (without our improvement), we would obtain a space complexity 3^n, but time complexity 9^n.

1.2.2 Heuristic Algorithms

In each sieving step of the algorithm from [5], the length of the lattice vectors reduce by a constant factor. It seems like if we continue to reduce the length of the lattice vectors until we get vectors of length λ_1 (where λ_1 is the length of the shortest vector), we should obtain the shortest vector during the sieving procedure. However, there is a risk that all vectors output by this sieving procedure are copies of the zero vector and this is the reason that the AKS algorithm [5] needs to start with much more vectors in order to provably argue that we obtain the shortest vector.

Nguyen and Vidick [23] observed that this view is perhaps too pessimistic in practice, and that the randomness in the initial set of vectors should ensure that the basic sieving procedure should output the shortest vector for most lattices, and in particular if the lattice is chosen randomly as is the case in cryptographic applications. The main ingredient to analyze the space and time complexity of their algorithm is to compute the expected number of centres necessary so that any point in S of length at most R_{i-1} is at a distance of at most $\gamma \cdot R_{i-1}$ from one of the centres. This number is roughly the reciprocal of the fraction of the ball B of radius R_{i-1} centred at the origin covered by a ball of radius $\gamma \cdot R_{i-1}$ centred at a uniformly random point in B. Here R_{i-1} is the maximum length of a lattice vector in S after $i-1$ sieving iterations.

In this work, we show that the heuristic algorithm of [23] can also be analyzed for the ℓ_∞ norm under similar assumptions. The main technical contribution in order to analyze the time and space complexity of this algorithm is to compute the expected fraction of an ℓ_∞ ball $B^{(\infty)}$ of radius R_{i-1} centered at the origin covered by an ℓ_∞ ball of radius $\gamma \cdot R_{i-1}$ centered at a uniformly random point in $B^{(\infty)}$.

In order to improve the running time of the NV sieve [23], a modified two-level sieve was introduced by Wang et al. [27]. Here they first partition the lattice into sets of vectors of larger norm and then within each set they carry out a sieving procedure similar to [23]. We have analyzed this in the ℓ_∞ norm and obtain algorithms much faster than the provable

algorithms. In particular, our two-level sieve algorithm runs in time $2^{0.62n}$. We would like to mention here that our result does not contradict the near 2^n lower bound for SVP obtained by [10] under the strong exponential time hypothesis. The reason for this is that the lattice obtained in the reduction in [10] is not a full-rank lattice, and has a dimension significantly larger than the rank n of the lattice. Moreover, as mentioned earlier, the heuristic algorithm is expected to work for a random looking lattice but might not work for *all* lattices.

Due to space constraints, we have deferred some descriptions and analysis to the full version of this paper [3].

1.3 Open problems

It would be interesting to see if such partitioning technique can be done for other norms or combined with heuristic algorithms like NNS, to yield better performance for sieving algorithms. We do not know if other provable algorithms like those based on Discrete Gaussian sampling [1, 2], Voronoi cells [21] or other heuristic algorithms can be analysed in other non-Euclidean norms. Another direction would be to understand the change in time and space complexity as the number of levels for multi-level sieve increases.

1.4 Organization of the paper

In Section 2 we give some basic definitions and results used in this paper. In Section 3 we introduce our sieving procedure and apply it to provably solve exact $\mathsf{SVP}^{(\infty)}$. In Section 4 we describe approximate algorithms for $\mathsf{SVP}^{(\infty)}$ and $\mathsf{CVP}^{(\infty)}$ using our sieving technique. In Section 5 we talk about heuristic sieving algorithms for $\mathsf{SVP}^{(\infty)}$.

2 Preliminaries

2.1 Notations

We write ln for natural logarithm and log for logarithm to the base 2.

▶ **Fact 1.** For $\mathbf{x} \in \mathbb{R}^n$ $\|\mathbf{x}\|_p \leq \|\mathbf{x}\|_2 \leq \sqrt{n}\|\mathbf{x}\|_p$ for $p \geq 2$ and $\frac{1}{\sqrt{n}}\|\mathbf{x}\|_p \leq \|\mathbf{x}\|_2 \leq \|\mathbf{x}\|_p$ for $1 \leq p < 2$.

▶ **Definition 2 (Ball).** A (closed) ball of radius r and centre at $\mathbf{x} \in \mathbb{R}^n$, is the set of all points whose distance (in ℓ_p norm) from $\mathbf{x}$ is at most r. $B_n^{(p)}(\mathbf{x}, r) = \{\mathbf{y} \in \mathbb{R}^n : \|\mathbf{y} - \mathbf{x}\|_p \leq r\}$.

The following result gives a bound on the size of intersection of two balls of a given radius in the ℓ_∞ norm.

▶ **Lemma 3.** *Let $\mathbf{v} = (v_1, v_2, \ldots, v_n) \in \mathbb{R}^n$, and let $a > 0$ be such that $2a \geq \|\mathbf{v}\|_\infty$. Let $D = B_n^{(\infty)}(\mathbf{0}, a) \cap B_n^{(\infty)}(\mathbf{v}, a)$. Then,* $|D| = \prod_{i=1}^n (2a - |v_i|)$.

Proof. It is easy to see that the intersection of two balls in the ℓ_∞ norm, i.e., hyperrectangles, is also a hyperrectangle. For all i, the length of the i-th side of this hyperrectangle is $2a - |v_i|$. The result follows. ◀

2.2 Lattice

▶ **Definition 4.** A lattice $\mathcal{L}$ is a discrete additive subgroup of $\mathbb{R}^n$. Each lattice has a basis $\mathbf{B} = [\mathbf{b}_1, \mathbf{b}_2, \ldots \mathbf{b}_n]$, where $\mathbf{b}_i \in \mathbb{R}^n$ and $\mathcal{L} = \mathcal{L}(\mathbf{B}) = \left\{ \sum_{i=1}^n x_i \mathbf{b}_i : x_i \in \mathbb{Z} \quad i = 1, \ldots, n \right\}$

For algorithmic purposes we can assume that $\mathcal{L} \subseteq \mathbb{Q}^d$.

▶ **Definition 5.** For any lattice basis $\mathbf{B}$ we define the fundamental parallelepiped as: $\mathcal{P}(\mathbf{B}) = \{\mathbf{Bx} : \mathbf{x} \in [0,1)^n\}$

If $\mathbf{y} \in \mathcal{P}(\mathbf{B})$ then $\|\mathbf{y}\|_p \leq n\|\mathbf{B}\|_p$ as can be easily seen by triangle inequality. For any $\mathbf{z} \in \mathbb{R}^n$ there exists a unique $\mathbf{y} \in \mathcal{P}(\mathbf{B})$ such that $\mathbf{z} - \mathbf{y} \in \mathcal{L}(\mathbf{B})$. This vector is denoted by $\mathbf{y} \equiv \mathbf{z} \mod \mathbf{B}$ and it can be computed in polynomial time given $\mathbf{B}$ and $\mathbf{z}$.

▶ **Definition 6.** For $i \in [n]$, the first successive minimum is defined as the length of the shortest non-zero vector in the lattice: $\lambda_1^{(p)}(\mathcal{L}) = \min\{\|\mathbf{v}\|_p : \mathbf{v} \in \mathcal{L} \setminus \{\mathbf{0}\}\}$

We consider the following lattice problems. In all the problems defined below $c \geq 1$ is some arbitrary approximation factor. We drop the subscript for exact versions (i.e. $c = 1$).

1. **Shortest Vector Problem ($\mathsf{SVP}_c^{(p)}$)** Given a lattice $\mathcal{L}$, find a vector $\mathbf{v} \in \mathcal{L} \setminus \{\mathbf{0}\}$ such that $\|\mathbf{v}\|_p \leq c\|\mathbf{u}\|_p$ for any other $\mathbf{u} \in \mathcal{L} \setminus \{\mathbf{0}\}$.
2. **Closest Vector Problem ($\mathsf{CVP}_c^{(p)}$)** Given a lattice $\mathcal{L}$ with rank n and a target vector $\mathbf{t} \in \mathbb{R}^n$, find $\mathbf{v} \in \mathcal{L}$ such that $\|\mathbf{v} - \mathbf{t}\|_p \leq c\|\mathbf{w} - \mathbf{t}\|_p$ for all other $\mathbf{w} \in \mathcal{L}$.

▶ **Lemma 7.** *The LLL algorithm [18] can be used to solve $\mathsf{SVP}_{2^{n-1}}^{(p)}$ in polynomial time.*

The following result shows that in order to solve $\mathsf{SVP}_{1+\epsilon}^{(p)}$, it is sufficient to consider the case when $2 \leq \lambda_1^{(p)}(\mathcal{L}) < 3$. This is done by appropriately scaling the lattice.

▶ **Lemma 8 (Lemma 4.1 in [11]).** *For all ℓ_p norms, if there is an algorithm A that for all lattices $\mathcal{L}$ with $2 \leq \lambda_1^{(p)}(\mathcal{L}) < 3$ solves $\mathsf{SVP}_{1+\epsilon}^{(p)}$ in time $T = T(n, b, \epsilon)$, then there is an algorithm A' that solves $\mathsf{SVP}_{1+\epsilon}^{(p)}$ for all lattices in time $O(nT + n^4 b)$.*

Thus henceforth we assume $2 \leq \lambda_1^{(\infty)}(\mathcal{L}) < 3$.

3 A faster algorithm for $\mathsf{SVP}^{(\infty)}$

In this section we present an algorithm for $\mathsf{SVP}^{(\infty)}$ that uses the framework of AKS algorithm [5] but uses a different sieving procedure that yields a faster running time. Using Lemma 7, we can obtain an estimate λ^* of $\lambda_1^{(\infty)}(\mathcal{L})$ such that $\lambda_1^{(\infty)}(\mathcal{L}) \leq \lambda^* \leq 2^n \cdot \lambda_1^{(\infty)}(\mathcal{L})$. Thus, if we try different values of $\lambda = (1 + 1/n)^{-i}\lambda^*$, for $0 \leq i \leq 10n^2$, then for one of them, we have $\lambda_1^{(\infty)}(\mathcal{L}) \leq \lambda \leq (1 + 1/n) \cdot \lambda_1^{(\infty)}(\mathcal{L})$ For the rest of this section, we assume that we know a guess λ of the length of the shortest vector in $\mathcal{L}$, which is correct upto a factor $1 + 1/n$.

As in the AKS algorithm, we start by generating a set S of many vector pairs $(\mathbf{e}, \mathbf{y})$, where the perturbation vectors $\mathbf{e}$ are uniformly sampled from $B_n^{(\infty)}(\xi\lambda)$ $(\xi > 1/2)$, and $\mathbf{y} \in \mathbf{e} \mod \mathcal{P}(\mathbf{B})$ which has length at most R, where $R \leq n \max_i \|\mathbf{b}_i\|$ and $\mathbf{y} - \mathbf{e} \in \mathcal{L}$. The desired situation is that after a polynomial number of such sieving iterations (sieve) we are left with a set of vector pairs $(\mathbf{e}', \mathbf{y}')$ such that $\mathbf{y}' - \mathbf{e}' \in \mathcal{L} \cap B_n^{(\infty)}(O(\lambda_1^{(\infty)}(\mathcal{L})))$. Finally we take pair-wise differences of the lattice vectors corresponding to the remaining vector pairs and output the one with the smallest non-zero norm. It was shown in [5] that with overwhelming probability, this algorithm outputs the shortest vector in the lattice.

An iteration of the sieving procedure on the pairs of vectors S does the following. We partition the interval $[-R, R]$ into $\ell = 1 + \left\lfloor \frac{2}{\gamma} \right\rfloor$ intervals of length γR. The intervals are $[-R, -R + \gamma R), [-R + \gamma R, -R + 2\gamma R), \ldots [-R + (\ell - 1)\gamma R, R]$. (Note that the last interval may be smaller than the rest.) The ball $[-R, R]^n$ can thus be partitioned into $\left(1 + \left\lfloor \frac{2}{\gamma} \right\rfloor\right)^n$

Algorithm 1: An exact algorithm for $\mathsf{SVP}^{(\infty)}$.

Input: (i) A basis $\mathbf{B} = [\mathbf{b}_1, \dots \mathbf{b}_n]$ of a lattice $\mathcal{L}$, (ii) $0 < \gamma < 1$, (iii) $\xi > 1/2$, (iv) $\lambda \approx \lambda_1^{(\infty)}(\mathcal{L})$,(v) $N \in \mathbb{N}$

Output: A shortest vector of $\mathcal{L}$

1 $S \leftarrow \emptyset$;
2 **for** $i = 1$ *to* N **do**
3 $\quad \mathbf{e}_i \leftarrow_{\text{uniform}} B_n^{(\infty)}(\mathbf{0}, \xi\lambda)$;
4 $\quad \mathbf{y}_i \leftarrow \mathbf{e}_i \mod \mathcal{P}(\mathbf{B})$;
5 $\quad S \leftarrow S \cup \{(\mathbf{e}_i, \mathbf{y}_i)\}$;
6 **end**
7 $R \leftarrow n \max_i \|\mathbf{b}_i\|_\infty$;
8 **for** $j = 1$ *to* $k = \left\lceil \log_\gamma \left(\frac{\xi}{nR(1-\gamma)} \right) \right\rceil$ **do**
9 $\quad S \leftarrow \text{sieve}(S, \gamma, R, \xi)$;
10 $\quad R \leftarrow \gamma R + \xi\lambda$;
11 **end**
12 Compute the non-zero vector $\mathbf{v}_0$ in $\{(\mathbf{y}_i - \mathbf{e}_i) - (\mathbf{y}_j - \mathbf{e}_j) : (\mathbf{e}_i, \mathbf{y}_i), (\mathbf{e}_j, \mathbf{y}_j) \in S\}$ with the smallest ℓ_∞ norm ;
13 **return** $\mathbf{v}_0$;

regions, such that no two vectors in a region are at a distance greater than γR in the ℓ_∞ norm. The sieving procedure maintains an n-dimensional array with an entry corresponding to each of these $\left(1 + \left\lfloor \frac{2}{\gamma} \right\rfloor\right)^n$ regions. Each position in the array contains the description of one pair $(\mathbf{e}, \mathbf{y}) \in S$ called a center, if $\mathbf{y}$ belongs to that region. For every other vector pair $(\mathbf{e}, \mathbf{y}) \in S$ that is not a center, we find the corresponding region and hence the corresponding center $(\mathbf{e_c}, \mathbf{c})$ such that $\|\mathbf{y} - \mathbf{c}\|_\infty \leq \gamma R$. We then add $(\mathbf{e}, \mathbf{y} - \mathbf{c} + \mathbf{e_c})$ to the output list S'. Finally we return S'. It is easy to see that the number of center pairs in each iteration is at most $|C| \leq 2^{c_c n}$ where $c_c = \log\left(1 + \left\lfloor \frac{2}{\gamma} \right\rfloor\right)$.

▶ **Claim 9.** *The following two invariants are maintained in Algorithm 1:*

1. $\forall (\mathbf{e}, \mathbf{y}) \in S, \quad \mathbf{y} - \mathbf{e} \in \mathcal{L}$
2. $\forall (\mathbf{e}, \mathbf{y}) \in S, \quad ||\mathbf{y}||_\infty \leq R$

Since the length of the vectors decrease until $R > \gamma R + \xi\lambda$, the following is easy to see.

▶ **Lemma 10.** *At the end of k iterations in Algorithm 1 the length of lattice vectors* $\|\mathbf{y} - \mathbf{e}\|_\infty \leq \frac{\xi(2-\gamma)\lambda}{1-\gamma} + \frac{\gamma\xi}{n(1-\gamma)} =: R'$.

Assuming $\lambda_1^{(\infty)} \leq \lambda \leq \lambda_1^{(\infty)}(1 + 1/n)$ we get an upper bound on the number of lattice vectors of length at most R', i.e. $|B_n^{(\infty)}(R') \cap \mathcal{L}| \leq 2^{c_b n + o(n)}$, where $c_b = \log\left(1 + \left\lfloor \frac{2\xi(2-\gamma)}{1-\gamma} \right\rfloor\right)$.

The above lemma along with the invariants imply that at the beginning of step 12 in Algorithm 1 we have "short" lattice vectors with norm bounded by R'. Using the randomness in the sampling of the initial set of vectors, we want to ensure that we do not end up with all zero vectors at the end of the sieving iterations. For this we use the idea of perturbing the vectors due to Ajtai, Kumar, Sivakumar, and the current formulation by Regev [26].

Let $\mathbf{u} \in \mathcal{L}$ such that $\|\mathbf{u}\|_\infty = \lambda_1^{(\infty)}(\mathcal{L}) \approx \lambda$ (where $2 < \lambda_1^{(\infty)}(\mathcal{L}) \leq 3$), $D_1 = B_n^{(\infty)}(\xi\lambda) \cap B_n^{(\infty)}(-\mathbf{u}, \xi\lambda)$ and $D_2 = B_n^{(\infty)}(\xi\lambda) \cap B_n^{(\infty)}(\mathbf{u}, \xi\lambda)$. Define a bijection σ on $B_n^{(\infty)}(\xi\lambda)$ that maps D_1 to D_2, $D_2 \setminus D_1$ to $D_1 \setminus D_2$ and $B_n^{(\infty)}(\xi\lambda) \setminus (D_1 \cup D_2)$ to itself.

For the analysis of the algorithm, we assume that for each perturbation vector $\mathbf{e}$ chosen by our algorithm, we replace $\mathbf{e}$ by $\sigma(\mathbf{e})$ with probability $1/2$ and it remains unchanged with probability $1/2$. We call this procedure *tossing* the vector $\mathbf{e}$. Further, we assume that this replacement of the perturbation vectors happens at the step where for the first time this has any effect on the algorithm. In particular, in the sieving algorithm, after we have identified a centre $(\mathbf{e}_c, \mathbf{c})$ we apply σ on $\mathbf{e}_c$ with probability $1/2$. Then at the beginning of step 12 in Algorithm 1 we apply σ to $\mathbf{e}$ for all pairs $(\mathbf{e}, \mathbf{y}) \in S$. The distribution of $\mathbf{y}$ remains unchanged by this procedure because $\mathbf{y} \equiv \mathbf{e} \mod \mathcal{P}(\mathbf{B})$ and $\mathbf{y} - \mathbf{e} \in \mathcal{L}$. A somewhat more detailed explanation of this can be found in the following result of [11].

▶ **Lemma 11 (Theorem 4.5 in [11] (re-stated)).** *The modification outlined above does not change the output distribution of the actual procedure.*

The following lemma will help us estimate the number of vector pairs to sample at the beginning of the algorithm.

▶ **Lemma 12 (Lemma 4.7 in [11]).** *Let $N \in \mathbb{N}$ and q denote the probability that a random point in $B_n^{(\infty)}(\xi\lambda)$ is contained in $D_1 \cup D_2$. If N points $\mathbf{x}_1, \dots \mathbf{x}_N$ are chosen uniformly at random in $B_n^{(\infty)}(\xi\lambda)$, then with probability larger than $1 - \frac{4}{qN}$, there are at least $\frac{qN}{2}$ points $\mathbf{x}_i \in \{\mathbf{x}_1, \dots \mathbf{x}_N\}$ with the property $\mathbf{x}_i \in D_1 \cup D_2$.*

Using Lemma 3, it can be shown that $q \geq 2^{-c_s n}$ where $c_s = -\log\left(1 - \frac{1}{2\xi}\right)$. Thus with probability at least $1 - \frac{4}{qN}$ we have at least $2^{-c_s n}N$ pairs $(\mathbf{e}_i, \mathbf{y}_i)$ before the sieving iterations such that $\mathbf{e}_i \in D_1 \cup D_2$.

▶ **Lemma 13.** *If $N \geq \frac{2}{q}(k|C| + 2^{c_b n} + 1)$, then with probability at least $1/2$ Algorithm 1 outputs a shortest non-zero vector in $\mathcal{L}$ with respect to ℓ_∞ norm.*

Proof. Of the N vector pairs $(\mathbf{e}, \mathbf{y})$ sampled in steps 2-6 of Algorithm 1, we consider those such that $\mathbf{e} \in (D_1 \cup D_2)$. We have already seen there are at least $\frac{qN}{2}$ such pairs with probability at least $1 - \frac{4}{qN}$. We remove $|C|$ vector pairs in each of the k sieve iterations. So at step 12 of Algorithm 1 we have $N' \geq 2^{c_b n} + 1$ pairs $(\mathbf{e}, \mathbf{y})$ to process.

By Lemma 10 each of them is contained within a ball of radius R' which can have at most $2^{c_b n}$ lattice vectors. So there exists at least one lattice vector $\mathbf{w}$ for which the perturbation is in $D_1 \cup D_2$ and it appears twice in S at the beginning of step 12. With probability $1/2$ it remains $\mathbf{w}$ or with the same probability it becomes either $\mathbf{w} + \mathbf{u}$ or $\mathbf{w} - \mathbf{u}$. Thus after taking pair-wise difference at step 12 with probability at least $1/2$ we find the shortest vector. ◀

Thus, the space complexity of our algorithm is $N \cdot \mathsf{poly}(n)$, and the time complexity for the sieving step is $N \cdot \mathsf{poly}(n)$ and for computing the pairwise differences at the end is $2^{2c_b n} \cdot \mathsf{poly}(n)$, thus giving the following result.

▶ **Theorem 14.** *Let $\gamma \in (0, 1)$, and let $\xi > 1/2$. Given a full rank lattice $\mathcal{L} \subset \mathbb{Q}^n$ there is a randomized algorithm for $\mathsf{SVP}^{(\infty)}$ with success probability at least $1/2$, space complexity at most $2^{c_{space} n + o(n)}$ and running time at most $2^{c_{time} n + o(n)}$, where $c_{space} = c_s + \max(c_c, c_b)$ and $c_{time} = \max(c_{space}, 2c_b)$, where $c_c = \log\left(1 + \left\lfloor \frac{2}{\gamma} \right\rfloor\right)$, $c_s = -\log\left(1 - \frac{1}{2\xi}\right)$ and $c_b = \log\left(1 + \left\lfloor \frac{2\xi(2-\gamma)}{1-\gamma} \right\rfloor\right)$.*

3.1 Improvement using the birthday paradox

The crucial step that ensures that Algorithm 1 outputs a shortest vector in the lattice is that at step 12, we should have enough vectors to make sure that two vectors are equal (before the tossing step). Pujol and Stehle [25] observed that by the birthday paradox, we only need $2^{c_b n/2+o(n)}$ independent and identically distributed vectors to ensure this. Though their idea was described for the ℓ_2 norm, we show that the idea can be used to improve the time and space complexity of our algorithm for the ℓ_∞ norm [3]. We thus obtain the following result.

▶ **Theorem 15.** *Let $\gamma \in (0,1)$, and let $\xi > 1/2$. Given a full rank lattice $\mathcal{L} \subset \mathbb{Q}^n$ there is a randomized algorithm for SVP$^{(\infty)}$ with success probability at least $1/2$, space complexity at most $2^{c_{space}n+o(n)}$ and running time at most $2^{c_{time}n+o(n)}$, where $c_{space} = c_s + \max(c_c, \frac{c_b}{2})$ and $c_{time} = \max(c_{space}, c_b)$, where $c_c = \log\left(1+\left\lfloor\frac{2}{\gamma}\right\rfloor\right)$, $c_s = -\log\left(1-\frac{1}{2\xi}\right)$ and $c_b = \log\left(1+\left\lfloor\frac{2\xi(2-\gamma)}{1-\gamma}\right\rfloor\right)$.*

In particular for $\gamma = 0.67$ and $\xi = 0.868$ the algorithm has time and space complexity $2^{2.82n+o(n)}$.

4 Faster Approximation Algorithms

4.1 Algorithm for Approximate SVP

Notice that Algorithm 1, at the end of the sieving procedure, obtains lattice vectors of length at most $R' = \frac{\xi(2-\gamma)\lambda}{1-\gamma} + O(\lambda/n)$. So, as long as we can ensure that one of the vectors obtained at the end of the sieving procedure is non-zero, we obtain a $\tau = \frac{\xi(2-\gamma)}{1-\gamma} + o(1)$-approximation of the shortest vector. Consider a new algorithm $\mathcal{A}$ that is identical to Algorithm 1, except that Step 12 is replaced by the following:

- Find a non-zero vector $\mathbf{v}_0$ in $\{(\mathbf{y}_i - \mathbf{e}_i) : (\mathbf{e}_i, \mathbf{y}_i) \in S\}$.

We now show that if we start with sufficiently many vectors, we must obtain a non-zero vector.

▶ **Lemma 16.** *If $N \geq \frac{2}{q}(k|C|+1)$, then with probability at least $1/2$ Algorithm $\mathcal{A}$ outputs a non-zero vector in $\mathcal{L}$ of length at most $\frac{\xi(2-\gamma)\lambda}{1-\gamma} + O(\lambda/n)$ with respect to ℓ_∞ norm.*

Proof. Of the N vector pairs $(\mathbf{e}, \mathbf{y})$ sampled in steps 2-6 of Algorithm $\mathcal{A}$, we consider those such that $\mathbf{e} \in (D_1 \cup D_2)$. We have already seen there are at least $\frac{qN}{2}$ such pairs with probability at least $1 - \frac{4}{qN}$. We remove $|C|$ vector pairs in each of the k sieve iterations. So at step 12 of Algorithm 1 we have $N' \geq 1$ pairs $(\mathbf{e}, \mathbf{y})$ to process.

With probability $1/2$, $\mathbf{e}$, and hence $\mathbf{w} = \mathbf{y} - \mathbf{e}$ is replaced by either $\mathbf{w} + \mathbf{u}$ or $\mathbf{w} - \mathbf{u}$. Thus, the probability that this vector is the zero vector is at most $1/2$. ◀

We thus obtain the following:

▶ **Theorem 17.** *Let $\gamma \in (0,1)$ and $\xi > 1/2$. Given a full rank lattice $\mathcal{L} \subset \mathbb{Q}^n$ there is a randomized algorithm that, for $\tau = \frac{\xi(2-\gamma)}{1-\gamma} + o(1)$, approximates SVP$^{(\infty)}$ with success probability at least $1/2$, space and time complexity $2^{(c_s+c_c)n+o(n)}$, where $c_c = \log\left(1+\left\lfloor\frac{2}{\gamma}\right\rfloor\right)$, and $c_s = -\log\left(1-\frac{1}{2\xi}\right)$. In particular, for $\gamma = 2/3 + o(1)$, $\xi = \tau/4$, the algorithm runs in time $3^n \cdot \left(\frac{\tau}{\tau-2}\right)^n$.*

4.2 Algorithm for Approximate CVP

Given a lattice $\mathcal{L}$ and a target vector $\mathbf{t}$, let d denote the distance of the closest vector in $\mathcal{L}$ to $\mathbf{t}$. Just as in Section 3, we assume that we know the value of d within a factor of $1+1/n$. We can get rid of this assumption by using Babai's [8] algorithm to guess the value of d within a factor of 2^n, and then run our algorithm for polynomially many values of d each within a factor $1+1/n$ of the previous one.

For $\tau > 0$ define the following $(n+1)$-dimensional lattice : $\mathcal{L}' = \mathcal{L}\Big(\{(\mathbf{v},0) : \mathbf{v} \in \mathcal{L}\} \cup \{(\mathbf{t}, \tau d/2)\}\Big)$. Let $\mathbf{z}^* \in \mathcal{L}$ be the lattice vector closest to $\mathbf{t}$. Then $\mathbf{u} = (\mathbf{z}^* - \mathbf{t}, -\tau d/2) \in \mathcal{L}' \setminus (\mathcal{L} - k\mathbf{t}, 0)$ for some $k \in \mathbb{Z}$.

We sample N vector pairs $(\mathbf{e}, \mathbf{y}) \in B_n^{(\infty)}(\xi d) \times \mathcal{P}(\mathbf{B}')$, like in steps 2-6 of Algorithm 1, where $\mathbf{B}' = [(\mathbf{b}_1, 0), \ldots, (\mathbf{b}_n, 0), (\mathbf{t}, \tau d/2)]$ is a basis for $\mathcal{L}'$. Next we run a polynomial number of iterations of the sieving algorithm (sieve) to get a number of vector pairs such that $\|\mathbf{y}\|_\infty \leq R = \frac{\xi d}{1-\gamma} + o(1)$.

From Lemma 10 we have seen that after $\lceil \log_\gamma \left(\frac{\xi}{nR_0(1-\gamma)}\right) \rceil$ iterations (where $R_0 = n \cdot \max_i \|\mathbf{b}_i\|_\infty$) $R \leq \frac{\xi\gamma}{n(1-\gamma)} + \frac{\xi d}{1-\gamma}\left[1 - \frac{\xi}{nR_0(1-\gamma)}\right]$. Thus after the sieving iterations the set S' consists of vector pairs such that the corresponding lattice vector $\mathbf{v}$ has $\|\mathbf{v}\|_\infty \leq \frac{\xi d}{1-\gamma} + \xi d + c = \frac{\xi(2-\gamma)d}{1-\gamma} + o(1)$.

In order to ensure that our sieving algorithm doesn't return vectors from $(\mathcal{L}, 0) - (k\mathbf{t}, k\tau d/2)$ for some k such that $|k| \geq 2$, we choose our parameter as : $\xi < \frac{(1-\gamma)\tau}{2-\gamma} - o(1)$.

Then every vector has $\|\mathbf{v}\|_\infty < \tau d$ and so either $\mathbf{v} = \pm(\mathbf{z}' - \mathbf{t}, 0)$ or $\mathbf{v} = \pm(\mathbf{z} - \mathbf{t}, -\tau d/2)$ for some lattice vector $\mathbf{z}, \mathbf{z}' \in \mathcal{L}$. We denote this set of vectors by S''.

We need to argue that we must have at least some vectors in $S'' \setminus (\mathcal{L} \pm \mathbf{t}, 0)$ after the sieving iterations. To do so, we again use the tossing argument from Section 3. Let $\mathbf{z}^* \in \mathcal{L}$ be the lattice vector closest to $\mathbf{t}$. Then let $\mathbf{u} = (\mathbf{z}^* - \mathbf{t}, -\tau d/2) \in S'' \setminus (\mathcal{L} \pm \mathbf{t}, 0)$. Let $D_1 = B_n^{(\infty)}(\xi d) \cap B_n^{(\infty)}(-\mathbf{u}, \xi d)$ and $D_2 = B_n^{(\infty)}(\xi d) \cap B_n^{(\infty)}(\mathbf{u}, \xi d)$.

From Lemma 3, we have that the probability q that a random perturbation vector is in $D_1 \cup D_2$ is at least $2^{-c_s n} \cdot \left(1 - \frac{\tau}{4\xi}\right)$ where $c_s = -\log\left(1 - \frac{1}{2\xi}\right)$

Thus, as long as $\xi > \max(1/2, \tau/4)$, we have at least $2^{-c_s n + o(n)} N$ pairs $(\mathbf{e}_i, \mathbf{y}_i)$ before the sieving iterations such that $\mathbf{e}_i \in D_1 \cup D_2$.

Thus, using the same argument as in Section 4.1, we obtain the following:

▶ **Theorem 18.** *Let $\gamma \in (0,1)$, and for any $\tau > 1$ let $\xi > \max(1/2, \tau/4)$. Given a full rank lattice $\mathcal{L} \subset \mathbb{Q}^n$ there is a randomized algorithm that, for $\tau = \frac{\xi(2-\gamma)}{1-\gamma} + o(1)$, approximates CVP$^{(\infty)}$ with success probability at least $1/2$, space and time complexity $2^{(c_s+c_c)n+o(n)}$, where $c_c = \log\left(1 + \left\lfloor \frac{2}{\gamma} \right\rfloor\right)$ and $c_s = -\log\left(1 - \frac{1}{2\xi}\right)$. In particular, for $\gamma = 1/2 + o(1)$ and $\xi = \tau/3$, the algorithm runs in time $4^n \cdot \left(\frac{2\tau}{2\tau - 3}\right)^n$.*

5 Heuristic algorithm for SVP$^{(\infty)}$

Nguyen and Vidick [23] introduced a heuristic variant of the AKS sieving algorithm. We have used it to solve SVP$^{(\infty)}$. A brief outline of the algorithm is given in this section while a more detailed description along with the analysis is deferred to the full version [3].

The basic framework is similar to AKS, except that here we do not work with perturbation vectors. We start with a set S of uniformly sampled lattice vectors of norm $2^{O(n)}\lambda_1^{(\infty)}(\mathcal{L})$. These are iteratively fed into a sieving procedure which when provided with a list of lattice

vectors of norm, say R, will return a list of lattice vectors of norm at most γR. In each iteration of the sieve a number of vectors are identified as centres. If a vector is within distance γR from a centre, we subtract it from the centre and add the resultant to the output list. The iterations continue till the list S of vectors currently under consideration is empty. After a linear number of iterations we expect to be left with a list of very short vectors and then we output the one with the minimum norm. Here we have to ensure that we do not end up with a list of all zero-vectors much before we get these short vectors.

So we make the following assumption about the distribution of vectors at any stage of the algorithm.

▶ **Heuristic 19.** At any stage of the algorithm the vectors in $S \cap B_n^{(\infty)}(\gamma R, R)$ are uniformly distributed in $B_n^{(\infty)}(\gamma R, R) = \{\mathbf{x} \in \mathbb{R}^n : \gamma R < \|\mathbf{x}\|_\infty \leq R\}$.

In the literature, such assumption has been made for ℓ_2 norm. We have extended the same assumption to ℓ_∞ norm, because we could not find evidence that it does not hold here.

Now after each sieving iteration we get a zero vector if there is a "collision" of a vector with a centre vector. With the above assumption we can have following estimate about the expected number of collisions.

▶ **Lemma 20** ([23]). *Let p vectors are randomly chosen with replacement from a set of cardinality N. Then the expected number of different vectors picked is $N - N(1 - \frac{1}{N})^p$. So the expected number of vectors lost through collisions is $p - N + N(1 - \frac{1}{N})^p$.*

This number is negligible for $p << \sqrt{N}$. Since the expected number of lattice points inside a ball of radius $R/\lambda_1^{(\infty)}$ is $O(R^n)$, the effect of collisions remain negligible till $R/\lambda_1^{(\infty)} < |S|^{2/n}$. It can be shown that it is sufficient to take $|S| \approx (4/3)^n$, which gives $R/\lambda_1^{(\infty)} \approx 16/9$. So collisions are expected to become significant only when we already have a good estimate of $\lambda_1^{(\infty)}$, and even then collisions will imply we had a good proportion of lattice vectors in the previous iteration and thus with good probability we expect to get the shortest vector or a constant approximation of it.

Choosing $\gamma = 1 - 1/n$, our algorithm has space complexity $\left(\frac{4}{3}\right)^{n+o(n)} = 2^{0.415n+o(n)}$ and time complexity $\left(\frac{4}{3}\right)^{2n+o(n)} = 2^{0.83n+o(n)}$.

In order to improve the running time, which is mostly dictated by the number of centres, Wang et al. [27] introduced a two-level sieving procedure that improves upon the NV sieve for large n. Here in the first level we identify a set of centres C_1 and to each $\mathbf{c} \in C_1$ we associate vectors within a distance $\gamma_1 R$ from it. Now within each such $\gamma_1 R$ radius "big ball" we have another set of vectors $C_2^{\mathbf{c}}$, which we call the second-level centre. From each $\mathbf{c}' \in C_2^{\mathbf{c}}$ we subtract those vectors which are in $B_n^{(\infty)}(\mathbf{c}', \gamma_2 R)$ and add the resultant to the output list.

We have analysed this two-level sieve in the ℓ_∞ norm and also found similar improvement in the running time. For suitable choice of parameters we achieve a space and time complexity of at most $2^{0.415n+o(n)}$ and $2^{0.62n+o(n)}$ respectively.

References

1 Divesh Aggarwal, Daniel Dadush, Oded Regev, and Noah Stephens-Davidowitz. Solving the Shortest Vector Problem in 2^n time via Discrete Gaussian Sampling. In *STOC*, 2015. Full version available at `https://arxiv.org/abs/1412.7994`.

2 Divesh Aggarwal, Daniel Dadush, and Noah Stephens-Davidowitz. Solving the Closest Vector Problem in 2^ n Time–The Discrete Gaussian Strikes Again! In *Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on*, pages 563–582. IEEE, 2015.

3 Divesh Aggarwal and Priyanka Mukhopadhyay. Faster algorithms for SVP and CVP in the ℓ_∞ norm. *arXiv preprint*, 2018. arXiv:1801.02358.

4 Divesh Aggarwal and Noah Stephens-Davidowitz. Just Take the Average! An Embarrassingly Simple 2^n-Time Algorithm for SVP (and CVP). *arXiv preprint*, 2017. arXiv:1709.01535.

5 Miklós Ajtai, Ravi Kumar, and D. Sivakumar. A Sieve Algorithm for the Shortest Lattice Vector Problem. In *STOC*, pages 601–610, 2001. doi:10.1145/380752.380857.

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

7 Vikraman Arvind and Pushkar S Joglekar. Some sieving algorithms for lattice problems. In *LIPIcs-Leibniz International Proceedings in Informatics*, volume 2. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2008.

8 L. Babai. On Lovász' lattice reduction and the nearest lattice point problem. *Combinatorica*, 6(1):1–13, 1986. doi:10.1007/BF02579403.

9 Anja Becker, Léo Ducas, Nicolas Gama, and Thijs 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*, pages 10–24. Society for Industrial and Applied Mathematics, 2016.

10 Huck Bennett, Alexander Golovnev, and Noah Stephens-Davidowitz. On the Quantitative Hardness of CVP. *arXiv preprint*, 2017. arXiv:1704.03928.

11 Johannes Blömer and Stefanie Naewe. Sampling methods for shortest vectors, closest vectors and successive minima. *Theoretical Computer Science*, 410(18):1648–1665, 2009.

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

13 Léo Ducas, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS–Dilithium: Digital Signatures from Module Lattices. Technical report, IACR Cryptology ePrint Archive, 2017: 633, 2017.

14 Friedrich Eisenbrand, Nicolai Hähnle, and Martin Niemeier. Covering cubes and the closest vector problem. In *Proceedings of the twenty-seventh annual symposium on Computational geometry*, pages 417–423. ACM, 2011.

15 O. Goldreich, D. Micciancio, S. Safra, and J.-P. Seifert. Approximating shortest lattice vectors is not harder than approximating closest lattice vectors. *Information Processing Letters*, 71(2):55–61, 1999. doi:10.1016/S0020-0190(99)00083-6.

16 Ravi Kannan. Minkowski's Convex Body Theorem and Integer Programming. *Mathematics of Operations Research*, 12(3):pp. 415–440, 1987. URL: http://www.jstor.org/stable/3689974.

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

18 A. K. Lenstra, H. W. Lenstra, Jr., and L. Lovász. Factoring polynomials with rational coefficients. *Math. Ann.*, 261(4):515–534, 1982. doi:10.1007/BF01457454.

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

20 Mingjie Liu, Xiaoyun Wang, Guangwu Xu, and Xuexin Zheng. Shortest lattice vectors in the presence of gaps. *IACR Cryptology ePrint Archive*, 2011:139, 2011.

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

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

23 Phong Q Nguyen and Thomas Vidick. Sieve algorithms for the shortest vector problem are practical. *Journal of Mathematical Cryptology*, 2(2):181–207, 2008.

24 Chris Peikert et al. A decade of lattice cryptography. *Foundations and Trends® in Theoretical Computer Science*, 10(4):283–424, 2016.

25 Xavier Pujol and Damien Stehlé. Solving the Shortest Lattice Vector Problem in Time $2^{2.465n}$. *IACR Cryptology ePrint Archive*, 2009:605, 2009.

26 Oded Regev. Lecture notes on lattices in computer science, 2009.

27 Xiaoyun Wang, Mingjie Liu, Chengliang Tian, and Jingguo Bi. Improved Nguyen-Vidick heuristic sieve algorithm for shortest vector problem. In *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security*, pages 1–9. ACM, 2011.

28 Wei Wei, Mingjie Liu, and Xiaoyun Wang. Finding Shortest Lattice Vectors in the Presence of Gaps. In *Topics in Cryptology - CT-RSA 2015, The Cryptographer's Track at the RSA Conference 2015, San Francisco, CA, USA, April 20-24, 2015. Proceedings*, pages 239–257, 2015. doi:10.1007/978-3-319-16715-2_13.