THEORY OF COMPUTING, Volume 8 (2012), pp. 487–512
www.theoryofcomputing.org

Inapproximability of the Shortest Vector Problem: Toward a Deterministic Reduction

Daniele Micciancio*

Received: May 12, 2011; published: September 24, 2012.

Abstract: We prove that the Shortest Vector Problem (SVP) on point lattices is NP-hard to approximate for any constant factor under polynomial-time randomized reductions with one-sided error that produce no false positives. We also prove inapproximability for quasi-polynomial factors under the same kind of reductions running in subexponential time. Previous hardness results for SVP either incurred two-sided error, or only proved hardness for small constant approximation factors. Close similarities between our reduction and recent results on the complexity of the analogous problem for linear codes make our new proof an attractive target for derandomization, paving the road to a possible NP-hardness proof for SVP under deterministic polynomial-time reductions.

ACM Classification: F.2.2, F.1.3, G.1.6

AMS Classification: 68Q17, 52C07, 11H06, 11H31, 05B40

Key words and phrases: lattices, shortest vector problem, NP-hardness, hardness of approximation, randomized reductions

1 Introduction

Lattices are regular arrangements of points in n-dimensional Euclidean space that arise in several areas of computer science and mathematics. Two central problems in the computational study of lattices are the

*Supported in part by NSF grant CNS-1117936. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Science Foundation.

 DOI: 10.4086/toc.2012.v008a022

Shortest Vector Problem (SVP) and the Closest Vector Problem (CVP). Informally, SVP asks to find a shortest nonzero vector in a lattice. CVP is the inhomogeneous counterpart of SVP, and asks to find a lattice point closest to a given target. The approximate version of SVP asks for a nonzero lattice point having length at most γ times the shortest nonzero lattice vector, while the approximate version of CVP asks for a lattice vector whose distance to a given point is at most γ times the distance of the closest lattice vector to that point. Both SVP and CVP are hard combinatorial problems, and the asymptotically fastest known deterministic algorithm to solve them exactly runs in time $\tilde{O}(4^n)$ [22]. (See also [2] and follow-up work [24, 23, 26] for a different class of *randomized* algorithms for SVP, which are theoretically slower than [22], but admit much faster heuristic implementations [24, 23, 33].)

SVP is the more famous and widely studied problem of the two. It is also the problem for which proving strong intractability results has been most challenging. The NP-hardness of SVP (in the Euclidean norm) was conjectured by van Emde Boas in 1981 [30], but remained an outstanding open problem in computational complexity theory for almost two decades. In 1998, Ajtai [1] gave a first answer to this problem, proving that solving SVP exactly is NP-hard[1] under *randomized reductions*. This should be contrasted with the inhomogeneous problem, CVP, which has been known to be NP-hard (even under *deterministic* reductions) since the early 80s [30]. Moreover, CVP admits much simpler NP-hardness proofs than SVP [19], and it is known to be NP-hard even in its approximate version (and, as before, under *deterministic* reductions) for factors as large as $n^{1/O(\log\log n)}$ [10]. Proving the NP-hardness of SVP under *deterministic* reductions is still an open problem, even for the exact version of SVP.

Immediately following Ajtai's breakthrough result, the complexity of SVP received renewed attention, leading to several improvements, with the main goal of showing that the problem is hard even in its approximate version. In [1], Ajtai had already observed that hardness for the exact version implies weak inapproximability results for approximation factors of the form $1+1/2^{n^c}$ that rapidly approach 1 as the lattice dimension n grows. This was slightly improved by Cai and Nerurkar [7] to factors $1+1/n^c$, still approaching 1 in the limit, though at a lower rate. The first significant inapproximability result for factors bounded away from 1 was achieved by the present author [20], who proved NP-hardness for any constant factor smaller than $\sqrt{2}$ (independent of the lattice dimension). The proof in [20] has a simple and intuitive high-level structure. Specifically, the NP-hardness of SVP is proved by reduction from (a variant of) CVP, through what can be called a "homogenization process" [21]. The idea is roughly the following: if the lattice vector $\mathbf{v} \in \Lambda$ is close to the target $\mathbf{t}$, then $\mathbf{t}-\mathbf{v}$ is a short vector in the lattice generated by Λ and $\mathbf{t}$. So, one can attempt to solve a CVP instance by means of an SVP computation on an augmented lattice. This process, often used as a heuristic in cryptanalysis [17], does not work in general (see Section 2). However, [20] showed that a natural geometric gadget (essentially consisting of a lattice coset in Euclidean space with large minimum distance and many short vectors) can be used to turn this simple idea into a valid reduction from CVP to SVP. The reduction of [20] still admits a nice geometric interpretation (see Section 5 for details), and served as a starting point to obtain similar results for the analogous Minimum Distance Problem (MDP) on linear codes.

The history of the coding problem MDP, along with its inhomogeneous counterpart the Nearest

[1] All known hardness results hold not only for the SVP and CVP search problems as informally defined above, but also for the decision/promise problems naturally associated to them. (See Definitions 2.1 and 2.2.) For simplicity in this introduction we ignore the technical distinction between search and decision/promise problems, and state the known hardness results only for search SVP and CVP.

Codeword Problem (NCP), mostly mirrors that of SVP and CVP. The NP-hardness of the inhomogeneous problem, NCP, was already proved in the late 70s [6] for the exact version of the problem, and improved to NP-hardness of approximation within any constant factor (or subpolynomial factors $n^{1/O(\log\log n)}$ under subexponential time reductions)[2] in [3]. Just like for lattices, proving the NP-hardness of the homogeneous problem, MDP, took much longer. Hardness for the exact version of MDP was proved by Vardy [32] around the same time as Ajtai's discovery for SVP [1]. However, while Ajtai's reduction was randomized, Vardy [32] could prove NP-hardness of MDP under *deterministic* reductions. Building on [20], Dumer, Micciancio, and Sudan proved that MDP is NP-hard even to approximate, for any constant approximation factor. However, [11] inherited from [20] the use of randomization for the construction of the embedding gadget required by the reduction. Finally, in a surprising development, Cheng and Wan [8] showed that the probabilistic construction of the embedding gadget employed in the reduction of [11] can be derandomized, leading to the NP-hardness of approximating MDP within any constant factor under *deterministic* reductions. (The result of Cheng and Wan [8] has also been recently simplified by Khot and Austrin [4].)

Going back to lattices, the strongest inapproximability results for SVP known to date are Khot's proof [14] that SVP is NP-hard to approximate within any constant factor $O(1)$, and Haviv and Regev's proof [12] that SVP cannot be approximated within some factor $n^{1/O(\log\log n)}$ unless NP is in random subexponential time.[3] However, just like Ajtai's original proof [1], all subsequent inapproximability results for SVP [7, 20, 14, 12] employed randomization, and little progress has been made in proving NP-hardness under deterministic reductions, even for the exact version of SVP. In fact, the most recent and quantitatively strongest results [14, 12] achieve larger inapproximability factors than [20] at the cost of introducing even more randomness: while the randomized reduction in [20] had one-sided error, the hardness proofs of Khot [14] and Haviv and Regev [12] incurred two-sided error. More specifically, depending on the value of the random string, the probabilistic reduction of [14, 12] may produce both *false negatives* (i. e., map YES instances to NO instances) and *false positives* (i. e., map NO instances to YES instances). By contrast, the probabilistic reduction of [20] is guaranteed not to produce false positives regardless of the random string used in the reduction process.[4] Moreover, in addition to introducing two-sided errors, the hardness proofs of [14, 12] depart from the homogenization framework of [20], and incorporate additional probabilistic techniques (namely, the intersection of lattices with randomly chosen subspaces) that make the high-level structure of the reduction more involved and harder to derandomize. In particular, the use of randomization in [14, 12] is not restricted to the construction of a gadget with a self-contained description as in [20], but permeates the entire reduction process.

[2]The hardness of NCP under subexponential time reductions easily follows from [3] using a standard tensor product argument. It may well be the case that, similar to CVP [10], the hardness of NCP for subpolynomial approximation factors $n^{1/O(\log\log n)}$ can be proved under *deterministic polynomial-time* reductions. However, [10] only considers CVP, and makes no claim about the hardness of NCP. Extending the methods of [10] to NCP is an interesting open problem.

[3]Inapproximability results under subexponential assumptions were already given by Khot in [14], but for smaller approximation factors than [12].

[4]Randomized reductions with this one-sided error property are called *reverse unfaithful random* (RUR) reductions [13]. For simplicity, in this paper, we avoid the use of this technical term, and refer to these reductions simply as probabilistic reductions with one-sided error and no false positives. Probabilistic reductions with one-sided error and no false negatives (*faithful random* reductions, using the terminology of [13]) also occur in the study of lattice problems [15], but they are not used in this paper.

Our results We present a new, simpler proof that SVP is NP-hard to approximate within any constant factor which goes back to the geometrically appealing approach of [20] and avoids the introduction of additional probabilistic techniques from [14, 12]. In particular, we prove

- the NP-hardness of SVP for any constant approximation factor, as in [14], and
- the hardness of SVP for subpolynomial factors $n^{1/O(\log\log n)}$ under the assumption that NP is not in subexponential time, as in [14, 12],

thus matching the strongest known hardness results for SVP, but under probabilistic reductions with one-sided error. We regard our results as a partial derandomization of the reductions [14, 12] with two-sided error, and a step toward an NP-hardness proof for SVP under deterministic reductions. Randomness is used within our proof exclusively for the construction of a geometric gadget with properties similar to the one originally introduced by Micciancio in [20]. Beside the technical advantage of resulting in a reduction with one-sided error, we believe this takes us closer to a possible NP-hardness proof for SVP under deterministic reductions for the following reasons:

- In [20], Micciancio showed that a lattice gadget similar to the one used in this paper can be constructed in deterministic polynomial time under a certain (plausible but unproven) number-theoretic conjecture on the distribution of smooth numbers.[5] While proving the number-theoretic conjecture of [20] seems difficult, the result in [20] suggests that randomness is not essential to prove NP-hardness results for SVP.
- The probabilistic construction of a similar gadget for linear codes used in [11] to prove the NP-hardness of MDP has been successfully derandomized [8]. This provides hope that a derandomization of the lattice gadget employed in this paper may be possible too.
- The lattice gadget presented in this paper is constructed using techniques from the theory of linear codes, rather than the number-theoretic methods of [20]. So, the techniques in [8, 4] for the derandomization of the coding gadget of [11] may help to derandomize the construction of the lattice gadget described in this paper.

While proving the NP-hardness of (approximating) SVP under deterministic polynomial time reductions is a goal yet to be reached, we believe that our results offer a viable approach to the resolution of this outstanding open problem.

Techniques A standard method to prove hardness results within large approximation factors for lattice and coding problems is to first prove hardness for some fixed small constant factor, and then amplify the constant using some polynomial-time (or quasi-polynomial-time) transformation. For example, the tensor product of linear codes is used in [11] to amplify the NP-hardness of approximating MDP to arbitrarily large constant factors. This suggests the use of tensor products of lattices to prove the NP-hardness of SVP within large constant factors, starting from the inapproximability result of [20] for factors below $\sqrt{2}$. In fact, using the tensor product is a common theme in the sequence of papers [20, 14, 12] proving

[5] Namely, [20] conjectured that for every $\varepsilon > 0$ there is a $c \geq 1$ such that for all sufficiently large n the interval $[n, n+n^{\varepsilon}]$ contains a square-free smooth number, i. e., an integer whose prime factors are all distinct and bounded by $\log^c n$.

hardness of approximation results for SVP. Unfortunately, while the minimum distance of a linear code gets squared when one takes the tensor product of the code with itself, the same is not always true for the length of the shortest vectors in a lattice. The length of a shortest vector in the tensor product of a lattice with itself can be essentially the same as the length of shortest vectors in the original lattice (e. g., see [12, Lemma 2.4]), and this is why Micciancio [20] could not prove NP-hardness (under randomized reductions with one-sided error) of SVP within larger constant factors. Subsequent work [14, 12] went around this obstacle in various ways. Khot [14] introduced a nonstandard notion of "augmented tensor products," and used it to prove NP-hardness results for any constant approximation factor (and some subpolynomial factors under subexponential reductions) starting from a new hardness result for small constants based on BCH codes. Haviv and Regev [12] were able to prove that the lattices produced by the basic reduction of [14] behave well[6] with respect to the standard tensor product operation, leading to stronger hardness results under superpolynomial-time reductions.

We remark that the proofs in [14, 12] that the (augmented) tensor product does amplify the approximation factor are specific to the basic lattices of [14]. In this paper we revisit the general problem of amplifying the approximation factor for SVP by the standard tensor product operation, and prove that tensoring works when applied to an appropriate variant of SVP. Specifically, we introduce a new method to measure the length of the vectors in a lattice, which can be seen as a hybrid between the Euclidean length typically used for lattices and the Hamming metric used for linear codes. Specifically, the measure associated to an integer vector $\mathbf{v}$ is given by the product of the largest power of 2 (or some other fixed integer q) that divides $\mathbf{v}$ times the square root of the number of nonzero coordinates in $\mathbf{v}$. Using this measure, we define a variant of SVP and prove that it behaves well with respect to the tensor product. Then, we prove that our SVP variant is NP-hard to approximate within some constant factor, under randomized reductions with one-sided error. Tensoring immediately yields inapproximability results for SVP within larger factors, still under randomized reductions with one-sided error. Moreover, our basic NP-hardness proof—within small approximation factors—is very similar to those in [20, 11] so, as explained in the previous paragraphs, it may be more easily derandomized. We remark that the standard tensor product operation amplifies the approximation factor for any instance of the SVP variant defined in this paper, not just for the output of our basic reduction. So, the amplification method proposed here is fairly general: any proof that the SVP variant is NP-hard to approximate within some constant factor under deterministic reductions (not necessarily obtained by derandomizing the specific reduction given in this paper) immediately yields similar deterministic NP-hardness results for arbitrarily large constants.

Similarly to recent SVP NP-hardness proofs [14, 12], our reductions use BCH codes, but only within the construction of the homogenization gadget. As a historical note, the use of BCH codes in the context of proving NP-hardness results for homogeneous lattice and coding problems was first suggested in [11]. More specifically, [11] proved that codes beating the Gilbert-Varshamov bound can be used to build geometric gadgets similar to the one of [20], and mentioned Reed-Solomon, Algebraic-Geometry and BCH codes as examples of codes beating this bound. BCH codes were later used by [14, 12] to prove the hardness of SVP for any constant approximation factor and beyond, but in an ad-hoc manner, i. e., without explicitly connecting them to the general framework of [20, 11]. Our work perhaps offers a more intuitive explanation of why BCH codes are useful in proving inapproximability results for SVP.

[6]Informally, a lattice Λ behaves well with respect to the tensor product if the minimum distance of the tensor product of Λ with itself is essentially the square of the minimum distance of Λ. See Theorem 3.5 for a formal definition.

Organization The rest of the paper is organized as follows. In Section 2 we give some background about lattices, codes, and the homogenization framework of [20]. In Section 3 we describe our basic techniques used to amplify inapproximability results for SVP via tensoring. In Section 4 we give a construction of very dense lattices with large minimum distance that behave well with respect to the tensor product operation. In Section 5 we give our main NP-hardness proof for SVP under nonuniform reductions with one-sided error. We chose to first present our result as a nonuniform reduction to make the reduction and analysis as simple as possible and self-contained. However, the nonuniformity of the advice is not used in any essential way in our proof, and in Section 6 we use combinatorial techniques from [1, 20] to replace the nonuniform advice with a uniformly chosen random string, leading to NP-hardness results under randomized reductions with one-sided error.

2 Background

In this section we give some standard background on computational complexity, lattices, and some combinatorial results used in this paper. The reader familiar with computational complexity can safely skip most of this section without much loss.

We use $\mathbb{R}$, $\mathbb{Z}$, and 2^A to denote the set of the real numbers, the set of the integers, and the power set of an arbitrary set A, respectively. We use standard asymptotic notation and write $f = O(g)$ or $g = \Omega(f)$ if $|f(n)/g(n)|$ is bounded, and $f = o(g)$ or $g = \omega(f)$ if $\lim_{n\to\infty} |f(n)/g(n)| = 0$. We call a function $f : \mathbb{Z} \to \mathbb{R}$ *negligible* if $f(n) = n^{-\omega(1)}$.

Computational complexity As is standard in the study of computational complexity, we formulate algorithmic problems on lattices as decision (or, more precisely, promise) problems. A *promise problem* is a pair of disjoint languages $\Pi_{\text{YES}}, \Pi_{\text{NO}} \subseteq \{0,1\}^*$. The elements of Π_{YES} and Π_{NO} are called YES instances and NO instances, respectively. The computational problem associated to $(\Pi_{\text{YES}}, \Pi_{\text{NO}})$ is, given a string $w \in \Pi_{\text{YES}} \cup \Pi_{\text{NO}}$, determine if $w \in \Pi_{\text{YES}}$ or not. (If $w \notin \Pi_{\text{YES}} \cup \Pi_{\text{NO}}$, then any answer is acceptable.) Decision problems correspond to the special case where $\Pi_{\text{YES}} \cup \Pi_{\text{NO}} = \{0,1\}^*$, and the promise $w \in \Pi_{\text{YES}} \cup \Pi_{\text{NO}}$ is trivially satisfied. When studying lattices and other computational problems, we assume that common mathematical objects (integers, matrices, vectors, pairs, etc.) are represented as binary strings in some standard way.

A *reduction* between two promise problems $(\Pi_{\text{YES}}, \Pi_{\text{NO}})$ and $(\Sigma_{\text{YES}}, \Sigma_{\text{NO}})$ is a function $f \colon \{0,1\}^* \to \{0,1\}^*$ such that $f(\Pi_{\text{YES}}) \subseteq \Sigma_{\text{YES}}$ and $f(\Pi_{\text{NO}}) \subseteq \Sigma_{\text{NO}}$. In this paper we also use randomized and nonuniform reductions, defined below. Both randomized and nonuniform reductions are specified by a function $f(x,y)$ that takes two inputs, a regular input x and an auxiliary input y. The auxiliary input y is called *advice* in the case of nonuniform reductions, and *randomness* in the case of randomized reductions. The complexity of the reduction is the time complexity of computing f, expressed as a function of the length of the first input x alone. So, we say that $f(x,y)$ is a polynomial-time reduction if there is an algorithm that on input x and y, outputs $f(x,y)$ in time $|x|^{O(1)}$.

A function $f(x,y)$ is a *nonuniform reduction* from $(\Pi_{\text{YES}}, \Pi_{\text{NO}})$ to $(\Sigma_{\text{YES}}, \Sigma_{\text{NO}})$ if for any input length n, there is an advice string y such that $f(\Pi_{\text{YES}} \cap \{0,1\}^n, y) \subseteq \Sigma_{\text{YES}}$ and $f(\Pi_{\text{NO}} \cap \{0,1\}^n, y) \subseteq \Sigma_{\text{NO}}$. When studying reductions between concrete computational problems (e. g., lattice problems in this paper) it is common to have the advice string depend not on the input length $|x|$, but on some other parameter of interest associated with the size of the input, e. g., the lattice dimension. (See the next paragraph for background about lattices.) This is without loss of generality, as the lattice dimension is always at most $|x|$, and therefore y can include a piece of advice for each dimension $n \le |x|$ while incurring only a polynomial blow-up in the size of the advice.

A function $f(x,y)$ is a *randomized reduction* from $(\Pi_{\mathrm{YES}}, \Pi_{\mathrm{NO}})$ to $(\Sigma_{\mathrm{YES}}, \Sigma_{\mathrm{NO}})$ if

$$\Pr_y\{f(x,y) \in \Sigma_{\mathrm{YES}}\} \ge 2/3 \text{ for all } x \in \Pi_{\mathrm{YES}}, \quad \text{and} \quad \Pr_y\{f(x,y) \in \Sigma_{\mathrm{NO}}\} \ge 2/3 \text{ for all } x \in \Pi_{\mathrm{NO}},$$

where both probabilities are computed over the uniformly random choice of $y \in \{0,1\}^{r(|x|)}$ among the set of all binary strings of a given length $r(|x|)$. (One can always assume that the number $r(|x|)$ of random bits used by the reduction is bounded by the reduction running time $t(|x|)$.) Randomized reductions between concrete computational problems often assume that the randomness y is not just a uniformly random binary string, but it is the binary representation of a structured object chosen according to some efficiently sampleable (but not necessarily uniform) distribution. Formally, the randomized reduction uses a uniformly random $y \in \{0,1\}^{r(|x|)}$ as a seed to run the efficient sampling algorithm that produces (or approximates up to negligible errors) the desired distribution.

Notice that nonuniform and randomized reductions have *two-sided error*: depending on the value of the advice or randomness y, they can produce both *false positives* (i. e., map $x \in \Pi_{\mathrm{NO}}$ to $f(x,y) \in \Sigma_{\mathrm{YES}}$) and *false negatives* (i. e., map $x \in \Pi_{\mathrm{YES}}$ to $f(x,y) \in \Sigma_{\mathrm{NO}}$). In this paper we consider reductions with *one-sided error*, that may produce false negatives, but are guaranteed not to produce false positives. Formally, nonuniform reductions with one-sided error and no false positives satisfy $f(\Pi_{\mathrm{NO}} \cap \{0,1\}^n, y) \subseteq \Sigma_{\mathrm{NO}}$ for *all* advice strings y. Similarly, randomized reductions with one-sided error and no false positives satisfy $\Pr_y\{f(x,y) \in \Sigma_{\mathrm{NO}}\} = 1$ for all $x \in \Pi_{\mathrm{NO}}$.

Lattices The n-dimensional Euclidean space is denoted $\mathbb{R}^n$. A *lattice* in $\mathbb{R}^n$ is the set of all integer combinations

$$\Lambda = \left\{ \sum_{i=1}^{k} x_i \mathbf{b}_i \colon x_i \in \mathbb{Z} \right\}$$

of k linearly independent vectors $\mathbf{b}_1, \ldots, \mathbf{b}_k$ in $\mathbb{R}^n$ ($n \ge k$). The set of vectors $\mathbf{b}_1, \ldots, \mathbf{b}_k$ is called a lattice *basis*, and the integer k is the lattice *rank* or *dimension*. When $k = n$, Λ is called a *full-rank* or *full-dimensional* lattice. A basis can be compactly represented by the matrix $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_k] \in \mathbb{R}^{n \times k}$ having the basis vectors as columns. The lattice generated by $\mathbf{B}$ is denoted $\mathcal{L}(\mathbf{B})$. Notice that $\mathcal{L}(\mathbf{B}) = \{\mathbf{Bx} \colon \mathbf{x} \in \mathbb{Z}^k\}$, where $\mathbf{Bx}$ is the usual matrix-vector multiplication. The *determinant* of a lattice $\mathcal{L}(\mathbf{B})$ is the volume of the parallelepiped spanned by the basis vectors $\mathbf{B}$, and does not depend on the choice of the basis $\mathbf{B}$. When $\mathbf{B}$ is a square matrix, it equals the absolute matrix determinant $\det(\mathcal{L}(\mathbf{B})) = |\det(\mathbf{B})|$. More generally, for nonsquare bases, $\det(\mathcal{L}(\mathbf{B})) = \sqrt{\det(\mathbf{B}^T\mathbf{B})}$, where $\mathbf{B}^T$ is the *matrix transpose* of $\mathbf{B}$.

Lattice problems can be defined with respect to any norm, but the Euclidean norm $\|\mathbf{x}\| = \sqrt{\sum_i x_i^2}$ is the most common, and the one we focus on in this paper. We recall that the Euclidean norm is, in a technical sense, the one for which lattice problems are algorithmically easiest, and hardness results for other norms can be obtained via norm embeddings [27]. The *minimum distance* $\lambda(\Lambda)$ of a lattice Λ is the smallest distance between any two distinct lattice points and equals the length of the shortest nonzero lattice vectors:

$$\lambda(\Lambda) = \min\{\|\mathbf{x}-\mathbf{y}\| : \mathbf{x} \neq \mathbf{y} \in \Lambda\} = \min\{\|\mathbf{x}\| : \mathbf{x} \in \Lambda, \mathbf{x} \neq \mathbf{0}\}.$$

For any vector $\mathbf{x} \in \mathbb{R}^n$ and real r, let $\mathcal{B}(\mathbf{v}, r) = \{\mathbf{w} \in \mathbb{R}^n : \|\mathbf{v}-\mathbf{w}\| \leq r\}$ be the ball of radius r centered at $\mathbf{v}$. When the ball is centered around the origin $\mathbf{v} = \mathbf{0}$, we simply write $\mathcal{B}(r)$.

When discussing computational issues related to lattices, it is customary to assume that the lattices are represented by a basis matrix $\mathbf{B}$ and that the basis $\mathbf{B}$ has integer entries. This is without much loss of generality as real lattices can be approximated by rational ones with arbitrarily high precision, and rational lattices are easily mapped to integer ones simply by scaling them. In any case, as in this paper we are concerned with (worst-case) hardness results, restricting the definitions to integer lattices only makes the results stronger. We study the decisional (length/distance estimation) variants of SVP and CVP as defined below.

Definition 2.1. The promise problem GapSVP_γ (where $\gamma \geq 1$ may be a function of the lattice dimension) is defined as follows. Instances are pairs $(\mathbf{B}, d)$, where $\mathbf{B} \in \mathbb{Z}^{n\times k}$ is a lattice basis and d is a positive number such that

1. $(\mathbf{B}, d)$ is a YES instance if $\lambda(\mathcal{L}(\mathbf{B})) \leq d$, i. e., $\|\mathbf{Bx}\| \leq d$ for some $\mathbf{x} \in \mathbb{Z}^k \setminus \{\mathbf{0}\}$;
2. $(\mathbf{B}, d)$ is a NO instance if $\lambda(\mathcal{L}(\mathbf{B})) > \gamma \cdot d$, i. e., $\|\mathbf{Bx}\| > \gamma \cdot d$ for all $\mathbf{x} \in \mathbb{Z}^k \setminus \{\mathbf{0}\}$.

Definition 2.2. The promise problem GapCVP_γ (where $\gamma \geq 1$ may be a function of the lattice dimension) is defined as follows. Instances are triples $(\mathbf{B}, \mathbf{y}, d)$, where $\mathbf{B} \in \mathbb{Z}^{n\times k}$ is a lattice basis, $\mathbf{y} \in \mathbb{Z}^n$ is a vector, and d is a positive number such that

1. $(\mathbf{B}, \mathbf{y}, d)$ is a YES instance if $\|\mathbf{y}-\mathbf{Bx}\| \leq d$ for some $\mathbf{x} \in \mathbb{Z}^k$;
2. $(\mathbf{B}, \mathbf{y}, d)$ is a NO instance if $\|\mathbf{y}-\mathbf{Bx}\| > \gamma \cdot d$ for all $\mathbf{x} \in \mathbb{Z}^k$.

We remark that any algorithm that (approximately) solves SVP in its standard formulation (i. e., given a lattice, finds a nonzero lattice vector of length within a factor γ from the shortest) can immediately be used to solve GapSVP_γ. So, proving hardness results for GapSVP_γ implies hardness of approximating the standard SVP as well. The same observation applies to CVP and GapCVP_γ. However we remark that the converse is not known to be true: giving a reduction from approximate SVP to GapSVP_γ (or from approximate CVP to GapCVP_γ) is an important open problem in the complexity of lattice problems.

Linear codes Some of our constructions rely on techniques from the study of linear codes. For any finite field $\mathbb{F}$, and finite-dimensional vector space $\mathbb{F}^n$ over $\mathbb{F}$, a *linear code* of *block length* n and *dimension* k is a k-dimensional linear subspace of $\mathbb{F}^n$. As for lattices, linear codes are usually represented by a *generator matrix* $\mathbf{C} \in \mathbb{F}^{n\times k}$, which is a basis for the code $\mathcal{C}(\mathbf{C}) = \{\mathbf{Cx} \colon \mathbf{x} \in \mathbb{F}^k\} \subseteq \mathbb{F}^n$. The difference $n-k$ is called the *co-dimension* of the code. The *Hamming weight* of a vector $\mathbf{v} \in \mathbb{F}^n$ is the number $\|\mathbf{v}\|_H$ of nonzero coordinates of $\mathbf{v}$. The *minimum distance* of a linear code $\mathcal{C} \subseteq \mathbb{F}^n$ is the smallest Hamming weight of a nonzero vector in the code $\min\{\|\mathbf{v}\|_H \colon \mathbf{v} \in \mathcal{C} \setminus \{\mathbf{0}\}\}$. In this paper, we are primarily interested in *binary* linear codes, i. e., linear codes over the field $\mathbb{F}_2 = \{0, 1\}$ with two elements. A binary linear code with block length n, dimension k, and minimum distance d is usually denoted $\mathcal{C}[n, k, d]_2$. In Section 4 we build a very dense lattice starting from the family of *extended primitive narrow-sense binary BCH codes*. For brevity, we refer to these codes just as *extended BCH codes*. Extended BCH codes can be defined for any block length $m = 2^\kappa$ that is a power of 2. The properties of these codes needed in this paper are summarized in the following theorem. For completeness, we also include a brief sketch of the construction and analysis of extended BCH codes, and refer the reader to [25, Chapter 1, Section 7] for details.

Theorem 2.3. *For any m being a power of two, and any even positive integer $h \le m$, the extended BCH code $\mathrm{EBCH}_h^m[m,k,d]$ has minimum distance $d \ge h$ and co-dimension $m-k \le (\log_2 m)\cdot(h/2-1)+1$. Moreover, these codes satisfy $\mathbb{F}_2^m = \mathrm{EBCH}_0^m \supseteq \mathrm{EBCH}_2^m \supseteq \cdots \supseteq \mathrm{EBCH}_m^m$. Generator matrices for these codes can be constructed in time polynomial in m.*

Proof. Extended BCH codes of block length $m = 2^\kappa$ are obtained by appending a parity check bit to a BCH code of block length $n = m-1$. BCH codes are polynomial codes, i. e., they can be described algebraically as the set of all (coefficient vectors of) polynomials of degree less than n that are divisible by a given generating polynomial $g(X) \in \mathbb{F}_2[X]$. The co-dimension of a polynomial code equals the degree $n-k = \deg(g)$ of the generating polynomial. Let α be a generator of the multiplicative group of $\mathbb{F}_m$, the finite field with $m = 2^\kappa$ elements. A basic fact in the theory of polynomial codes (BCH bound, see [25, Chapter 1, Theorem 6.1]) is that if $g(\alpha^i) = 0$ for t consecutive powers of α, then the polynomial code generated by $g(X)$ has minimum distance at least $t+1$.

For any even $h \le m$, the BCH code with designed minimum distance $h-1 \le m-1$ is the polynomial code generated by the least common multiple $g_h(X)$ of the minimal polynomials

$$p_1(X), p_3(X), \ldots, p_{h-3}(X)$$

of the first $h/2-1$ odd powers $\alpha^1, \alpha^3, \ldots, \alpha^{h-3}$ of the primitive element α. Notice that for any even power α^{2j}, it holds that $g_h(\alpha^{2j}) = (g_h(\alpha^j))^2$ because squaring is a linear operation in the polynomial ring $\mathbb{F}_m[X]$ (as a vector space over $\mathbb{F}_m$). So, $g_h(\alpha^j) = 0$ for all $j = 1, \ldots, h-2$, and the minimum distance of the BCH code is at least $h-1$. The extended BCH code $\mathrm{EBCH}_h^m[m,k,d]_2$ is obtained by appending a parity check bit to the code generated by $g_h(X)$. Since $h-1$ is odd, appending a parity check bit increases the (designed) minimum distance of the code to $d \ge h$. The block length and co-dimension also increase by 1, while the dimension of the code remains the same. Since the degree of each minimal polynomial p_j is at most κ, the degree of g_h is bounded by $\kappa \cdot (h/2-1)$, and the co-dimension of $\mathrm{EBCH}_h^m[m,k,d]$ is at most $m-k \le \kappa(h/2-1)+1$. Notice that for any $h \le h'$, $g_{h'}$ is a multiple of g_h, and therefore $\mathrm{EBCH}_h^m \supseteq \mathrm{EBCH}_{h'}^m$. □

Tensor products For any two matrices $\mathbf{B}^{(1)} \in \mathbb{R}^{n_1 \times k_1}$ and $\mathbf{B}^{(2)} \in \mathbb{R}^{n_2 \times k_2}$, define the *Kronecker product* $\mathbf{B} = \mathbf{B}^{(1)} \otimes \mathbf{B}^{(2)} \in \mathbb{R}^{n_1 n_2 \times k_1 k_2}$ as the matrix with entries

$$b_{i,j} = b^{(1)}_{i_1,j_1} \cdot b^{(2)}_{i_2,j_2}, \quad \text{where } i = (i_1-1)\cdot n_2 + i_2 \text{ and } j = (j_1-1)\cdot k_2 + j_2,$$

for $i_1 = 1, \ldots, n_1$, $i_2 = 1, \ldots, n_2$, $j_1 = 1, \ldots, k_1$ and $j_2 = 1, \ldots, k_2$. Informally, $\mathbf{B}^{(1)} \otimes \mathbf{B}^{(2)}$ is the block matrix obtained by replacing each entry $b^{(1)}_{i_1,j_1}$ of $\mathbf{B}^{(1)}$ with the matrix $b^{(1)}_{i_1,j_1} \cdot \mathbf{B}^{(2)}$. The *tensor product* of lattices $\mathcal{L}(\mathbf{B}^{(1)})$ and $\mathcal{L}(\mathbf{B}^{(2)})$ is the k_1k_2-dimensional lattice $\mathcal{L}(\mathbf{B}^{(1)} \otimes \mathbf{B}^{(2)})$ in n_1n_2-dimensional space generated by the Kronecker product of the two basis matrices. Identifying the set $\mathbb{R}^{n_1n_2}$ of n_1n_2-dimensional vectors with the set $\mathbb{R}^{n_1\times n_2}$ of $n_1 \times n_2$ matrices in the obvious way, the tensor product of two lattices can be conveniently defined as the set of all matrices

$$\mathcal{L}(\mathbf{B}^{(1)} \otimes \mathbf{B}^{(2)}) = \{\mathbf{B}^{(1)}\mathbf{X}(\mathbf{B}^{(2)})^T \mid \mathbf{X} \in \mathbb{Z}^{k_1\times k_2}\}.$$

The tensor product of two linear codes is defined similarly. As mentioned in the introduction, the tensor product operation can be used to amplify hardness results for certain coding and lattice problems to large approximation factors. For example, if $\mathcal{C}$ is a linear code with minimum distance d, then the product code $\mathcal{C} \otimes \mathcal{C}$ has minimum distance d^2. So, if the minimum distance of $\mathcal{C}$ is hard to approximate within a factor γ, then the minimum distance of $\mathcal{C} \otimes \mathcal{C}$ is also hard to approximate within a factor γ^2. Similar amplification results are also possible for NCP and CVP. However, this method to amplify the approximation factor of a problem does not work for GapSVP. It is easy to prove that $\lambda(\Lambda_1 \otimes \Lambda_2) \le \lambda(\Lambda_1) \cdot \lambda(\Lambda_2)$ for any two lattices Λ_1 and Λ_2, and therefore $\lambda(\Lambda \otimes \Lambda) \le \lambda(\Lambda)^2$. However, in general $\lambda(\Lambda \otimes \Lambda)$ can be much smaller than $\lambda(\Lambda)^2$. (E. g., see [12, Lemma 2.4].) Lattices Λ_1 for which $\lambda(\Lambda_1 \otimes \Lambda_2) = \lambda(\Lambda_1) \cdot \lambda(\Lambda_2)$ for every lattice Λ_2 are called "E-type" lattices [16], and are special in this regard.

We will prove the NP-hardness of GapSVP by reduction from the following NP-hard variant of CVP.

Definition 2.4. The promise problem TensorCVP$_\gamma$ (where $\gamma \ge 1$ may be a function of the lattice dimension) is defined as follows. Instances are triples $(\mathbf{B}, \mathbf{y}, t)$ where $\mathbf{B} \in \mathbb{Z}^{n\times k}$ is a lattice basis, $\mathbf{y} \in \mathbb{Z}^n$ is a vector, and t is a positive number such that

- $(\mathbf{B}, \mathbf{y}, t)$ is a YES instance if $\|\mathbf{y} - \mathbf{B}\mathbf{x}\| \le t$ for some $\mathbf{x} \in \{0,1\}^k$;
- $(\mathbf{B}, \mathbf{y}, t)$ is a NO instance if $\sqrt{\|\mathbf{y} - \mathbf{B}\mathbf{x}\|_H} > \gamma t$ for all $\mathbf{x} \in \mathbb{R}^k$.

TensorCVP differs from CVP as follows. In the YES instances, the target is required to be close to a *binary* combination of the basis vectors. In the NO instances, the target is required to be far in *Hamming distance* from the entire *linear space* spanned by the lattice. TensorCVP is a fairly standard NP-hard variant of CVP, similar to those used in previous works on the computational complexity of lattice problems [20, 14, 12]. We call this CVP variant TensorCVP because we will use it to prove the NP-hardness of TensorSVP, the variant of SVP defined in Definition 3.4, which is closely related to the use of the tensor product to amplify the approximation factor. The NP-hardness of TensorCVP is proved by reduction from the *exact set cover* problem. Remember that an instance of *exact set cover* consists of a collection of sets $S_1, \ldots, S_n \subseteq \{1, \ldots, u\}$ and an integer $t \le n$. A cover C is a subcollection $C \subseteq \{1, \ldots, n\}$ such that $\bigcup_{i\in C} S_i = \{1, \ldots, u\}$. A cover is *exact* if the sets S_i ($i \in C$) in the cover are pairwise disjoint, i. e., $\{S_i\}_{i\in C}$ is a *partition* of $\{1, \ldots, u\}$. When reducing set cover problems to lattice problems it is convenient to represent the collection $\{S_1, \ldots, S_n\}$ as a matrix $\mathbf{S} = [\mathbf{s}_1, \ldots, \mathbf{s}_n] \in \{0,1\}^{u\times n}$ where the columns $\mathbf{s}_i$ are the indicator vectors of the sets S_i. Using matrix notation, a cover of size t is represented by a binary vector $\mathbf{c} \in \{0,1\}^n$ of Hamming weight t such that $\mathbf{S}\mathbf{c} \ge \mathbf{1}$, where $\mathbf{1}$ is the all-ones vector and the inequality holds componentwise. The cover is exact if $\mathbf{S}\mathbf{c} = \mathbf{1}$.

Definition 2.5. For any $\gamma \ge 1$, an instance of the γ-approximate exact set cover problem is a pair $(\mathbf{S}, t)$ with $\mathbf{S} \in \{0,1\}^{u\times n}$ and $t \in \{1, \ldots, n\}$ such that

- $(\mathbf{S},t)$ is a YES instance if there is an exact cover of size at most t, i. e., a binary vector $\mathbf{c} \in \{0,1\}^n$ of Hamming weight $\|\mathbf{c}\|_H \leq t$ such that $\mathbf{Sc} = \mathbf{1}$;

- $(\mathbf{S},t)$ is a NO instance if all covers have size bigger than γt, i. e., all binary vectors $\mathbf{c} \in \{0,1\}^n$ such that $\mathbf{Sc} \geq \mathbf{1}$ have Hamming weight $\|\mathbf{c}\|_H > \gamma t$.

The NP-hardness of TensorCVP$_\gamma$ easily follows from the following NP-hardness result for exact set cover proved in [5].

Theorem 2.6. *The exact set cover problem is NP-hard to approximate within any constant factor $\gamma \geq 1$.*

The NP-hardness of TensorCVP$_\gamma$ is already implicit in [3], and, with minor variations, it has been used in many previous works on the complexity of GapSVP [20, 14, 12]. Here we give a slightly simpler proof than the one originally given in [3] and commonly reported in the literature [14, 12], that avoids the introduction of auxiliary variables and large constants.

Theorem 2.7. *For any constant $\gamma \geq 1$,* TensorCVP$_\gamma$ *is NP-hard.*

Proof. We use the fact that exact set cover is NP-hard for any constant approximation factor γ (Theorem 2.6), and we reduce it to TensorCVP$_{\sqrt{\gamma}}$. On an input that is an exact set cover instance $(\mathbf{S},t)$, the reduction produces a TensorCVP$_{\sqrt{\gamma}}$ instance $(\mathbf{B},\mathbf{y},\sqrt{t})$ where $\mathbf{B} \in \mathbb{Z}^{n\times k}$ is a basis for the lattice of all integer vectors $\mathbf{v}$ such that $\mathbf{Sv} = \mathbf{0}$, and $\mathbf{y} \in \mathbb{Z}^n$ is an arbitrary integer solution to $\mathbf{Sy} = \mathbf{1}$. (Both $\mathbf{B}$ and $\mathbf{y}$ can be efficiently computed using linear algebra. If no solution $\mathbf{y}$ exists, then $(\mathbf{S},t)$ is necessarily a NO instance and the reduction can output an arbitrary NO instance of TensorCVP$_{\sqrt{\gamma}}$.) We need to prove that the reduction is correct.

If $(\mathbf{S},t)$ is a YES instance, then there is an exact cover of size t, i. e., a vector $\mathbf{c} \in \{0,1\}^n$ of Hamming weight $\|\mathbf{c}\|_H \leq t$ such that $\mathbf{Sc} = \mathbf{1}$. It follows that $\mathbf{S}(\mathbf{y}-\mathbf{c}) = \mathbf{0}$, i. e., $\mathbf{y}-\mathbf{c}$ is a lattice vector. Moreover this lattice vector is within distance $\|\mathbf{y}-(\mathbf{y}-\mathbf{c})\| = \|\mathbf{c}\| \leq \sqrt{t}$ from $\mathbf{y}$, proving that $(\mathbf{B},\mathbf{y},\sqrt{t})$ is a YES instance of TensorCVP$_{\sqrt{\gamma}}$. Now assume $(\mathbf{S},t)$ is a NO instance. Notice that for any $\mathbf{v}$ in the linear span of $\mathbf{B}$, $\mathbf{S}(\mathbf{y}-\mathbf{v}) = \mathbf{Sy} = \mathbf{1}$. So, the nonzero coordinates of $\mathbf{y}-\mathbf{v}$ form a set cover. It follows that $\mathbf{y}-\mathbf{v}$ must have more than γt nonzero coordinates, i. e., $\sqrt{\|\mathbf{y}-\mathbf{v}\|_H} > \sqrt{\gamma t}$. □

The homogenization framework We briefly recall the framework of [20] to prove hardness results for GapSVP. Let $(\mathbf{B},\mathbf{y})$ be a CVP instance. A common heuristic to find a lattice vector $\mathbf{Bx}$ closest to $\mathbf{y}$ is to search for a short vector in the augmented lattice $\mathcal{L}([\mathbf{B},\mathbf{y}])$. However, this simple heuristic does not work in general (even if one can solve SVP exactly), and it only yields a reduction from Bounded Distance Decoding (BDD) to the unique Shortest Vector Problem (uSVP), two restricted versions of CVP and SVP [18]. For the general CVP and SVP, there are two different ways in which this approach may fail:

- A shortest nonzero vector in $\mathcal{L}([\mathbf{B},\mathbf{y}])$ may be of the form $\mathbf{Bx}+c\cdot\mathbf{y}$ with $|c| \geq 2$. This yields a lattice vector $\mathbf{Bx}$ close to a *multiple* of the original target $\mathbf{y}$.

- A shortest nonzero vector in $\mathcal{L}([\mathbf{B},\mathbf{y}])$ may be of the form $\mathbf{Bx}$. This will be the case if the distance of the target $\mathbf{y}$ from the lattice $\mathcal{L}(\mathbf{B})$ is larger than $\lambda(\mathcal{L}(\mathbf{B}))$.

In the context of proving the NP-hardness of SVP, the first problem is easily solved by reducing from a variant of CVP (like TensorCVP, see Definition 2.4) where either the target is close to the lattice, or all of its nonzero (integer) multiples are far from it. The second problem is more fundamental, and also arises in the context of proving similar results for linear codes [11]. Building on techniques from [1], Micciancio [20] solved this problem essentially by embedding $\mathbf{B}$ and $\mathbf{y}$ into a higher dimensional space to obtain a new lattice $\mathcal{L}(\mathbf{B}')$ and target vector $\mathbf{t}'$ such that

- if $\mathbf{y}$ is close to the lattice $\mathcal{L}(\mathbf{B})$, the embedded target $\mathbf{y}'$ is still close to the lattice $\mathcal{L}(\mathbf{B}')$, and
- the embedding operation increases the minimum distance of the lattice $\mathcal{L}(\mathbf{B})$, so that the distance of $\mathbf{y}'$ from $\mathcal{L}(\mathbf{B}')$ is strictly smaller than $\lambda(\mathcal{L}(\mathbf{B}'))$.

This transformation ensures that the shortest vectors in $\mathcal{L}([\mathbf{B}',\mathbf{y}'])$ are not in $\mathcal{L}(\mathbf{B}')$, and therefore must necessarily make use of the target vector $\mathbf{y}'$. In [20], it is shown that such a transformation can be easily carried out using a geometric gadget consisting of a lattice coset $\mathcal{L}(\mathbf{L})-\mathbf{s}$ with large minimum distance $\lambda(\mathcal{L}(\mathbf{L}))$ and many short vectors $(\mathcal{L}(\mathbf{L})-\mathbf{s})\cap\mathcal{B}(r)$. (Specifically, the length bound r on these vectors should be strictly smaller than the minimum distance of $\mathcal{L}(\mathbf{L})$ by a constant factor.) Moreover, if $\mathcal{L}(\mathbf{L})$ is sufficiently dense (i. e., if its determinant is not too big), then an appropriate coset is guaranteed to exist and can be probabilistically found by choosing $\mathbf{s}$ as a random short vector.

In [20] a gadget of this type is constructed using techniques from elementary number theory, which are less "combinatorial" than the coding theory tools used in the NP-hardness proof of [14], and arguably harder to derandomize. In this paper we give an alternative and more refined construction of Micciancio's geometric gadget. Similarly to the proofs in [14, 12], we rely on tools from coding theory, namely the construction of BCH codes (see Theorem 2.3) rather than number-theoretic methods, and make extensive use of integer vectors with all-even coordinates, an element already present in [14].

Beside its potential for easier derandomization, the new construction, which combines lattice and coding elements, has the advantage of behaving well with respect to the tensor product of lattices.

Our nonuniform reduction from TensorCVP to GapSVP, similarly to previous work [1, 20], uses the following combinatorial result, due to Vapnik and Chervonenkis [31], Sauer [28], and Shelah [29] and commonly referred to as "Sauer's lemma."

Lemma 2.8 (Sauer's Lemma)**.** *Let M be a set of size m, and let $\mathcal{A}\subseteq 2^M$ be an arbitrary collection of subsets of M. For any integer $k\geq 1$ such that $|\mathcal{A}|>\sum_{i=0}^{k-1}\binom{m}{i}$, there exists a subset $T\subset M$ of size $|T|=k$ that is shattered by $\mathcal{A}$, i. e., the restriction $\mathcal{A}|_T=\{A\cap T : A\in\mathcal{A}\}$ equals 2^T.*

We restate Sauer's lemma in matrix language, interpreting the elements of $\mathcal{A}$ as (the indicator functions of) subsets of $M=\{1,\dots,m\}$, and letting $\mathbf{T}\in\{0,1\}^{k\times m}$ be the projection matrix corresponding to the set of coordinates $T\subset M$.

Lemma 2.9 (Sauer's Lemma restated)**.** *Let m be a positive integer, and $\mathcal{A}\subset\{0,1\}^m$ an arbitrary set of m-dimensional binary vectors. If $|\mathcal{A}|>\sum_{i=0}^{k-1}\binom{m}{i}$, then there exists a matrix $\mathbf{T}\in\{0,1\}^{k\times m}$ such that $\{0,1\}^k\subseteq\{\mathbf{T}\mathbf{z}\colon \mathbf{z}\in\mathcal{A}\}$.*

3 Basic techniques

Central to our construction and hardness results is a new method to measure the length of a vector that is, in a sense, a hybrid between the Euclidean norm and the Hamming metric. The definition is parametrized by an integer q that we will later set to $q = 2$.

Definition 3.1. For any integer vector $\mathbf{x}$, let $\mathrm{pow}_q(\mathbf{x})$ be the largest power of q that divides $\mathbf{x}$, and define $\tau_q(\mathbf{x}) = \mathrm{pow}_q(\mathbf{x}) \cdot \sqrt{\|\mathbf{x}\|_H}$, where $\|\mathbf{x}\|_H$ is the Hamming weight of $\mathbf{x}$. For any integer lattice Λ,

$$\tau_q(\Lambda) = \min\big\{\tau_q(\mathbf{x}) \colon \mathbf{x} \in \Lambda \setminus \{\mathbf{0}\}\big\}$$

is the minimum of τ_q over all nonzero lattice vectors.

Notice that τ_q is not a norm because it satisfies neither the linearity property $\|c \cdot \mathbf{x}\| = c \cdot \|\mathbf{x}\|$, nor the triangle inequality $\|\mathbf{x}+\mathbf{y}\| \le \|\mathbf{x}\| + \|\mathbf{y}\|$ required of a norm. Still, the quantity $\tau_q(\Lambda)$ is useful to study SVP because it gives a lower bound on the norm of integer vectors, and it behaves well with respect to the tensor product of lattices, as shown below.

Lemma 3.2. *For any integer vector $\mathbf{x} \in \mathbb{Z}^n$, $\tau_q(\mathbf{x}) \le \|\mathbf{x}\|$.*

Proof. The vector $\mathbf{x}$ has $\|\mathbf{x}\|_H$ nonzero entries, and each of them is at least $\mathrm{pow}_q(\mathbf{x})$ in absolute value. Therefore $\|\mathbf{x}\| \ge \mathrm{pow}_q(\mathbf{x}) \cdot \sqrt{\|\mathbf{x}\|_H} = \tau_q(\mathbf{x})$. □

Lemma 3.3. *For any integer lattice Λ and (arbitrary) lattice Λ',*

$$\tau_q(\Lambda) \cdot \lambda(\Lambda') \;\le\; \lambda(\Lambda \otimes \Lambda') \;\le\; \lambda(\Lambda) \cdot \lambda(\Lambda').$$

Proof. Let $\Lambda = \mathcal{L}(\mathbf{B})$ and $\Lambda' = \mathcal{L}(\mathbf{B}')$. For the upper bound, simply observe that for any two lattice vectors $\mathbf{Bx}$ and $\mathbf{B}'\mathbf{y}$, the product lattice $\Lambda \otimes \Lambda'$ contains a vector $\mathbf{Bx}(\mathbf{B}'\mathbf{y})^T$ of length $\|\mathbf{Bx}\| \cdot \|\mathbf{B}'\mathbf{y}\|$. Choosing $\mathbf{Bx}$ and $\mathbf{B}'\mathbf{y}$ as shortest nonzero vectors in Λ and Λ' yields a vector in $\Lambda \otimes \Lambda'$ of length $\lambda(\Lambda) \cdot \lambda(\Lambda')$. In order to prove the lower bound we consider an arbitrary nonzero $\mathbf{v} = \mathbf{BX}(\mathbf{B}')^T$ in the tensor product lattice $\Lambda \otimes \Lambda'$, and show that $\|\mathbf{v}\| \ge \tau_q(\Lambda) \cdot \lambda(\Lambda')$. Let h be the number of nonzero rows in $\mathbf{BX}$. Clearly, all columns $\mathbf{c}$ of $\mathbf{BX}$ have Hamming weight at most $\|\mathbf{c}\|_H \le h$, and therefore $\tau_q(\mathbf{c}) \le \mathrm{pow}_q(\mathbf{c}) \cdot \sqrt{h}$. It follows that all nonzero columns $\mathbf{c}$ satisfy $\mathrm{pow}_q(\mathbf{c}) \ge \tau_q(\mathbf{c})/\sqrt{h} \ge \tau_q(\Lambda)/\sqrt{h}$. In particular, the largest power q^i that divides the entire matrix $\mathbf{BX}$ satisfies $q^i \ge \tau_q(\Lambda)/\sqrt{h}$. Notice that $\mathbf{v} = (\mathbf{BX}) \cdot (\mathbf{B}')^T$ contains exactly h nonzero rows (because $\mathbf{BX}$ has h nonzero rows, and the columns of $\mathbf{B}'$ are linearly independent), and each of them is a nonzero vector in $q^i\Lambda'$. Therefore, $\|\mathbf{v}\| \ge \sqrt{h}\, q^i \lambda(\Lambda') \ge \tau_q(\Lambda)\lambda(\Lambda')$. □

We use the quantity $\tau_q(\Lambda)$ to define a variant of SVP that behaves well with respect to the tensor product of lattices. Our variant of SVP is defined using the Euclidean norm for the YES instances, and our new measure τ_q for the NO instances.

Definition 3.4. The promise problem $\mathrm{TensorSVP}_\gamma$ (where $\gamma \ge 1$ may be a function of the lattice dimension) is defined as follows. Instances are pairs $(\mathbf{B}, d)$ where $\mathbf{B} \in \mathbb{Z}^{n \times k}$ is a lattice basis and d is a positive number such that

- $(\mathbf{B}, d)$ is a YES instance if $\lambda(\mathcal{L}(\mathbf{B})) \leq d$;
- $(\mathbf{B}, d)$ is a NO instance if $\tau_q(\mathcal{L}(\mathbf{B})) > \gamma d$.

Notice that TensorSVP$_\gamma$ is a special case of the standard GapSVP$_\gamma$ problem because the defining condition for YES instances is the same, and in the NO instances $\tau_q(\mathcal{L}(\mathbf{B}))$ is a lower bound on $\lambda(\mathcal{L}(\mathbf{B}))$. So, in order to establish NP-hardness results for GapSVP$_\gamma$ it is enough to prove the NP-hardness of TensorSVP$_\gamma$. Moreover, TensorSVP$_\gamma$ behaves well with respect to the tensor product of lattices, as described in the next theorem.

Theorem 3.5. *For any positive integer c, the map* $(\mathbf{B}, d) \mapsto (\mathbf{B}^{\otimes c}, d^c)$ *is a reduction from* TensorSVP$_\gamma$ *to* GapSVP$_{\gamma^c}$*, where* $\mathbf{B}^{\otimes c}$ *denotes the iterated tensor product of c copies of* $\mathbf{B}$*. The running time of the reduction is polynomial in* S^c *where* $S = |(\mathbf{B}, d)|$ *is the size of the input.*

Proof. Let $(\mathbf{B}, d)$ be an instance of TensorSVP$_\gamma$. If $(\mathbf{B}, d)$ is a YES instance, then $\lambda(\mathcal{L}(\mathbf{B})) \leq d$ and, by Lemma 3.3, $\lambda(\mathcal{L}(\mathbf{B}^{\otimes c})) \leq d^c$. So, $(\mathbf{B}^{\otimes c}, d^c)$ is a YES instance of GapSVP$_{\gamma^c}$. Conversely, if $(\mathbf{B}, d)$ is a NO instance, then $\lambda(\mathcal{L}(\mathbf{B})) \geq \tau_q(\mathcal{L}(\mathbf{B})) > \gamma d$ and, by Lemma 3.3, $\lambda(\mathcal{L}(\mathbf{B}^{\otimes c})) > (\gamma d)^c$. So, $(\mathbf{B}^{\otimes c}, d^c)$ is a NO instance of GapSVP$_{\gamma^c}$. □

Notice that for any constant c, the transformation in Theorem 3.5 runs in polynomial time. So, if TensorSVP$_\gamma$ is NP-hard for *some* constant $\gamma > 1$, then GapSVP$_\gamma$ is NP-hard for *any* constant $\gamma' \leq \gamma^c$. Similarly, using reductions that run in superpolynomial time, one obtains inapproximability results for even larger factors. (See Corollary 5.2.)

Notice that all definitions and results presented in this section hold for any integer q. For simplicity, in the rest of the paper, we fix $q = 2$, as this is enough to prove the hardness of GapSVP.

4 A dense lattice construction

Similarly to [20], our NP-hardness proof is based on the construction of a very dense lattice $\mathcal{L}(\mathbf{L})$ with large minimum distance. However, since we want to prove the NP-hardness of TensorSVP (rather than just GapSVP as in [20]), we need a lattice $\mathcal{L}(\mathbf{L})$ such that not only $\lambda(\mathcal{L}(\mathbf{L}))$, but also $\tau_2(\mathcal{L}(\mathbf{L}))$ is large. Our lattice is obtained by combining together a carefully chosen sequence of linear codes described in the following lemma.

Lemma 4.1. *There is an efficient algorithm that, on input m and h, both being powers of two and satisfying* $h \leq \sqrt{m}$*, outputs a sequence of generator matrices* $\mathbf{C}_i \in \mathbb{F}_2^{m \times k_i}$ $(i = 0, \ldots, \log_2 h)$ *for binary linear codes* $\mathcal{C}_i[m, k_i, d_i]_2 = \mathcal{C}(\mathbf{C}_i)$ *of common block length m, minimum distance* $d_i \geq 4^i$*, and co-dimension* $m - k_i \leq (\log_2 m) \cdot (4^i/2 - 1) + 1$ *(for* $i \geq 1$*), such that* $\mathbb{F}_2^m = \mathcal{C}_0 \supseteq \mathcal{C}_1 \supseteq \cdots \supseteq \mathcal{C}_{\log_2 h}$*. The running time of the algorithm is polynomial in m.*

Proof. For $i = 0, \ldots, \ell = \log_2 h$, let $\mathbf{C}_i$ be a generator matrix for the extended BCH code $\mathrm{EBCH}^m_{4^i}$ of block length m and designed distance $4^i \leq d_i$. By Theorem 2.3 the generator matrices can be constructed in time polynomial in m, they have co-dimension $m - k_i \leq (\log_2 m) \cdot (4^i/2 - 1) + 1$ for $i \geq 1$, and the codes they generate form a chain $\mathbb{F}_2^m = \mathcal{C}_0 \supseteq \mathcal{C}_1 \supseteq \cdots \supseteq \mathcal{C}_\ell$. □

These codes are combined into a lattice $\mathcal{L}(\mathbf{L})$ using a method that is essentially "construction D" from [9, Chapter 8], the only difference being that here we use a scaled copy of the lattice so that $\mathbf{L}$ is an integer matrix, and we express the bound in terms of $\tau_2(\mathcal{L}(\mathbf{L}))$ rather than $\lambda(\mathcal{L}(\mathbf{B}))$.

Theorem 4.2. *There is an efficient algorithm that, on input m and h, both being powers of two and satisfying $h \le \sqrt{m}$, outputs an m-dimensional full-rank integer lattice basis $\mathbf{L} \in \mathbb{Z}^{m\times m}$ such that $\tau_2(\mathcal{L}(\mathbf{L})) \ge h$ and $\det(\mathcal{L}(\mathbf{L})) < m^{2h^2/3}$. The running time of the algorithm is polynomial in m.*

Proof. We need to give an efficient construction that, on input $m = 2^\kappa$ and $h = 2^\ell$ with $\ell \le \kappa/2$, produces an m-dimensional full-rank integer lattice basis $\mathbf{L}$ such that $\tau_2(\mathcal{L}(\mathbf{L})) \ge h$ and $\det(\mathcal{L}(\mathbf{L})) < m^{2h^2/3}$.

Run the algorithm of Lemma 4.1 on input m and h to obtain the generator matrices $\mathbf{C}_0, \mathbf{C}_1, \cdots, \mathbf{C}_\ell$ for the sequence of binary linear codes $\mathcal{C}_i[m, k_i, d_i]_2$. We recall that these are codes of common block length m, minimum distance $d_i \ge 4^i$, and co-dimension $m - k_i \le \kappa(4^i/2 - 1) + 1$ for $i \ge 1$, and $\mathcal{C}_0[m, m, 1]_2 = \mathbb{F}_2^m$. We combine these codes into a lattice using "construction D" from [9, Chapter 8, Theorem 13]. More specifically, we define the m-dimensional integer lattice $\mathcal{L}(\mathbf{L})$ generated by the columns of $2^{\ell-i}\mathbf{C}_i$ for all $i = 0, \ldots, \ell$, where where each $\mathbf{C}_i \in {\mathbb{F}_2}^{m\times k_i}$ is interpreted as an integer matrix in $\{0,1\}^{m\times k_i}$ by identifying the elements of $\mathbb{F}_2 = \{0,1\}$ with the integers $\{0,1\} \subset \mathbb{Z}$. Of course, the vectors in $2^\ell \mathbf{C}_0, 2^{\ell-1}\mathbf{C}_1, \ldots, \mathbf{C}_\ell$ are not linearly independent, but a basis for the lattice they generate can be easily obtained as follows. Using the inclusions $\mathcal{C}_0 \supseteq \cdots \supseteq \mathcal{C}_\ell$, we may assume that each generating matrix $\mathbf{C}_i$ equals the last k_i columns of $\mathbf{C}_0$. In other words, $\mathbf{C}_0 = [\mathbf{K}_0, \ldots, \mathbf{K}_\ell]$, and each generating matrix $\mathbf{C}_i = [\mathbf{K}_i, \mathbf{C}_{i+1}]$ is obtained extending the generating matrix of the next code in the sequence $\mathbf{C}_{i+1}$ with $k_i' = k_i - k_{i+1}$ more columns $\mathbf{K}_i$. (For convenience, we also define $k_{\ell+1} = 0$ and $k_\ell' = k_\ell - k_{\ell+1} = k_\ell$.) By properly choosing the order of the rows and performing elementary column operations (see below for details), we may further assume that each $\mathbf{K}_i$ has the form

$$\mathbf{K}_i = \begin{bmatrix} \mathbf{K}_i' \\ \mathbf{T}_i \\ \mathbf{O} \end{bmatrix}$$

where $\mathbf{K}_i' \in {\mathbb{F}_2}^{(m-k_i)\times k_i'}$, $\mathbf{T}_i$ is a $k_i' \times k_i'$ nonsingular upper-triangular[7] matrix, and $\mathbf{O}$ is the $k_{i+1} \times k_i'$ all-zero matrix. In more detail, this can be achieved using a simple variant of the Gaussian elimination process as follows:

1. Let $c_{i,j}$ be the entries of the matrix $\mathbf{C}_0$, and select an index i such that $c_{i,m} = 1$.
2. Swap rows i and m so that $c_{m,m} = 1$.
3. Subtract the column m from all columns $j < m$ with $c_{m,j} = 1$, so that $c_{m,j} = 0$.
4. Proceed inductively on the $(m-1)\times(m-1)$ upper left submatrix of $\mathbf{C}_0$.

This results in a nonsingular upper-triangular matrix $\mathbf{C}_0$ that is equivalent to the original $\mathbf{C}_0$.

Define the $m \times m$ integer matrix

$$\mathbf{L} = [2^\ell \mathbf{K}_0, 2^{\ell-1}\mathbf{K}_1, \ldots, \mathbf{K}_\ell]$$

[7] While not needed here, one could also turn each $\mathbf{T}_i$ into the identity matrix $\mathbf{T}_i = \mathbf{I}_{k_i'}$ by Gaussian elimination.

where, as before, each $\mathbf{K}_i \in \mathbb{F}_2{}^{m\times k'_i}$ is interpreted as an integer matrix in $\{0,1\}^{m\times k'_i}$.

The columns of $\mathbf{L}$ are a subset of $2^\ell\mathbf{C}_0, 2^{\ell-1}\mathbf{C}_1, \ldots, \mathbf{C}_\ell$. Moreover, all vectors in $2^\ell\mathbf{C}_0, 2^{\ell-1}\mathbf{C}_1, \ldots, \mathbf{C}_\ell$ can be obtained by multiplying the columns of $\mathbf{L}$ by appropriate powers of 2. Therefore $\mathcal{L}(\mathbf{L})$ is precisely the lattice generated by $2^\ell\mathbf{C}_0, 2^{\ell-1}\mathbf{C}_1, \ldots, \mathbf{C}_\ell$.

Consider an arbitrary nonzero lattice vector $\mathbf{v} = \sum_i (2^{\ell-i}\cdot\mathbf{K}_i)\mathbf{x}_i = \sum_i \mathbf{K}_i\mathbf{y}_i$, where $\mathbf{y}_i = 2^{\ell-i}\mathbf{x}_i$. We want to prove that $\tau_2(\mathbf{v}) \geq h$. Let $2^P = \mathrm{pow}_2(\mathbf{y}_0, \ldots, \mathbf{y}_k)$ be the largest power of 2 that divides $\mathbf{y}_i$ for $0 \leq i \leq k$. Clearly, 2^P also divides $\mathbf{v}$. If $P \geq \ell$, then we immediately get $\tau_2(\mathbf{v}) \geq \mathrm{pow}_2(\mathbf{v}) \geq 2^\ell = h$. So, assume $P < \ell$ and let

$$p = \min\{i\colon \mathbf{y}_i \neq \mathbf{0}, \mathrm{pow}_2(\mathbf{y}_i) = 2^P\}$$

be the smallest index such that $\mathbf{y}_p$ is divisible precisely by 2^P. Notice that $P \geq \ell - p$ because

$$2^P = \mathrm{pow}_2(\mathbf{y}_p) = \mathrm{pow}_2(2^{\ell-p}\mathbf{x}_p) \geq 2^{\ell-p}.$$

By definition of p and P, all $\mathbf{y}_i/2^P$ are integer vectors, and $\mathbf{y}_p/2^P \neq \mathbf{0} = \mathbf{y}_i/2^P \pmod 2$ for all $i < p$. So,

$$\|\mathbf{v}\|_H = \|\mathbf{v}/2^P\|_H \geq \|(\mathbf{v}/2^P) \bmod 2\|_H = \left\|\sum_{i\geq p}\mathbf{K}_i(\mathbf{y}_i/2^P) \bmod 2\right\|_H \geq d_p \geq 4^p$$

where we have used the fact that $\sum_{i\geq p}\mathbf{K}_i(\mathbf{y}_i/2^P) \bmod 2$ is a nonzero codeword in $\mathcal{C}_p$. It follows that

$$\tau_2(\mathbf{v}) = \mathrm{pow}_2(\mathbf{v})\sqrt{\|\mathbf{v}\|_H} \geq 2^P\cdot\sqrt{4^p} \geq 2^{\ell-p}\cdot 2^p = 2^\ell.$$

This proves that $\tau_2(\mathcal{L}(\mathbf{L})) \geq 2^\ell = h$.

In order to bound the determinant of the lattice, we notice that, by our choice of $\mathbf{K}_i$, the matrix $\mathbf{C}_0$ is upper triangular. It follows that $\mathbf{L}$ is also a triangular matrix with k'_i diagonal entries equal to $2^{\ell-i}$ for $i = 0, \ldots, \ell$. So, the determinant satisfies

$$\log_2\det(\mathcal{L}(\mathbf{L})) = \sum_{i=0}^{\ell}(\ell-i)k'_i = \sum_{i=1}^{\ell}(m-k_i). \tag{4.1}$$

Finally, using the bound on the co-dimension $m - k_i \leq \kappa\cdot(4^i/2 - 1) + 1$ from Lemma 4.1 we get

$$\sum_{i=1}^{\ell}(m-k_i) \leq \sum_{i=1}^{\ell}\left(\kappa\cdot\frac{4^i}{2} - (\kappa-1)\right) = \kappa\frac{2}{3}(4^\ell - 1) - (\kappa-1)\ell < \frac{2\kappa h^2}{3}.$$

(In fact, using the assumption $\ell \leq \kappa/2 < (2/3)\kappa$, one can slightly strengthen the upper bound to $\kappa\left((2/3)h^2 - \ell\right)$.) Substituting this bound into the expression for the determinant gives

$$\det(\mathcal{L}(\mathbf{L})) < 2^{2\kappa h^2/3} = m^{\frac{2}{3}h^2}.$$

□

We conclude this section with some remarks about the lattice $\mathcal{L}(\mathbf{L})$ constructed in Theorem 4.2. To start with, we observe that Theorem 4.2 can be applied using a single (BCH) code $\mathcal{C}_1 \subseteq \mathbb{F}_2^n$. This is reminiscent of the use of BCH codes found in [14, 12], and corresponds to "Construction A" of [9]. However, using only a single code (i. e., $\ell = \log_2 h = 1$) results in lattices with very small $\tau_2(\mathcal{L}(\mathbf{L})) =$

$h = 2$, which are not directly useful to prove the NP-hardness of SVP. Of course, one can always increase $\tau_2(\mathcal{L}(\mathbf{L}))$ by multiplying the lattice by a power of 2. For example, the orthogonal lattice $h\mathbb{Z}^m$ satisfies $\tau_2(h\mathbb{Z}^m) = h$ and $\det(h\mathbb{Z}^m) = h^m$, but it is not very interesting. Our lattice construction achieves the same $\tau_2(\mathcal{L}(\mathbf{L})) \geq h$ but, by (4.1), it has much smaller determinant $\det(\mathcal{L}(\mathbf{L})) = h^m/2^{\sum_{i=1}^{\ell} k_i}$, where $k_1, \ldots, k_\ell$ are the dimensions of the codes $\mathcal{C}_i$ from Lemma 4.1.

A systematic way to compare different lattices obtained from this or other constructions is to evaluate their Hermite factor $\gamma(\Lambda) = (\lambda(\Lambda)/\det(\Lambda)^{1/m})^2$. This factor is closely related to the packing density of the lattice

$$\frac{\mathrm{vol}(\mathcal{B}(\lambda(\Lambda)/2))}{\det(\Lambda)} = \mathrm{vol}(\mathcal{B}(1)) \cdot \left(\sqrt{\gamma(\Lambda)}/2\right)^m,$$

i. e., the largest fraction of space occupied by disjoint equal spheres centered around all lattice points. The trivial fact that the packing density cannot be higher than 1 yields Minkowski's upper bound on the Hermite factor $\gamma(\Lambda) \leq O(m)$. Notice that the Hermite factor $\gamma(\Lambda)$ is invariant under scaling of the lattice. So, one can consider arbitrary values of h in Theorem 4.2, and then scale the lattice to increase $\tau_2(\mathcal{L}(\mathbf{L})) \leq \lambda(\mathcal{L}(\mathbf{L}))$ as desired, without affecting the Hermite factor of the lattice. Using

$$\det(\mathcal{L}(\mathbf{L})) = \frac{h^m}{2^{\sum_{i=1}^{\ell} k_i}} \quad \text{and} \quad \lambda(\mathcal{L}(\mathbf{L})) \geq \tau_2(\mathcal{L}(\mathbf{L})) \geq h$$

gives Hermite factor $\gamma(\mathcal{L}(\mathbf{L})) \geq 4^{\sum_{i=1}^{\ell} k_i/m}$. The integer lattice $\mathbb{Z}^m$ (corresponding to $\ell = 0$) achieves Hermite factor $\gamma \geq 1$. Using a single BCH code $\mathcal{C}_1 \subseteq \mathbb{F}_2^m$ (corresponding to $\ell = 1$) as in [14, 12] gives only marginally higher Hermite factor $\gamma \geq 4^{k_1/m} \geq 4^{1+o(1)}$. By contrast, choosing $h = \Theta(\sqrt{m/\log m})$ in Theorem 4.2 so to maximize the Hermite factor yields an almost optimal

$$\gamma \geq \left(h/m^{O(h^2/m)}\right)^2 = \Omega(h^2) = \Omega(m/\log m),$$

within a logarithmic factor from Minkowski's upper bound $\gamma \leq O(m)$.

Our lattice should also be compared to the "Schnorr-Adleman" prime number lattice, already used in [1, 20, 21] to prove NP-hardness results for SVP. In [21, Chapter 5] it is shown that this lattice (parametrized by a real $\alpha > 0$ and a sequence of relatively prime integers $a_1, \ldots, a_m$) has minimum distance $\lambda \geq 2\ln\alpha$ (see [21, Lemma 5.3]) and determinant

$$\det(\Lambda) = \sqrt{\left(1 + \alpha^2 \sum_k \ln a_k\right) \prod_k \ln a_k}$$

(see [21, Proposition 5.9]). The Hermite factor is maximized by setting $a_1, \ldots, a_m$ to the first m prime numbers and $\alpha \approx e^{m/2}$, which yields $\gamma(\Lambda) = \Omega(m/\log m)$, just like for the lattice given in Theorem 4.2. For other examples of explicit lattice constructions achieving similar or even higher density see [9, Chapter 1, Section 1.5]. The novelty in Theorem 4.2 is that it provides not only a lattice with large minimum distance (and Hermite factor), but also large $\tau_2(\Lambda)$.

5 The main reduction

In this section we prove that TensorSVP is NP-hard to approximate within some constant factor under *nonuniform* polynomial-time reductions with one-sided error. In Section 6 we show how the nonuniform advice required by our proof can be computed in probabilistic polynomial time. We present our main result as a nonuniform reduction first in order to make the presentation as simple as possible. We remark that the nonuniform reduction presented in this section is just as good a starting point for derandomization as the probabilistic reduction presented in the next section. A randomized uniform reduction is presented in Section 6 mostly to reassure the reader that here we are not using the nonuniformity of the advice in any essential way.

For any $1 < \lambda < \sqrt{3/2}$, the reduction uses a gadget $(\mathbf{L}, \mathbf{s}, \mathbf{T}, r)$ consisting of a lattice basis $\mathbf{L} \in \mathbb{Z}^{m\times \ell}$, a vector $\mathbf{s} \in \mathbb{Z}^m$, a linear transformation $\mathbf{T} \in \mathbb{Z}^{k\times m}$, and a bound r such that

$$\begin{aligned} 1 \leq \quad r \quad &\leq \sqrt{m}\,, && (5.1)\\ \tau_2(\mathcal{L}(\mathbf{L})) \quad &\geq \quad \lambda\cdot r\,, && (5.2)\\ \mathbf{T}((\mathcal{L}(\mathbf{L})-\mathbf{s})\cap\mathcal{B}(r)) \quad &\supseteq \quad \{0,1\}^k\,. && (5.3)\end{aligned}$$

Informally, the lattice has large τ_2, and still one of its cosets contains 2^k short vectors that are mapped by the linear transformation $\mathbf{T}$ to the set of all binary vectors in dimension k.

The gadget is obtained using Theorem 4.2 to efficiently build a very dense lattice $\mathcal{L}(\mathbf{L})$ satisfying (5.2). Since this lattice has small determinant, there must be a coset $(\mathcal{L}(\mathbf{L})-\mathbf{s})$ containing many short binary vectors of norm bounded by r. Provided the number of such short binary vectors is large enough, Lemma 2.9 implies that there is a matrix $\mathbf{T} \in \{0,1\}^{k\times m}$ such that the image of the short vectors under $\mathbf{T}$ includes all binary strings in $\{0,1\}^k$, i. e., (5.3) is satisfied. Both $\mathbf{s}$ and $\mathbf{T}$ are part of the nonuniform advice used by the reduction.

Theorem 5.1. *For any $1 \leq \gamma < \lambda < \sqrt{3/2}$ and*

$$\tilde{\gamma} = \gamma\sqrt{1+\frac{4}{(\lambda/\gamma)^2-1}}$$

there is a nonuniform polynomial-time reduction with one-sided error and no false positives from TensorCVP$_{\tilde{\gamma}}$ *to* TensorSVP$_\gamma$*.*

Proof. Let $(\mathbf{B}, \mathbf{y}, t)$ be a TensorCVP$_{\tilde{\gamma}}$ instance with $\mathbf{B} \in \mathbb{Z}^{n\times k}$ and $\mathbf{y} \in \mathbb{Z}^n$. We recall that membership in any lattice $\mathcal{L}(\mathbf{B})$ can be efficiently determined by linear algebra. If the target is a lattice vector $\mathbf{y} \in \mathcal{L}(\mathbf{B})$, then $(\mathbf{B}, \mathbf{y}, t)$ is certainly a YES instance and the reduction can output an arbitrary YES instance of TensorSVP$_\gamma$. So, assume without loss of generality that $\mathbf{y} \notin \mathcal{L}(\mathbf{B})$. Assume also that $1 \leq t < \sqrt{n}$. Again, this is without loss of generality because if $t \geq \sqrt{n}$, then $(\mathbf{B}, \mathbf{y}, t)$ is not a NO instance and it can be mapped to an arbitrary YES instance of TensorSVP$_\gamma$. Similarly, if $t < 1$, then $(\mathbf{B}, \mathbf{y}, t)$ is not a YES instance and it can be mapped to an arbitrary NO instance of TensorSVP$_\gamma$.

We will give a nonuniform polynomial-time construction of a gadget $(\mathbf{L}, \mathbf{s}, \mathbf{T}, r)$ satisfying properties (5.1), (5.2) and (5.3), where $(\mathbf{s}, \mathbf{T})$ is the nonuniform advice. Notice that (5.1) and (5.2) do not depend on the advice $(\mathbf{s}, \mathbf{T})$, and so they are always satisfied. Only (5.3) relies on the advice $(\mathbf{s}, \mathbf{T})$ being properly chosen. Given, $(\mathbf{L},\mathbf{s},\mathbf{T},r)$, we proceed as follows. We begin by scaling the input $(\mathbf{B},\mathbf{y},t)$ and the gadget $(\mathbf{L},\mathbf{s},\mathbf{T},r)$ so that $\varepsilon/2 \le t/r < \varepsilon$, where $\varepsilon = \sqrt{(\lambda/\gamma)^2-1} \in (0,1/\sqrt{2})$. Specifically, if $t/r < \varepsilon/2$, then we replace $(\mathbf{B},\mathbf{y},t)$ with $(\mathbf{1}_{c^2}\otimes\mathbf{B},\mathbf{1}_{c^2}\otimes\mathbf{y},c\cdot t)$ where $c = \lceil \varepsilon r/(2t)\rceil$. Similarly, if $t/r \ge \varepsilon$, then we replace $(\mathbf{L},\mathbf{s},\mathbf{T},r)$ with $(\mathbf{1}_{c^2}\otimes\mathbf{L},\mathbf{1}_{c^2}\otimes\mathbf{s},\mathbf{e}_1\otimes\mathbf{T},c\cdot r)$, where $\mathbf{e}_1 = [1,0,\ldots,0]$ is the first c^2-dimensional standard unit row vector and $c = \lfloor 2t/(\varepsilon r)\rfloor$. Notice that in either case $c \le O(\max(r/t,t/r)) \le O(\max(\sqrt{m},\sqrt{n}))$ is polynomially bounded, and therefore the scaling transformation runs in polynomial time. It is also easy to verify that the transformation results in an equivalent TensorCVP$_{\tilde{\gamma}}$ instance $(\mathbf{B},\mathbf{y},t)$ and gadget $(\mathbf{L},\mathbf{s},\mathbf{T},r)$ such that $\varepsilon/2 \le t/r < \varepsilon$.

The output of the reduction is $(\mathbf{V},d)$, where $d = \sqrt{t^2+r^2}$ and

$$\mathbf{V} = \left[\begin{array}{c|c} \mathbf{BTL} & \mathbf{BTs}+\mathbf{y} \\ \mathbf{L} & \mathbf{s} \end{array}\right].$$

Notice that $\mathbf{V} \in \mathbb{Z}^{(n+m)\times(\ell+1)}$ is a basis, i. e., its columns are linearly independent. To see this, notice that $\mathbf{L}$ itself is a basis (making the first ℓ columns of $\mathbf{V}$ linearly independent). So, the only way that $\mathbf{V}$ could possibly fail to be a basis is for the last column to be a linear combination of the first ℓ, i. e., $\mathbf{s} = \mathbf{Lz}$ and $\mathbf{BTs}+\mathbf{y} = \mathbf{BTLz} = \mathbf{BTs}$ for some $\mathbf{z} \in \mathbb{R}^\ell$. But this implies $\mathbf{y} = \mathbf{0} \in \mathcal{L}(\mathbf{B})$, which we assumed not to be the case.

We show that the reduction is correct. First, assume $(\mathbf{B},\mathbf{y},t)$ is a NO instance of TensorCVP$_{\tilde{\gamma}}$, i. e., $\|\mathbf{y}-\mathbf{Bx}\|_H > (\tilde{\gamma}t)^2$ for all $\mathbf{x} \in \mathbb{R}^k$, and let the advice $(\mathbf{s},\mathbf{T})$ be arbitrary. Consider any nonzero lattice vector

$$\mathbf{v} = \mathbf{V}\left[\begin{array}{c} \mathbf{z} \\ w \end{array}\right] = \left[\begin{array}{c} \mathbf{BT}(\mathbf{Lz}+w\mathbf{s})+w\mathbf{y} \\ \mathbf{Lz}+w\mathbf{s} \end{array}\right].$$

On the one hand, if $w \ne 0$, then the vector $\mathbf{v}$ satisfies

$$\begin{array}{rcl} \tau_2(\mathbf{v})^2 & \ge & \|\mathbf{v}\|_H \\ & \ge & \|\mathbf{BT}(\mathbf{Lz}+w\mathbf{s})+w\mathbf{y}\|_H \\ & = & \|\mathbf{y}-\mathbf{B}(-\mathbf{T}(\mathbf{Lz}/w+\mathbf{s}))\|_H \\ & > & (\tilde{\gamma}t)^2 = \gamma^2(t^2+(2t/\varepsilon)^2) \ge \gamma^2 d^2\,, \end{array}$$

where the last inequality is equivalent to the condition $\varepsilon/2 \le t/r$. On the other hand, if $w = 0$, then $\mathbf{z} \ne \mathbf{0}$ and

$$\mathbf{v} = \left[\begin{array}{c} \mathbf{BT}(\mathbf{Lz}) \\ \mathbf{Lz} \end{array}\right]$$

is divisible by $\mathrm{pow}_2(\mathbf{Lz})$. Moreover, $\|\mathbf{v}\|_H \ge \|\mathbf{Lz}\|_H$, and therefore

$$\tau_2(\mathbf{v})^2 \ge \mathrm{pow}_2(\mathbf{Lz})^2\cdot\|\mathbf{Lz}\|_H = \tau_2(\mathbf{Lz})^2 \ge \tau_2(\mathcal{L}(\mathbf{L}))^2 \ge \lambda^2 r^2 > \gamma^2\cdot d^2\,,$$

where we have used (5.2), and the last inequality is equivalent to the condition $t/r < \varepsilon$. This proves that $\tau_2(\mathcal{L}(\mathbf{B})) > \gamma d$, i. e., $(\mathbf{B},d)$ is a NO instance of TensorSVP$_\gamma$.

Now let $(\mathbf{s},\mathbf{T})$ be an advice such that (5.3) holds true, and assume $(\mathbf{B},\mathbf{y},t)$ is a YES instance of TensorCVP$_{\tilde{\gamma}}$, i. e., there is an $\mathbf{x} \in \{0,1\}^k$ such that $\|\mathbf{y}-\mathbf{Bx}\| \le t$. By (5.3), there is an integer vector

$\mathbf{z} \in \mathbb{Z}^l$ such that $\mathbf{T}(\mathbf{L}\mathbf{z}-\mathbf{s}) = \mathbf{x}$ and $\|\mathbf{L}\mathbf{z}-\mathbf{s}\| \le r$. So, the lattice vector

$$\mathbf{v} = \mathbf{V}\begin{bmatrix} \mathbf{z} \\ -1 \end{bmatrix} = \begin{bmatrix} \mathbf{B}\mathbf{T}(\mathbf{L}\mathbf{z}-\mathbf{s}) - \mathbf{y} \\ \mathbf{L}\mathbf{z}-\mathbf{s} \end{bmatrix} = \begin{bmatrix} \mathbf{B}\mathbf{x} - \mathbf{y} \\ \mathbf{L}\mathbf{z}-\mathbf{s} \end{bmatrix}$$

has squared norm $\|\mathbf{v}\|^2 = \|\mathbf{B}\mathbf{x}-\mathbf{y}\|^2 + \|\mathbf{L}\mathbf{z}-\mathbf{s}\|^2 \le t^2 + r^2 = d^2$. This proves that $(\mathbf{B}, d)$ is a YES instance of TensorSVP$_\gamma$.

In order to complete the proof we need to construct a gadget $(\mathbf{L}, \mathbf{s}, \mathbf{T}, r)$ as required by the reduction. We recall that the construction takes as input the lattice dimension k and a nonuniform advice $(\mathbf{s}, \mathbf{T})$, and should run in time polynomial in k. Let $h = \omega(\sqrt{k})$ be a sufficiently large (still, polynomially bounded $h = k^{O(1)}$) power of 2, and set $m = h^c$ for some integer constant

$$c > \left(\frac{1}{2} - \frac{1}{3}\lambda^2\right)^{-1} > 2\,.$$

Define $r = \sqrt{\lfloor (h/\lambda)^2 \rfloor} \le h/\lambda$, and run the algorithm of Theorem 4.2 on input m and h to obtain a basis $\mathbf{L} \in \mathbb{Z}^{m\times m}$ such that $\tau_2(\mathcal{L}(\mathbf{L})) \ge h \ge \lambda r$ and $\det(\mathcal{L}(\mathbf{L})) < m^{2h^2/3}$. Clearly, (5.2) is always satisfied by construction. Also (5.1) holds true because $r \le h/\lambda < h < \sqrt{m}$ and $r \ge \sqrt{\lfloor h^2/(3/2) \rfloor} > 1$.

We need to show that there exists an advice $(\mathbf{s}, \mathbf{T})$ such that (5.3) also holds true. Let $\mathcal{A}$ be the set of all vectors in $\{0,1\}^m$ of norm r. Notice that r^2 is an integer, and $\mathcal{A}$ equals the set of all binary vectors of Hamming weight r^2. In particular, the size of $\mathcal{A}$ is

$$|\mathcal{A}| = \binom{m}{r^2} \ge \left(\frac{m}{r^2}\right)^{r^2} > \left(\frac{m}{h^2}\right)^{r^2} = m^{\left(1-\frac{2}{c}\right)r^2}. \tag{5.4}$$

We claim that there exists an $\mathbf{s} \in \mathbb{Z}^m$ such that

$$|\mathcal{A} \cap (\mathcal{L}(\mathbf{L}) - \mathbf{s})| \ge m^k > \sum_{i=0}^{k-1} \binom{m}{i}.$$

It will follow from Lemma 2.9 that there is a matrix $\mathbf{T} \in \{0,1\}^{k\times m}$ such that $\mathbf{T}((\mathcal{L}(\mathbf{L}) - \mathbf{s}) \cap \mathcal{A}) \supseteq \{0,1\}^k$, i. e., (5.3) holds true.

It remains to show that $|\mathcal{A} \cap (\mathcal{L}(\mathbf{L}) - \mathbf{s})| \ge m^k$ for some $\mathbf{s} \in \mathbb{Z}^m$. Notice that $\mathcal{L}(\mathbf{L})$ has precisely $\det(\mathcal{L}(\mathbf{L}))$ cosets of the form $\mathcal{L}(\mathbf{L}) - \mathbf{s}$ with $\mathbf{s} \in \mathbb{Z}^m$. It follows by an averaging argument that there is some $\mathbf{s} \in \mathbb{Z}^m$ such that

$$|\mathcal{A} \cap (\mathcal{L}(\mathbf{L}) - \mathbf{s})| \ge \frac{|\mathcal{A}|}{\det(\mathcal{L}(\mathbf{L}))} > m^{\left(1-\frac{2}{c}\right)r^2 - \left(\frac{2}{3}\right)h^2} > m^{\left(\frac{1-2/c}{\lambda^2} - \frac{2}{3}\right)h^2 - \left(1-\frac{2}{c}\right)}.$$

We need the exponent in this last expression to be at least k. But the coefficient of h^2 in the exponent is a strictly positive constant because $c > (\frac{1}{2} - \frac{1}{3}\lambda^2)^{-1}$. Therefore the exponent is at least $\Omega(h^2) - O(1) = \omega(k) - O(1) = \omega(k) \ge k$ for all sufficiently large k. $\square$

It easily follows that GapSVP$_\gamma$ is NP-hard to approximate within any constant approximation factor under polynomial-time nonuniform reductions with one-sided error. Larger inapproximability factors can also be obtained under superpolynomial-time reductions.

Corollary 5.2. GapSVP_γ *is* NP*-hard for any constant factor* γ *under polynomial-time nonuniform reductions with one-sided error and no false positives. Moreover, for every* $\varepsilon > 0$ *there is a* $\delta > 0$ *such that* GapSVP_γ *is* NP*-hard for* $\gamma(n) = n^{\delta/\log\log n}$ *under nonuniform reductions with no false positives, running in subexponential time* $2^{O(n^\varepsilon)}$.

Proof. By Theorem 2.7, $\mathrm{TensorCVP}_{\tilde{\gamma}}$ is NP-hard (under deterministic polynomial-time reductions) for any constant factor $\tilde{\gamma}$. If follows from Theorem 5.1 that $\mathrm{TensorSVP}_{\gamma_0}$ is NP-hard for some constant $\gamma_0 > 1$ under nonuniform reductions. Finally, for any constant γ, applying Theorem 3.5 with $c = \lceil \log\gamma / \log\gamma_0 \rceil$, we get that GapSVP_γ is NP-hard under the same kind of reductions. (Notice that for any constant γ, c is a constant and the reduction in Theorem 3.5 runs in polynomial time.)

In general, the reduction runs in time polynomial in $N = n^c$ and produces GapSVP_γ instances in dimension N that are hard to approximate within a factor $\gamma = \gamma_0^c$. For any $\varepsilon > 0$, let $\delta = \varepsilon \cdot \log\gamma_0$ and set $c = n^\varepsilon / \log n$, so that $N = n^c = 2^{n^\varepsilon}$ and the reduction runs in subexponential time $N^{O(1)} = 2^{O(n^\varepsilon)}$. The resulting inapproximability factor is $\gamma(N) = \gamma_0^c = N^{\delta/\log\log N}$. □

6 A probabilistic reduction

The reduction presented in Section 5 uses the nonuniform advice only for the construction of a gadget $(\mathbf{L}, \mathbf{s}, \mathbf{T}, r)$ satisfying properties (5.1), (5.2) and (5.3). In the proof of Theorem 5.1, we gave a (uniform, deterministic) polynomial-time construction of $\mathbf{L}$ and r satisfying (5.1) and (5.2), and then we proved that there exist $\mathbf{s}$ and $\mathbf{T}$ such that (5.3) holds true as well. This leads to a nonuniform reduction using $(\mathbf{s}, \mathbf{T})$ as advice. In this section we show that nonuniformity is not essential, and an advice $(\mathbf{s}, \mathbf{T})$ with the desired property can be efficiently found in probabilistic polynomial time. The idea is simple, and follows the same path as previous work [1, 20, 11]. First we find a coset $\mathcal{L}(\mathbf{L}) - \mathbf{s}$ containing many short vectors. Since the lattice $\mathcal{L}(\mathbf{L})$ has small determinant, the average number of short vectors in a random coset $\mathcal{L}(\mathbf{L}) - \mathbf{s}$ is large, and choosing $\mathbf{s}$ at random will give with high probability a coset containing many short vectors. After finding a coset $\mathcal{L}(\mathbf{L}) - \mathbf{s}$ that contains many short vectors, we use the following combinatorial theorem from [20] which can be interpreted as a constructive (probabilistic) variant of Sauer's lemma. We remark that a weaker form of the following theorem had already been proved and used by Ajtai [1].

Theorem 6.1 (Theorem 5.9 of [20])**.** *Let* $\mathcal{Z} \subseteq \{0,1\}^m$ *be a set of vectors of Hamming weight u. For any k and* $\varepsilon > 0$, *if* $|\mathcal{Z}| \geq u!\, m^{4\sqrt{uk}/\varepsilon}$, *and* $\mathbf{T} \in \{0,1\}^{k \times m}$ *is chosen by setting each entry to* 1 *independently at random with probability* $p = 1/(4uk)$, *then the probability that all binary vectors* $\{0,1\}^k$ *are contained in* $\mathbf{T}(\mathcal{Z}) = \{\mathbf{Tz} : \mathbf{z} \in \mathcal{Z}\}$ *is at least* $1 - 6\varepsilon$.

We use Theorem 6.1 to prove the following probabilistic variant of Theorem 5.1.

Theorem 6.2. *For any* $\gamma < \lambda < \sqrt{3/2}$ *and*

$$\tilde{\gamma} = \gamma \sqrt{1 + \frac{4}{(\lambda/\gamma)^2 - 1}}$$

there is a probabilistic polynomial-time reduction with one-sided error and no false positives from $\mathrm{TensorCVP}_{\tilde{\gamma}}$ *to* $\mathrm{TensorSVP}_\gamma$.

Proof. All that is needed to turn the nonuniform reduction of Theorem 5.1 into a randomized one is a probabilistic polynomial-time construction of a gadget $(\mathbf{L}, \mathbf{s}, \mathbf{T}, r)$, where $\mathbf{L} \in \mathbb{Z}^{m\times m}$, $\mathbf{s} \in \mathbb{Z}^m$, $\mathbf{T} \in \mathbb{Z}^{k\times m}$ and $r \in \mathbb{R}$ satisfy (5.1), (5.2) and (5.3). Moreover, for the reduction to have one-sided error with no false positives, properties (5.1) and (5.2) should hold for any value of the randomness. The lattice basis $\mathbf{L}$ and bound r are defined as in the proof of Theorem 5.1, but for larger values of $h = \omega(k)$,

$$c > 2\left(\frac{1}{2} - \frac{1}{3}\lambda^2\right)^{-1}$$

and $m = h^c$. Specifically, $r = \sqrt{\lfloor (h/\lambda)^2 \rfloor} \leq h/\lambda$, and $\mathbf{L} \in \mathbb{Z}^{m\times m}$ is a matrix (obtained invoking Theorem 4.2 on input m and h) such that $\tau_2(\mathcal{L}(\mathbf{L})) \geq h \geq \lambda r$ and $\det(\mathcal{L}(\mathbf{L})) < m^{2h^2/3}$. As in the proof of Theorem 5.1, properties (5.1) and (5.2) are always satisfied by construction. Here we give a probabilistic polynomial-time construction of $\mathbf{s}$ and $\mathbf{T}$ satisfying (5.3) with probability arbitrarily close to 1.

The construction is simple. The vector $\mathbf{s}$ is chosen by first picking $\mathbf{a} \in \mathcal{A}$ uniformly at random among all binary vectors $\mathcal{A}$ of norm r, and then setting $\mathbf{s} = -\mathbf{a}$. The matrix $\mathbf{T}$ is chosen at random as specified in Theorem 6.1 with $u = r^2$. For any $\varepsilon > 0$, if the set $\mathcal{Z} = \mathcal{A} \cap (\mathcal{L}(\mathbf{L}) + \mathbf{a}) = \mathcal{A} \cap (\mathcal{L}(\mathbf{L}) - \mathbf{s})$ has size at least $|\mathcal{Z}| \geq u!\, m^{4rk/\varepsilon}$, then by Theorem 6.1 property (5.3) is satisfied, except with probability at most 6ε over the choice of $\mathbf{T}$. It remains to show that $|\mathcal{Z}| \geq u!\, m^{4rk/\varepsilon}$ with high probability, over the choice of $\mathbf{a} \in \mathcal{A}$. This bound is based on the simple observation (also used in previous work [1, 20, 11, 14]) that for any finite function $f : A \to B$, the size of a random preimage satisfies

$$\Pr_{a\in A}\left\{ |f^{-1}(f(a))| \leq \varepsilon \frac{|A|}{|B|} \right\} \leq \varepsilon\,,$$

when $a \in A$ is chosen uniformly at random. In our setting $A = \mathcal{A}$, $a = \mathbf{a} \in \mathcal{A} \subset \mathbb{Z}^m$, $B = \mathbb{Z}^m/\mathcal{L}(\mathbf{L})$ and $f(\mathbf{a}) = \mathbf{a} \pmod{\mathcal{L}(\mathbf{L})}$. Notice that $\mathcal{Z} = \mathcal{A} \cap (\mathcal{L}(\mathbf{L}) + \mathbf{a}) = f^{-1}(f(\mathbf{a}))$. Therefore, by the above observation,

$$\Pr_{\mathbf{a}}[|\mathcal{Z}| \leq u!m^{4rk/\varepsilon}] \leq \frac{|B|}{|A|} \cdot u!m^{4rk/\varepsilon} = \frac{\det(\mathcal{L}(\mathbf{L}))}{|\mathcal{A}|} \cdot u!m^{4rk/\varepsilon}.$$

Using (5.4), $\det(\mathcal{L}(\mathbf{L})) < m^{2h^2/3}$ and the bound $u! < u^u < h^{2u} = m^{2r^2/c}$, we get

$$\frac{\det(\mathcal{L}(\mathbf{L}))}{|\mathcal{A}|} \cdot u!m^{4rk/\varepsilon} < m^{\frac{2}{3}h^2 - \left(1-\frac{4}{c}\right)r^2 + \frac{4rk}{\varepsilon}} = m^{\left(\frac{2}{3} - \left(1-\frac{4}{c}\right)\frac{1}{\lambda^2}\right)h^2 + o(h^2)}.$$

Since $c > 2/(\frac{1}{2} - \frac{1}{3}\lambda^2)$, the coefficient of h^2 in the last expression is a strictly negative constant, and the probability that $\Pr_{\mathbf{a}}[|\mathcal{Z}| \leq u!m^{4rk/\varepsilon}]$ is at most $m^{-\Omega(h^2)+o(h^2)} = m^{-\Omega(h^2)} < \varepsilon$. □

As in the previous section, the inapproximability factor can be amplified using the tensor product. The proof is virtually identical to that of Corollary 5.2.

Corollary 6.3. GapSVP_γ *is NP-hard for any constant factor* γ *under probabilistic polynomial-time reductions with one-sided error and no false positives. Moreover, for every* $\varepsilon > 0$ *there is a* $\delta > 0$ *such that* GapSVP_γ *is NP-hard for* $\gamma(n) = n^{\delta/\log\log n}$ *under randomized reductions with one-sided error and no false positives running in subexponential time* $2^{O(n^\varepsilon)}$.

7 Conclusion

We proved the hardness of approximating the Shortest Vector Problem for approximation factors matching the best currently known results [14, 12], but under probabilistic reductions with one-sided error. In particular, our reductions make more restricted use of randomness than [14, 12] and may be easier to derandomize. Randomness in our reductions is used only to produce a lattice coset $\mathcal{L}(\mathbf{L}) - \mathbf{s}$ with large minimum (τ_2) distance and still containing a large number of short vectors, which map via an integer linear transformation $\mathbf{T}$ onto the set of all binary vectors $\{0,1\}^k$. We gave a deterministic polynomial-time construction of the lattice $\mathcal{L}(\mathbf{L})$, and randomness is used only for the selection of $\mathbf{s}$ and $\mathbf{T}$. In fact, the matrix $\mathbf{T}$ is chosen at random mostly as a byproduct of the fact that the selection of $\mathbf{s}$ is probabilistic: intuitively, no matrix $\mathbf{T}$ is good for every $\mathbf{s}$, so if $\mathbf{s}$ is chosen at random, then $\mathbf{T}$ must be chosen at random as well. We believe that all that is needed in order to derandomize our proof is an explicit description of a vector $\mathbf{s}$ such that $\mathcal{L}(\mathbf{L}) - \mathbf{s}$ contains many short vectors. With such a vector $\mathbf{s}$ (and a proof that $\mathbf{s}$ is good), finding a matrix $\mathbf{T}$ that maps all short vectors in $\mathcal{L}(\mathbf{L}) - \mathbf{s}$ to $\{0,1\}^k$ is likely to be easy.

8 Acknowledgments

The author thanks Oded Regev and the anonymous referees of *Theory of Computing* for their many and useful comments and corrections to an earlier draft of this paper.

References

[1] MIKLÓS AJTAI: The shortest vector problem in L_2 is NP-hard for randomized reductions. In *Proc. 30th STOC*, pp. 10–19. ACM Press, 1998. ECCC. [doi:10.1145/276698.276705] 488, 489, 492, 498, 503, 507, 508

[2] MIKLÓS AJTAI, RAVI KUMAR, AND D. SIVAKUMAR: A sieve algorithm for the shortest lattice vector problem. In *Proc. 33rd STOC*, pp. 601–610. ACM Press, 2001. [doi:10.1145/380752.380857] 488

[3] SANJEEV ARORA, LÁSZLÓ BABAI, JACQUES STERN, AND ELIZABETH Z. SWEEDYK: The hardness of approximate optima in lattices, codes, and systems of linear equations. *J. Comput. System Sci.*, 54(2):317–331, 1997. Preliminary version in FOCS'93. [doi:10.1006/jcss.1997.1472] 489, 497

[4] PER AUSTRIN AND SUBHASH KHOT: A simple deterministic reduction for the gap minimum distance of code problem. In *Proc. 38th Internat. Colloq. on Automata, Languages and Programming (ICALP'11)*, pp. 474–485. Springer, 2011. [doi:10.1007/978-3-642-22006-7_40] 489, 490

[5] MIHIR BELLARE, SHAFI GOLDWASSER, CARSTEN LUND, AND ALEX RUSSELL: Efficient probabilistically checkable proofs and applications to approximations. In *Proc. 25th STOC*, pp. 294–304. ACM Press, 1993. [doi:10.1145/167088.167174] 497

[6] ELWYN R. BERLEKAMP, ROBERT J. MCELIECE, AND HENK C. A. VAN TILBORG: On the inherent intractability of certain coding problems. *IEEE Trans. Inform. Theory*, 24(3):384–386, 1978. [doi:10.1109/TIT.1978.1055873] 489

[7] JIN-YI CAI AND AJAY NERURKAR: Approximating the SVP to within a factor $(1+1/\dim^{\varepsilon})$ is NP-hard under randomized reductions. *J. Comput. System Sci.*, 59(2):221–239, 1999. Preliminary version in CCC'98. [doi:10.1006/jcss.1999.1649] 488, 489

[8] QI CHENG AND DAQING WAN: A deterministic reduction for the gap minimum distance problem. In *Proc. 41st STOC*, pp. 33–38. ACM Press, 2009. [doi:10.1145/1536414.1536421] 489, 490

[9] JOHN H. CONWAY AND NEIL J. A. SLOANE: *Sphere Packings, Lattices and Groups*. Springer, 3rd edition, 1998. 501, 502, 503

[10] IRIT DINUR, GUY KINDLER, RAN RAZ, AND SHMUEL SAFRA: Approximating CVP to within almost-polynomial factors is NP-hard. *Combinatorica*, 23(2):205–243, 2003. Preliminary version in FOCS'98. [doi:10.1007/s00493-003-0019-y] 488, 489

[11] ILYA DUMER, DANIELE MICCIANCIO, AND MADHU SUDAN: Hardness of approximating the minimum distance of a linear code. *IEEE Trans. Inform. Theory*, 49(1):22–37, 2003. Preliminary version in FOCS'99. [doi:10.1109/TIT.2002.806118] 489, 490, 491, 498, 507, 508

[12] ISHAY HAVIV AND ODED REGEV: Tensor-based hardness of the Shortest Vector Problem to within almost polynomial factors. *Theory of Computing*, 8(23):513–531, 2012. Preliminary version in STOC'07. [doi:10.4086/toc.2012.v008a023] 489, 490, 491, 496, 497, 498, 502, 503, 509

[13] DAVID S. JOHNSON: A catalog of complexity classes. In *Handbook of Theoretical Computer Science*, volume A (Algorithms and Complexity), chapter 2, pp. 67–161. Elsevier, 1990. 489

[14] SUBHASH KHOT: Hardness of approximating the shortest vector problem in lattices. *J. ACM*, 52(5):789–808, 2005. Preliminary version in FOCS'04. [doi:10.1145/1089023.1089027] 489, 490, 491, 496, 497, 498, 502, 503, 508, 509

[15] SUBHASH KHOT: Hardness of approximating the shortest vector problem in high ℓ_p norms. *J. Comput. System Sci.*, 72(2):206–219, 2006. Preliminary version in FOCS'03. [doi:10.1016/j.jcss.2005.07.002] 489

[16] YOSHIYUKI KITAOKA: Tensor products of positive definite quadratic forms. *Proc. Japan Acad.*, 52(9):498–500, 1976. [doi:10.3792/pja/1195518215] 496

[17] JEFFERY C. LAGARIAS AND ANDREW M. ODLYZKO: Solving low-density subset sum problems. *J. ACM*, 32(1):229–246, 1985. Preliminary version in FOCS'83. [doi:10.1145/2455.2461] 488

[18] VADIM LYUBASHEVSKY AND DANIELE MICCIANCIO: On bounded distance decoding, unique shortest vectors, and the minimum distance problem. In *29th Ann. Internat. Cryptology Conf. (CRYPTO'09)*, pp. 577–594. Springer, 2009. [doi:10.1007/978-3-642-03356-8_34] 497

[19] DANIELE MICCIANCIO: The hardness of the closest vector problem with preprocessing. *IEEE Trans. Inform. Theory*, 47(3):1212–1215, 2001. [doi:10.1109/18.915688] 488

[20] DANIELE MICCIANCIO: The shortest vector in a lattice is hard to approximate to within some constant. *SIAM J. Comput.*, 30(6):2008–2035, 2001. Preliminary version in FOCS'98. [doi:10.1137/S0097539700373039] 488, 489, 490, 491, 492, 496, 497, 498, 500, 503, 507, 508

[21] DANIELE MICCIANCIO AND SHAFI GOLDWASSER: *Complexity of Lattice Problems: a cryptographic perspective*. Volume 671 of *The Kluwer International Series in Engineering and Computer Science*. Kluwer Academic Publishers, 2002. 488, 503

[22] DANIELE MICCIANCIO AND PANAGIOTIS VOULGARIS: A deterministic single exponential time algorithm for most lattice problems based on Voronoi cell computations. In *Proc. 42nd STOC*, pp. 351–358. ACM Press, 2010. ECCC. [doi:10.1145/1806689.1806739] 488

[23] DANIELE MICCIANCIO AND PANAGIOTIS VOULGARIS: Faster exponential time algorithms for the shortest vector problem. In *Proc. 21st Ann. ACM-SIAM Symp. on Discrete Algorithms (SODA'10)*, pp. 1468–1480. ACM Press, 2010. ECCC. [ACM:1873720] 488

[24] PHONG Q. NGUYEN AND THOMAS VIDICK: Sieve algorithms for the shortest vector problem are practical. *J. Math. Cryptology*, 2(2):181–207, 2008. [doi:10.1515/JMC.2008.009] 488

[25] VERA S. PLESS AND WILLIAM CARY HUFFMAN, editors. *Handbook of Coding Theory*. North-Holland, 1998. 495

[26] XAVIER PUJOL AND DAMIEN STEHLÉ: Solving the shortest lattice vector problem in time $2^{2.465n}$. Report 2009/605, IACR ePrint archive, 2009. IACR. 488

[27] ODED REGEV AND RICKY ROSEN: Lattice problems and norm embeddings. In *Proc. 38th STOC*, pp. 447–456. ACM Press, 2006. [doi:10.1145/1132516.1132581] 493

[28] NORBERT SAUER: On the density of families of sets. *J. Combin. Theory Ser. A*, 13(1):145–147, 1972. [doi:10.1016/0097-3165(72)90019-2] 498

[29] SAHARON SHELAH: A combinatorial problem; stability and order for models and theories in infinitary languages. *Pacific J. Math.*, 41(1):247–261, 1972. Project Euclid. 498

[30] PETER VAN EMDE BOAS: Another NP-complete partition problem and the complexity of computing short vectors in a lattice. Technical Report 81-04, Mathematische Instituut, University of Amsterdam, 1981. Available at author's home page. 488

[31] VLADIMIR N. VAPNIK AND ALEXEY YA. CHERVONENKIS: On the uniform covergence of relative frequencies of events to their probabilities. *Theory of Probability and Appl.*, XVI(2):264–280, 1971. Translation from the Russian. 498

[32] ALEXANDER VARDY: The intractability of computing the minimum distance of a code. *IEEE Trans. Inform. Theory*, 43(6):1757–1766, 1997. [doi:10.1109/18.641542] 489

[33] XIAOYUN WANG, MINGJIE LIU, CHENGLIANG TIAN, AND JINGGUO BI: Improved Nguyen-Vidick heuristic sieve algorithm for shortest vector problem. In *Proc. 6th ACM Symp. on Information, Computer and Communications Security (ASIACCS'11)*, pp. 1–9. ACM Press, 2011. [doi:10.1145/1966913.1966915] 488

AUTHOR

Daniele Micciancio
Professor
University of California at San Diego (UCSD), La Jolla, CA, USA
daniele@cs.ucsd.edu
`http://cseweb.ucsd.edu/~daniele/`

ABOUT THE AUTHOR

DANIELE MICCIANCIO got his Ph. D. in Computer Science from M.I.T. in 1998, under the supervision of Shafi Goldwasser. He is a full professor in CSE department at the University of California, San Diego (UCSD), where he has worked since 1999. He is broadly interested in theoretical computer science and cryptography. His research focuses on lattice cryptography and the complexity of lattice and coding problems.