58th Annual IEEE Symposium on Foundations of Computer Science

On the Quantitative Hardness of CVP[†]

Huck Bennett and Noah Stephens-Davidowitz
Courant Institute
New York University
New York, USA
Email: huckbennett@gmail.com, noahsd@gmail.com

Alexander Golovnev
Yahoo Research
New York, USA
Email: alexgolovnev@gmail.com

***Abstract*—For odd integers $p \geq 1$ (and $p = \infty$), we show that the Closest Vector Problem in the ℓ_p norm (CVP_p) over rank n lattices cannot be solved in $2^{(1-\varepsilon)n}$ time for any constant $\varepsilon > 0$ unless the Strong Exponential Time Hypothesis (SETH) fails. We then extend this result to "almost all" values of $p \geq 1$, not including the even integers. This comes tantalizingly close to settling the quantitative time complexity of the important special case of CVP_2 (i.e., CVP in the Euclidean norm), for which a $2^{n+o(n)}$-time algorithm is known. In particular, our result applies for any $p = p(n) \neq 2$ that approaches 2 as $n \to \infty$.**

We also show a similar SETH-hardness result for SVP_∞; hardness of approximating CVP_p to within some constant factor under the so-called Gap-ETH assumption; and other hardness results for CVP_p and CVPP_p for any $1 \leq p < \infty$ under different assumptions.

***Keywords*-Lattices; CVP; SETH; Closest Vector Problem; Fine-grained complexity**

I. Introduction

A lattice $\mathcal{L}$ is the set of all integer combinations of linearly independent basis vectors $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n \in \mathbb{R}^d$,

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

We call n the *rank* of the lattice $\mathcal{L}$ and d the *dimension* or the *ambient dimension*.

The two most important computational problems on lattices are the Shortest Vector Problem (SVP) and the Closest Vector Problem (CVP). Given a basis for a lattice $\mathcal{L} \subset \mathbb{R}^d$, SVP asks us to compute the minimal length of a non-zero vector in $\mathcal{L}$, and CVP asks us to compute the distance from some target point $\boldsymbol{t} \in \mathbb{R}^d$ to the lattice. Typically, we define shortness and closeness in terms of the ℓ_p norm for some $1 \leq p \leq \infty$, given by

$$\|\boldsymbol{x}\|_p := (|x_1|^p + |x_2|^p + \cdots + |x_d|^p)^{1/p}$$

for finite p and

$$\|\boldsymbol{x}\|_\infty := \max_{1 \leq i \leq d} |x_i| .$$

In particular, the ℓ_2 norm is the familiar Euclidean norm, and it is by far the best studied in this context. We write SVP_p and CVP_p for the respective problems in the ℓ_p norm.

Starting with the breakthrough work of Lenstra, Lenstra, and Lovász in 1982 [1], algorithms for solving these problems in both their exact and approximate forms have found innumerable applications, including factoring polynomials over the rationals [1], integer programming [2], [3], [4], cryptanalysis [5], [6], [7], etc. More recently, many cryptographic primitives have been constructed whose security is based on the *worst-case* hardness of these or closely related lattice problems [8], [9], [10], [11], [12]. Given the obvious importance of these problems, their complexity is quite well-studied. Below, we survey some of these results. We focus on algorithms for the exact and near-exact problems since these are most relevant to our work and because the best known algorithms for the approximate variants of these problems use algorithms for the exact problems as subroutines [13], [14], [15]. (Many of the results described below are also summarized in Table I-D.)

A. Algorithms for SVP and CVP

The AKS algorithm and its descendants: The current fastest known algorithms for solving SVP_p all use the celebrated randomized sieving technique due to Ajtai, Kumar, and Sivakumar [16]. The original algorithm from [16] was the first $2^{O(n)}$-time algorithm for SVP, and it worked for both $p = 2$ and $p = \infty$.

In the $p = 2$ case, a sequence of works improved upon the constant in the exponent [17], [18],

[†]In this extended abstract, we summarize our results. The full version of this paper is available at http://arxiv.org/abs/1704.03928.

0272-5428/17 $31.00 © 2017 IEEE
DOI 10.1109/FOCS.2017.11

[19], [20], and the current fastest running time of an algorithm that provably solves SVP_2 exactly is $2^{n+o(n)}$ [21].[1] While progress has slowed, this seems unlikely to be the end of the story. Indeed, there are heuristic sieving algorithms that run in time $(3/2)^{n/2+o(n)}$ [17], [22], [23], [24], and there is some reason to believe that the provably correct [21] algorithm can be improved. In particular, there is a provably correct $2^{n/2+o(n)}$-time algorithm that approximates SVP_2 up to a small constant approximation factor [21].

A different line of work extended the randomized sieving approach of [16] to obtain $2^{O(n)}$-time algorithms for SVP in additional norms. In particular, Blömer and Naewe extended it to all ℓ_p norms [25]. Subsequent work extended this further, first to arbitrary symmetric norms [26] and then to the "near-symmetric norms" that arise in integer programming [27].

Finally, a third line of work extended the [16] approach to approximate CVP. Ajtai, Kumar, and Sivakumar themselves showed a $2^{O(n)}$-time algorithm for approximating CVP_2 to within any constant approximation factor strictly greater than one [28]. Blömer and Naewe obtained the same result for all ℓ_p norms [25], and Dadush extended it further to arbitrary symmetric norms and again to "near-symmetric norms" [27]. We stress, however, that none of these results apply to exact CVP, and indeed, there are fundamental barriers to extending these algorithms to exact CVP. (See, e.g., [29].)

Exact algorithms for CVP*:* CVP is known to be at least as hard as SVP (in any norm, under an efficient reduction that preserves the rank and approximation factor) [30], and *exact* CVP appears to be a much more subtle problem than exact SVP.[2] Indeed, progress on exact CVP has been much slower than the progress on exact SVP. Over a decade after [16], Micciancio and Voulgaris presented the first $2^{O(n)}$-time algorithm for exact CVP_2 [31], using elegant new techniques built upon the approach of Sommer, Feder, and Shalvi [32]. Specifically, they achieved a running time of $4^{n+o(n)}$, and subsequent work even showed a running time of $2^{n+o(n)}$ for CVP_2 with Preprocessing (in which the algorithm is allowed access to arbitrary advice that depends on the lattice but not the target vector) [33]. Later, [29] showed a $2^{n+o(n)}$-time algorithm for CVP_2, so that the current best proven asymptotic running time is actually the same for SVP_2 and CVP_2.

However, for $p \neq 2$, progress for exact CVP_p has been minimal. Indeed, the fastest known algorithms for exact CVP_p with $p \neq 2$ are still the $n^{O(n)}$-time enumeration algorithms first developed by Kannan in 1987 [3], [4], [34]. Both algorithms for exact CVP_2 mentioned in the previous paragraph use many special properties of the ℓ_2 norm, and it seems that substantial new ideas would be required to extend them to arbitrary ℓ_p norms.

B. Hardness of SVP and CVP

Van Emde Boas showed the NP-hardness of CVP_p for any p and SVP_∞ in 1981 [35]. Extending this to SVP_p for finite p was a major open problem until it was proven (via a randomized reduction) for all $1 \le p \le \infty$ by Ajtai in 1998 [36]. There has since been much follow-up work, showing the hardness of these problems for progressively larger approximation factors, culminating in NP-hardness of approximating CVP_p up to a factor of $n^{c/\log\log n}$ for some constant $c > 0$ [37], [38] and hardness of SVP_p with the same approximation factor under plausible complexity-theoretic assumptions [39], [40], [41], [42]. These results are nearly the best possible under plausible assumptions, since approximating either problem up to a factor of $\sqrt{n}$ is known to be in $\mathrm{NP} \cap \mathrm{coNP}$ [43], [44], [11].

However, such results only rule out the possibility of polynomial-time algorithms (under reasonable complexity-theoretic assumptions). They say very little about the *quantitative* hardness of these problems for a fixed lattice rank n.[3]

This state of affairs is quite frustrating for two reasons. First, in the specific case of CVP_2, algorithmic progress has reached an apparent barrier. In particular, both known techniques for solving exact CVP_2 in singly exponential time are fundamentally unable to produce algorithms whose running time

[1] The algorithm in [21] is quite a bit different than the other algorithms in this class, but it can still be thought of as a sieving algorithm.

[2] In particular, there can be arbitrarily many lattice points that are approximate closest vectors, which makes sieving techniques seemingly useless for solving exact CVP. (See, e.g., [29] for a discussion of this issue.) We note, however, that hardness results (including ours) tend to produce CVP instances with a bounded number of approximate closest vectors (e.g., $2^{O(n)}$).

[3] One can derive certain quantitative hardness results from known hardness proofs, but in most cases the resulting lower bounds are quite weak. The one true quantitative hardness result known prior to this work was an unpublished result due to Samuel Yeom, showing that CVP cannot be solved in time $2^{10^{-4}n}$ under plausible complexity-theoretic assumptions [45].

is asymptotically better than the current best of $2^{n+o(n)}$ [31], [29].[4] Second, some lattice-based cryptographic constructions are close to deployment [46], [47], [48]. In order to be practically secure, these constructions require the quantitative hardness of certain lattice problems, and so their designers rely on quantitative hardness assumptions [49]. If, for example, there existed a $2^{n/20}$-time algorithm for SVP_p or CVP_p, then these cryptographic schemes would be insecure in practice.

We therefore move in a different direction. Rather than trying to extend non-quantitative hardness results to larger approximation factors, we show quantitative hardness results for exact (or nearly exact) problems. To do this, we use the tools of *fine-grained complexity*.

C. *Fine-grained complexity*

Impagliazzo and Paturi [50] introduced the *Exponential Time Hypothesis* (ETH) and the *Strong Exponential Time Hypothesis* (SETH) to help understand the precise hardness of k-SAT. Informally, ETH asserts that 3-SAT takes $2^{\Omega(n)}$-time to solve in the worst case, and SETH asserts that k-SAT takes essentially 2^n-time to solve for unbounded k. I.e., SETH asserts that brute-force search is essentially optimal for solving k-SAT for large k.

Recently, the study of fine-grained complexity has leveraged ETH, SETH, and several other assumptions to prove quantitative hardness results about a wide range of problems. These include both problems in P (see, e.g., [51], [52], [53] and the survey by Vassilevska Williams [54]), and NP-hard problems (see, e.g., [55], [56], [57]). Although these results are all conditional, they help to explain *why* making further algorithmic progress on these problems is difficult—and suggest that it might be impossible. Namely, any non-trivial algorithmic improvement would disprove a very well-studied hypothesis.

One proves quantitative hardness results using *fine-grained* reductions (see [54] for a formal definition). For example, there is a mapping from k-SAT formulas on n variables to Hitting Set instances with universes of n elements [56]. This reduction is fine-grained in the sense that for any constant $\varepsilon > 0$, a $2^{(1-\varepsilon)n}$-time algorithm for Hitting Set implies a $2^{(1-\varepsilon)n}$-time algorithm for k-SAT, breaking SETH.

Despite extensive effort, no faster-than-2^n-time algorithm for k-SAT with unbounded k has been found. Nevertheless, there is no consensus on whether SETH is true or not, and recently, Williams [58] refuted a very strong variant of SETH. This makes it desirable to base quantitative hardness results on weaker assumptions when possible, and indeed our main result holds even assuming a weaker variant of SETH based on the hardness of Weighted Max-k-SAT (except for the case of $p = \infty$).

[4] Both techniques require short vectors in each of the 2^n cosets of $\mathcal{L}$ mod $2\mathcal{L}$ (though for apparently different reasons).

Problem	Upper Bound	Lower Bounds			
		SETH	Max-2-SAT	ETH	Gap-ETH
CVP_p	$n^{O(n)}$ $(2^{O(n)})$	2^n	$2^{\omega n/3}$	$2^{\Omega(n)}$	$2^{\Omega(n)}*$
CVP_2	2^n	—	$2^{\omega n/3}$	$2^{\Omega(n)}$	$2^{\Omega(n)}*$
SVP_∞	$2^{O(n)}$	2^n*	—	$2^{\Omega(n)}$	$2^{\Omega(n)}*$
CVPP_p	$n^{O(n)}$ $(2^{O(n)})$	—	$2^{\Omega(\sqrt{n})}$	$2^{\Omega(\sqrt{n})}$	—

Figure 1. Summary of known quantitative upper and lower bounds, with new results in blue. **The first lower bound holds for "almost all $p \geq 1$, as in Theorem I.2**. Upper bounds in parentheses hold for any constant approximation factor strictly greater than one, and lower bounds with a $*$ apply for some constant approximation factor strictly greater than one. ω is the matrix multiplication exponent, satisfying $2 \leq \omega < 2.373$. We have suppressed smaller factors.

D. *Our contribution*

We now enumerate our results. See also Table I-D.

SETH-hardness of CVP_p*:* Our main result is the SETH-hardness of CVP_p for any odd integer $p \geq 1$ and $p = \infty$ (and SVP_∞). Formally, we prove the following.

Theorem I.1. *For any constant integer $k \geq 2$ and any odd integer $p \geq 1$ or $p = \infty$, there is an efficient reduction from k-SAT with n variables and m clauses to CVP_p (or SVP_∞) on a lattice of rank n (with ambient dimension $n + O(m)$).*

In particular, there is no $2^{(1-\varepsilon)n}$-time algorithm for CVP_p for any odd integer $p \geq 1$ or $p = \infty$ (or SVP_∞) and any constant $\varepsilon > 0$ unless SETH is false.

Unfortunately, we are unable to extend this result to even integers p, and in particular, to the important special case of $p = 2$. In fact, this is inherent, as we show that our approach necessarily fails for even integers $p \leq k - 1$. In spite of this, we actually prove the following result that generalizes Theorem I.1 to "almost all" $p \geq 1$ (including non-integer p).

Theorem I.2. *For any constant integer $k \geq 2$, there is an efficient reduction from k-SAT with n variables and m clauses to CVP_p on a lattice of rank n (with ambient dimension $n + O(m)$) for any $p \geq 1$ such that*

1) *p is an odd integer or $p = \infty$;*

2) *$p \notin S_k$, where S_k is some finite set (containing all even integers $p \leq k-1$); or*
3) *$p = p_0 + \delta(n)$ for any $p_0 \geq 1$ and any $\delta(n) \neq 0$ that converges to zero as $n \to \infty$.*

In particular, if SETH holds then for any constant $\varepsilon > 0$, there is no $2^{(1-\varepsilon)n}$-time algorithm for CVP_p for any $p \geq 1$ such that

1) *p is an odd integer or $p = \infty$;*
2) *$p \notin S_k$ for some sufficiently large k (depending on ε); or*
3) *$p = p_0 + \delta(n)$.*

Notice that this lower bound (Theorem I.2) comes tantalizingly close to resolving the quantitative complexity of CVP_2. In particular, we obtain a 2^n-time lower bound on $\mathrm{CVP}_{2+\delta}$ for any $\delta(n) = o(1)$, and the fastest algorithm for CVP_2 run in time $2^{n+o(n)}$. But, formally, Theorems I.1 and I.2 say nothing about CVP_2. (Indeed, there is at least some reason to believe that CVP_2 is easier than CVP_p for $p \neq 2$ [59].)

We note that our reductions actually work for Weighted Max-k-SAT for all finite $p \neq \infty$, so that our hardness results holds under a weaker assumption than SETH, namely, the corresponding hypothesis for Weighted Max-k-SAT.

Finally, we note that in the special case of $p = \infty$, our reduction works even for approximate CVP_∞, or even approximate SVP_∞, with an approximation factor of $\gamma := 1 + 2/(k-1)$. In particular, γ is constant for fixed k. This implies that for every constant $\varepsilon > 0$, there is a $\gamma_\varepsilon > 1$ such that no $2^{(1-\varepsilon)n}$-time algorithm approximates SVP_∞ or CVP_∞ to within a factor of γ_ε unless SETH fails.

Quantitative hardness of approximate CVP*:* As we discussed above, many $2^{O(n)}$-time algorithms for CVP_p only work for γ-approximate CVP_p for constant approximation factors $\gamma > 1$. However, the reduction described above only works for *exact* CVP_p (except when $p = \infty$), or at best for γ-approximate CVP_p with some approximation factor $\gamma = 1 + o(1)$. (Standard techniques for "boosting" the approximation factor are useless for us because they increase the rank quite a bit.)

So, it would be preferable to show hardness for some constant approximation factor $\gamma > 1$. One way to show such a hardness result is via a fine-grained reduction from the problem of approximating Max-k-SAT to within a constant factor. Indeed, in the $k = 2$ case, we show that such a reduction exists, so that there is no $2^{o(n)}$-time algorithm for approximating CVP_p to within some constant factor unless a $2^{o(n)}$-time algorithm exists for approximating Max-2-SAT. We also note that a $2^{o(n)}$-time algorithm for approximating Max-2-SAT to within a constant factor would imply one for Max-3-SAT as well.[5]

We present this result informally here (without worrying about specific parameters and the exact definition of approximate Max-2-SAT). See the full version [60] for the formal statement.

Theorem I.3. *There is an efficient reduction from approximating a Max-2-SAT on n variables and m clauses to within a constant factor to approximating CVP_p to within a constant factor on a lattice of rank n (with ambient dimension $n + O(m)$) for any $p \geq 1$.*

Quantitative hardness of CVP *with Preprocessing. :* CVP with Preprocessing (CVPP) is the variant of CVP in which we are allowed arbitrary advice that depends on the lattice, but not the target vector. CVPP and its variants have potential applications in both cryptography (e.g., [10]) and cryptanalysis. And, an algorithm for CVPP_2 is used as a subroutine in the celebrated Micciancio-Voulgaris algorithm for CVP_2 [31], [33]. The complexity of CVPP_p is well studied, with both hardness of approximation results [61], [62], [63], [64], [65], and efficient approximation algorithms [44], [66].

We prove the following quantitative hardness result for CVPP_p.

Theorem I.4. *For any $1 \leq p < \infty$, there is no $2^{o(\sqrt{n})}$-time algorithm for* CVPP *unless there is a (non-uniform) $2^{o(n)}$-time algorithm for Max-2-SAT. In particular, no such algorithm exists unless (non-uniform) ETH fails.*

Additional quantitative hardness results for CVP_p*:* We also observe the following weaker hardness result for CVP_p for any $1 \leq p < \infty$ based on different assumptions. The ETH-hardness of CVP_p was already known in folklore, and even written down by Samuel Yeom in unpublished work [45]. We present a slightly stronger theorem than what was previously known, showing a reduction from Max-2-SAT on n variables to CVP_p on a lattice of rank n. (Prior to this work, we were only aware of reductions from 3-SAT on n variables to CVP_p on a lattice of rank Cn for some very large positive constant C.)

Theorem I.5. *For any $1 \leq p < \infty$, there is an efficient reduction from Max-2-SAT with n variables*

[5] Recall that, while there is a polynomial-time algorithm for 2-SAT, Max-2-SAT is hard.

to CVP_p *on a lattice of rank* n *(and dimension* $n+m$*, where* m *is the number of clauses).*

In particular, for any constant $c > 0$*, there is no* $(\mathrm{poly}(n) \cdot 2^{cn})$*-time algorithm for* CVP_p *unless there is a similar algorithm for Max-2-SAT, and there is no* $2^{o(n)}$*-time algorithm for* CVP_p *unless ETH fails.*

The fastest known algorithm for the Max-2-SAT problem is the $\mathrm{poly}(n) \cdot 2^{\omega n/3}$-time algorithm due to Williams [67], where $2 \leq \omega < 2.373$ is the matrix multiplication exponent [68], [69]. This implies that a faster than $2^{\omega n/3}$-time algorithm for CVP_p (and CVP_2 in particular) would yield a faster algorithm for Max-2-SAT.[6] (See, e.g., [70] Open Problem 4.7 and the preceding discussion.)

E. Techniques

Max-2-SAT: We first show a straightforward reduction from Max-2-SAT to CVP_p for any $1 \leq p < \infty$. I.e., we prove Theorem I.5. This simple reduction will introduce some of the high-level ideas needed for our more difficult reductions.

Given a Max-2-SAT instance Φ with n variables and m clauses, we construct the lattice basis

$$B := \begin{pmatrix} \bar{\Phi} \\ 2\alpha I_n \end{pmatrix} , \tag{1}$$

where $\alpha > 0$ is some very large number and $\bar{\Phi} \in \mathbb{R}^{m \times n}$, where

$$\bar{\Phi}_{i,j} := \begin{cases} 2 & \text{if } x_j \text{ is in the } i\text{th clause,} \\ -2 & \text{if } \neg x_j \text{ is in the } i\text{th clause,} \\ 0 & \text{otherwise .} \end{cases} \tag{2}$$

I.e., the rows of $\bar{\Phi}$ correspond to clauses and the columns correspond to variables. Each entry encodes whether the relevant variable is included in the relevant clause unnegated, negated, or not at all, using 2, -2, and 0 respectively. (We assume without loss of generality that no clause contains repeated literals or a literal and its negation simultaneously.) The target $\boldsymbol{t} \in \mathbb{R}^{m+n}$ is given by

$$\boldsymbol{t} := (t_1, t_2, \ldots, t_m, \alpha, \alpha, \ldots, \alpha)^T , \tag{3}$$

where

$$t_i := 3 - 2\eta_i , \tag{4}$$

where η_i is the number of negated variables in the ith clause.

[6] This also implies that a polynomial-space algorithm with running time $2^{(1-\varepsilon)\cdot n}$ for CVP_p would beat the current fastest such algorithm for Max-2-SAT, a long-standing open problem. All known algorithms for CVP or SVP that run in $2^{O(n)}$ time require exponential space, and it is a major open problem to find a polynomial-space, singly exponential-time algorithm.

Notice that the copy of $2\alpha I_n$ at the bottom of B together with the sequence of α's in the last coordinates of $\boldsymbol{t}$ guarantee that any lattice vector $B\boldsymbol{z}$ with $\boldsymbol{z} \in \mathbb{Z}^n$ is at distance at least $\alpha n^{1/p}$ away from $\boldsymbol{t}$. Furthermore, if $\boldsymbol{z} \notin \{0,1\}^n$, then this distance increases to at least $\alpha(n-1+3^p)^{1/p}$. This is a standard gadget, which will allow us to ignore the case $\boldsymbol{z} \notin \{0,1\}^n$ (as long as α is large enough). I.e., we can view $\boldsymbol{z}$ as an assignment to the n variables of Φ.

Now, suppose $\boldsymbol{z}$ does not satisfy the ith clause. Then, notice that the ith coordinate of $B\boldsymbol{z}$ will be exactly $-2\eta_i$, so that $(B\boldsymbol{z} - \boldsymbol{t})_i = 0 - 3 = -3$. If, on the other hand, exactly one literal in the ith clause is satisfied, then the ith coordinate of $B\boldsymbol{z}$ will be $2-2\eta_i$, so that $(B\boldsymbol{z}-\boldsymbol{t})_i = 2-3 = -1$. Finally, if both literals are satisfied, then the ith coordinate will be $4-2\eta_i$, so that $(B\boldsymbol{z}-\boldsymbol{t})_i = 4-3 = 1$. In particular, if the clause is not satisfied, then $|(B\boldsymbol{z})_i - t_i| = 3$. Otherwise, $|(B\boldsymbol{z})_i - t_i| = 1$.

It follows that the distance to the target is exactly $\mathrm{dist}_p(\boldsymbol{t}, \mathcal{L})^p = \alpha^p n + S + 3^p(m-S) = \alpha^p n - (3^p-1)S + 3^p m$, where S is the maximal number of satisfied clauses on the input. So, the distance $\mathrm{dist}_p(\boldsymbol{t}, \mathcal{L})$ tells us exactly the maximum number of satisfiable clauses, which is what we needed.

Difficulties extending this to k-SAT: The above reduction relied on one very important fact: that $|4-3| = |2-3| < |0-3|$. In particular, a 2-SAT clause can be satisfied in two different ways; either one variable is satisfied or two variables are satisfied. We designed our CVP instance above so that the ith coordinate of $B\boldsymbol{z} - \boldsymbol{t}$ is $4-3$ if two literals in the ith clause are satisfied by $\boldsymbol{z} \in \{0,1\}^n$, $2-3$ if one literal is satisfied, and $0-3$ if the clause is unsatisfied. Since $|4-3| = |2-3|$, the "contribution" of this ith coordinate to the distance $\|B\boldsymbol{z}-\boldsymbol{t}\|_p^p$ is the same for any satisfied clause. Since $|0-3| > |4-3|$, the contribution to the ith coordinate is larger for unsatisfied clauses than satisfied clauses.

Suppose we tried the same construction for a k-SAT instance. I.e., suppose we take $\bar{\Phi} \in \mathbb{R}^{m \times n}$ to encode the literals in each clause as in Eq. (2) and construct our lattice basis B as in Eq. (1) and target $\boldsymbol{t}$ as in Eq. (3), perhaps with the number 3 in the definition of $\boldsymbol{t}$ replaced by an arbitrary $t^* \in \mathbb{R}$. Then, the ith coordinate of $B\boldsymbol{z}-\boldsymbol{t}$ would be $2S_i - t^*$, where S_i is the number of literals satisfied in the ith clause.

No matter how cleverly we choose $t^* \in \mathbb{R}$, some satisfied clauses will contribute more to the distance than others as long as $k \geq 3$. I.e., there will always be some "imbalance" in this contribution. As a result, we will not be able to distinguish between, e.g., an assignment that satisfies all clauses but has S_i far from $t^*/2$ for all i and an assignment that satisfies fewer clauses but has $S_i \approx t^*/2$ whenever i corresponds to a satisfying clause.

In short, for $k \geq 3$, we run into trouble because satisfying assignments to a clause may satisfy anywhere between 1 and k literals, but k distinct numbers obviously cannot all be equidistant from some number t^*. (See the full version [60] for a simple way to get around this issue by adding to the rank of the lattice. Below, we show a more technical way to do this without adding to the rank of the lattice, which allows us to prove SETH-hardness.)

A solution via isolating parallelepipeds: To get around the issue described above for $k \geq 3$, we first observe that, while many distinct *numbers* cannot all be equidistant from some *number* t^*, it is trivial to find many distinct *vectors* in $\mathbb{R}^{d^*}$ that are equidistant from some *vector* $\boldsymbol{t}^* \in \mathbb{R}^{d^*}$.

We therefore consider modifying the reduction from above by replacing the scalar ± 1 values in our matrix $\bar{\Phi}$ with vectors in $\mathbb{R}^{d^*}$ for some d^*. In particular, for some vectors $V = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_k) \in \mathbb{R}^{d^* \times k}$, we define $\bar{\Phi} \in \mathbb{R}^{d^* m \times n}$ as

$$\bar{\Phi}_{i,j} := \begin{cases} \boldsymbol{v}_s & x_j \text{ is } s\text{th literal in } i\text{th clause,} \\ -\boldsymbol{v}_s & \neg x_j \text{ is } s\text{th literal in } i\text{th clause,} \\ \boldsymbol{0}_d & \text{otherwise ,} \end{cases} \tag{5}$$

where we have abused notation and taken $\bar{\Phi}_{i,j}$ to be a column vector in d^* dimensions. By defining $\boldsymbol{t} \in \mathbb{R}^{d^* m + n}$ appropriately,[7] we will get that the "contribution of the ith clause to the distance" $\|B\boldsymbol{z} - \boldsymbol{t}\|_p^p$ is exactly $\|V\boldsymbol{y} - \boldsymbol{t}^*\|_p^p$ for some $\boldsymbol{t}^* \in \mathbb{R}^{d^*}$, where $\boldsymbol{y} \in \{0,1\}^k$ such that $y_s = 1$ if and only if $\boldsymbol{z}$ satisfies the sth literal of the relevant clause. (See Table 4 for a diagram showing the output of the reduction and Theorem II.2 for the formal statement.) We stress that, while we have increased the *ambient dimension* by nearly a factor of d^*, the *rank* of the lattice is still n.

This motivates the introduction of our primary technical tool, which we call *isolating parallelepipeds*. For $1 \leq p \leq \infty$, a (p,k)-isolating parallelepiped is represented by a matrix $V \in \mathbb{R}^{d^* \times k}$ and a shift vector $\boldsymbol{t}^* \in \mathbb{R}^{d^*}$ with the special property that one vertex of the parallelepiped $V\{0,1\}^k - \boldsymbol{t}^*$ is "isolated." (Here, $V\{0,1\}^k - \boldsymbol{t}^*$ is an affine transformation of the hypercube, i.e., a parallelepiped.) In particular, every vertex of the parallelepiped, $V\boldsymbol{y} - \boldsymbol{t}^*$ for $\boldsymbol{y} \in \{0,1\}^k$ has unit length $\|V\boldsymbol{y} - \boldsymbol{t}^*\|_p = 1$ *except for the vertex* $-\boldsymbol{t}^*$, which is longer, i.e., $\|\boldsymbol{t}^*\|_p > 1$. (See Figure 2.)

In terms of the reduction above, an isolating parallelepiped is exactly what we need. In particular, if we plug V and $\boldsymbol{t}^*$ into the above reduction, then all satisfied clauses (which correspond to non-zero $\boldsymbol{y}$ in the above description) will "contribute" 1 to the distance $\|B\boldsymbol{z} - \boldsymbol{t}\|_p^p$, while unsatisfied clauses (which correspond to $\boldsymbol{y} = \boldsymbol{0}$) will contribute $1 + \delta$ for some $\delta > 0$. Therefore, the total distance will be exactly $\|B\boldsymbol{z} - \boldsymbol{t}\|_p^p = \alpha^p n + m^+(\boldsymbol{z}) + (m - m^+(\boldsymbol{z}))(1+\delta) = \alpha^p n - \delta m^+(\boldsymbol{z}) + (1+\delta)m$, where $m^+(\boldsymbol{z})$ is the number of clauses satisfied by $\boldsymbol{z}$. So, the distance $\mathrm{dist}_p(\boldsymbol{t}, \mathcal{L})$ exactly corresponds to the maximal number of satisfied clauses, as needed.

Constructing isolating parallelepipeds: Of course, in order for the above to be useful, we must show how to construct these (p,k)-isolating parallelepipeds. In this extended abstract, we only show the reduction that assumes the existence of such objects (see Section II). In the full version, we show how to construct these objects. Indeed, it is not hard to find constructions for all $p \geq 1$ when $k = 2$, and even for all k in the special case when $p = 1$ (see Figure 2). Some other fairly nice examples can also be found for small k, as shown in Figure 3. For $p > 1$ and large k, these objects seem to be much harder to find. (In fact, in the full version we show that there is no (p,k)-isolating parallelepiped for any even integer $p \leq k - 1$.) Our solution is therefore a bit technical.

At a high level, we consider a natural class of parallelepipeds $V \in \mathbb{R}^{2^k \times k}, \boldsymbol{t}^* \in \mathbb{R}^{2^k}$ parametrized by some weights $\alpha_0, \alpha_1, \ldots, \alpha_k \geq 0$ and a scalar shift $t^* \in \mathbb{R}$. These parallelepipeds are constructed so that the length of the vertex $\|V\boldsymbol{y} - \boldsymbol{t}^*\|_p^p$ for $\boldsymbol{y} \in \{0,1\}^k$ depends only on the Hamming weight of $\boldsymbol{y}$ and is linear in the α_i for fixed t^*. In other words, there is a matrix $M_k(p, t^*) \in \mathbb{R}^{(k+1)\times(k+1)}$ such that $M_k(p,t^*)(\alpha_0, \ldots, \alpha_k)^T$ encodes the value of $\|V\boldsymbol{y} - \boldsymbol{t}^*\|_p^p$ for each possible Hamming weight of $\boldsymbol{y} \in \{0,1\}^k$.

We show that, in order to find weights $\alpha_0, \ldots, \alpha_k \geq 0$ such that V and $\boldsymbol{t}^*$ define a (p,k)-isolating parallelepiped, it suffices to find a t^* such that $M_k(p,t^*)$ is invertible. For each odd

[7] In particular, we replace the scalars t_i in Eq. (4) with vectors

$$\boldsymbol{t}_i := \boldsymbol{t}^* - \sum \boldsymbol{v}_s \in \mathbb{R}^{d^*} \ ,$$

where the sum is over s such that the sth literal in the ith clause is negated.

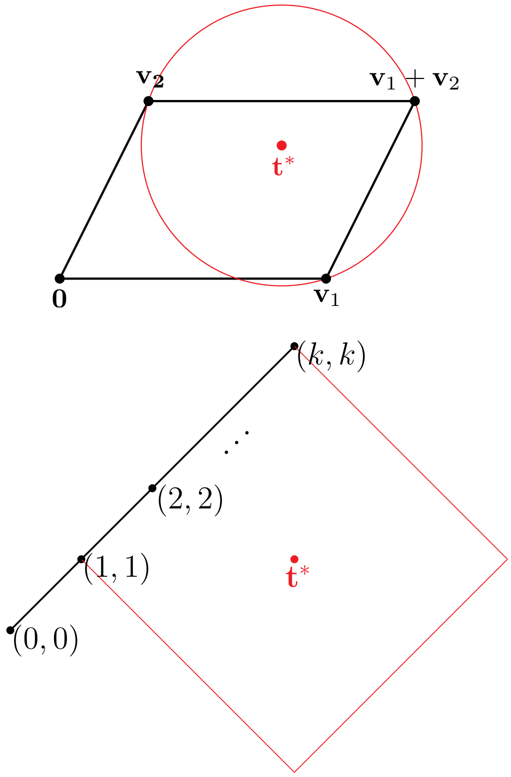

Figure 2. (p, k)-isolating parallelepipeds for $p = 2, k = 2$ (left) and $p = 1, k \geq 1$ (right). On the left, the vectors $\boldsymbol{v}_1$, $\boldsymbol{v}_2$, and $\boldsymbol{v}_1 + \boldsymbol{v}_2$ are all at the same distance from $\boldsymbol{t}^*$, while $\mathbf{0}$ is strictly farther away. On the right is the degenerate parallelepiped generated by k copies of the vector $(1, 1)$. The vectors (i, i) are all at the same ℓ_1 distance from $\boldsymbol{t}^*$ for $1 \leq i \leq m$, while $(0, 0)$ is strictly farther away. The (scaled) unit balls centered at $\boldsymbol{t}^*$ are shown in red, while the parallelepipeds are shown in black.

$$V := \frac{1}{2 \cdot 12^{1/3}} \cdot \begin{pmatrix} 12^{1/3} & 12^{1/3} & 12^{1/3} \\ 1 & 1 & -1 \\ 1 & -1 & 1 \\ 1 & -1 & -1 \\ -1 & 1 & 1 \\ -1 & 1 & -1 \\ -1 & -1 & 1 \end{pmatrix},$$

$$\boldsymbol{t}^* := \frac{1}{2 \cdot 12^{1/3}} \cdot \begin{pmatrix} 2 \cdot 12^{1/3} \\ 2 \\ 2 \\ 2 \\ 2 \\ 2 \\ 2 \end{pmatrix}.$$

Figure 3. A $(3, 3)$-isolating parallelepiped in seven dimensions. One can verify that $\|V\boldsymbol{y} - \boldsymbol{t}^*\|_3^3 = 1$ for all non-zero $\boldsymbol{y} \in \{0, 1\}^3$, and $\|\boldsymbol{t}^*\|_3^3 = 3/2$.

integer $p \geq 1$ and each $k \geq 2$, we show how to explicitly find such a t^*.

To extend this result to other $p \geq 1$, we consider the determinant of $M_k(p, t^*)$ for fixed k and t^*, viewed as a function of p. We observe that this function has a rather nice form—it is a Dirichlet polynomial. I.e., for fixed t^* and k, the determinant can be written as $\sum \exp(a_i p)$ for some $a_i \in \mathbb{R}$. Such a function has finitely many roots unless it is identically zero. So, we take the value of t^* from above such that, say, $M_k(1, t^*)$ is invertible. Since $M_k(1, t^*)$ does not have zero determinant, the Dirichlet polynomial corresponding to $\det(M_k(p, t^*))$ cannot be identically zero and therefore has finitely many roots. This is how we prove Theorem I.2.

Extension to constant-factor approximation: In order to extend our hardness results to approximate CVP_p, we can try simply using the same reduction with k-SAT replaced by Gap-k-SAT. Unfortunately, this does not quite work. Indeed, it is easy to see that the "identity matrix gadget" that we use to restrict our attention to lattice vectors whose coordinates are in $\{0, 1\}$ (Eq. (1)) cannot tolerate an approximation factor larger than $1 + O(1/n)$ (for finite p).

However, we observe that when $k = 2$, this identity matrix gadget is actually unnecessary. In particular, even without this gadget, it "never helps" to consider a lattice vector whose coordinates are not all in $\{0, 1\}$. It then follows immediately from the analysis above that Gap-2-SAT reduces to approximate CVP_p with a constant approximation factor strictly greater than one. We note in passing that we do not know how to extend this result to larger $k > 2$ (except when $p = 1$). We show that the case $k = 2$ is sufficient for proving Gap-ETH-hardness, but we suspect that one can just "remove the identity matrix gadget" from all of our reductions for finite p. If this were true, it would show Gap-ETH-hardness of approximation for slightly larger constant approximation factors and imply even stronger hardness results under less common assumptions.

F. Open questions

The most important question that we leave open is the extension of our SETH-hardness result to arbitrary $p \geq 1$. In particular, while our result applies to $p = p(n) \neq 2$ that approaches 2 asymptotically, it does not apply to the specific case $p = 2$. An extension to $p = 2$ would settle the time complexity of CVP_2 up to a factor of $2^{o(n)}$ (assuming SETH).

However, we know that our technique does not work in this case (in that $(2,k)$-parallelepipeds do not exist for $k \geq 3$), so substantial new ideas might be needed to resolve this issue.

In a different direction, one might try to prove quantitative hardness results for SVP_p. While our SETH-hardness result does apply to SVP_∞, we do not even have ETH-hardness of SVP_p for finite p. Any such result would be a major breakthrough in understanding the complexity of lattice problems, with relevance to cryptography as well as theoretical computer science.

Finally, we note that our main reduction constructs lattices of *rank* n, but the ambient dimension d can be significantly larger. (Specifically, $d = n + O(m)$, where m is the number of clauses in the relevant SAT instance, and where the hidden constant depends on k and can be very large.) Lattice problems are typically parameterized in terms of the rank of the lattice (and for the ℓ_2 norm, one can assume without loss of generality that $d = n$), but it is still interesting to ask whether we can reduce the ambient dimension d.

Acknowledgments

We would like to thank Oded Regev for many fruitful discussions and for helpful comments on an earlier draft of this work. We also thank Vinod Vaikuntanathan for recommending that we consider Gap-ETH.

II. SETH-hardness from isolating parallelepipeds

Here, we show a reduction from instances of weighted Max-k-SAT on formulas with n variables to instances of CVP_p with rank n for all p that uses a certain geometric object, which we define next. Let $\mathbf{1}_n$ and $\mathbf{0}_n$ denote the all 1s and all 0s vectors of length n respectively, and let I_n denote the $n \times n$ identity matrix.

Definition II.1. *For any $1 \leq p \leq \infty$ and integer $k \geq 2$, we say that $V \in \mathbb{R}^{d^* \times k}$ and $\boldsymbol{t}^* \in \mathbb{R}^{d^*}$ define a* (p,k)-isolating parallelepiped *if $\|\boldsymbol{t}\|_p > 1$ and $\|V\boldsymbol{x} - \boldsymbol{t}^*\|_p = 1$ for all $\boldsymbol{x} \in \{0,1\}^k \setminus \{\mathbf{0}_k\}$.*

In order to give the reduction, we first introduce some notation related to SAT. Let Φ be a k-SAT formula on n variables $x_1, \dots, x_n$ and m clauses $C_1, \dots, C_m$. Let $\mathrm{ind}(\ell)$ denote the index of the variable underlying a literal ℓ. I.e., $\mathrm{ind}(\ell) = j$ if $\ell = x_j$ or $\ell = \neg x_j$. Call a literal ℓ *positive* if $\ell = x_j$ and *negative* if $\ell = \neg x_j$ for some variable x_j. Given a clause $C_i = \vee_{s=1}^k \ell_{i,s}$, let $P_i := \{s \in [k] : \ell_{i,s} \text{ is positive}\}$ and let $N_i := \{s \in [k] : \ell_{i,s} \text{ is negative}\}$ denote the indices of positive and negative literals in C_i respectively. Given an assignment $\boldsymbol{a} \in \{0,1\}^n$ to the variables of Φ, let $S_i(\boldsymbol{a})$ denote the indices of literals in C_i satisfied by $\boldsymbol{a}$. I.e., $S_i(\boldsymbol{a}) := \{s \in P_i : a_{\mathrm{ind}(\ell_{i,s})} = 1\} \cup \{s \in N_i : a_{\mathrm{ind}(\ell_{i,s})} = 0\}$. Finally, let $m^+(\boldsymbol{a})$ denote the number of clauses of Φ satisfied by the assignment $\boldsymbol{a}$, i.e., the number of clauses i for which $|S_i(\boldsymbol{a})| \geq 1$.

$$
\begin{array}{c|ccccc|c}
 & x_1 & x_2 & \cdots & x_{n-1} & x_n & \\
\hline
C_1\left\{\begin{matrix} \\ \\ \end{matrix}\right. & \boldsymbol{v}_1 & \boldsymbol{v}_2 & \cdots & \mathbf{0}_{d^*} & -\boldsymbol{v}_3 & \boldsymbol{t}^* - \boldsymbol{v}_3 \\
\vdots & \vdots & \cdots & \ddots & \vdots & \vdots & \vdots \\
C_m\left\{\begin{matrix} \\ \\ \end{matrix}\right. & \mathbf{0}_{d^*} & -\boldsymbol{v}_1 & \cdots & \boldsymbol{v}_2 & \boldsymbol{v}_3 & \boldsymbol{t}^* - \boldsymbol{v}_1 \\
\hline
x_1 & 2\alpha^{1/p} & 0 & \cdots & 0 & 0 & \alpha^{1/p} \\
x_2 & 0 & 2\alpha^{1/p} & \cdots & 0 & 0 & \alpha^{1/p} \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\
x_{n-1} & 0 & 0 & \cdots & 2\alpha^{1/p} & 0 & \alpha^{1/p} \\
x_n & 0 & 0 & \cdots & 0 & 2\alpha^{1/p} & \alpha^{1/p} \\
\hline
 & & & B & & & \boldsymbol{t}
\end{array}
$$

Figure 4. A basis B and target vector $\boldsymbol{t}$ output by the reduction from Theorem II.2 with some $(p,3)$-isolating parallelepiped given by $V = (\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3) \in \mathbb{R}^{d^* \times 3}$ and $\boldsymbol{t}^* \in \mathbb{R}^{d^*}$. In this example, the first clause is $C_1 \equiv x_1 \vee x_2 \vee \neg x_n$ and the mth clause is $C_m \equiv \neg x_2 \vee x_{n-1} \vee x_n$. By the definition of an isolating parallelepiped (Definition II.1), the contribution of the first d coordinates to the distance $\|B\boldsymbol{z} - \boldsymbol{t}\|_p^p$ will be 1 for any assignment $\boldsymbol{z} \in \{0,1\}^n$ satisfying C_1, while non-satisfying assignments contribute $(1+\delta)$ for some $\delta > 0$. For example, if $z_1 = 1, z_2 = 0, z_n = 1$, the clause C_1 is satisfied, and the first d coordinates will contribute $\|\boldsymbol{v}_1 - \boldsymbol{v}_3 - (\boldsymbol{t}^* - \boldsymbol{v}_3)\|_p^p = \|\boldsymbol{v}_1 - \boldsymbol{t}^*\|_p^p = 1$. On the other hand, if $z_1 = 0, z_2 = 0, z_n = 1$, then C_1 is not satisfied, and $\|-\boldsymbol{v}_3 - (\boldsymbol{t}^* - \boldsymbol{v}_3)\|_p^p = \|\boldsymbol{t}^*\|_p^p = 1 + \delta$.

Theorem II.2. *If there exists a computable (p,k)-isolating parallelepiped for some $p = p(n) \in [1,\infty)$ and integer $k \geq 2$, then there exists a polynomial-time reduction from any (weighted-)Max-k-SAT instance with n variables to a CVP_p instance of rank n.*

Proof: For simplicity, we give a reduction from unweighted Max-k-SAT, and afterwards sketch how to modify our reduction to handle the weighted case as well. Namely, we give a reduction from any Max-k-SAT instance (Φ, W) to an instance $(B, \boldsymbol{t}^*, r)$ of CVP_p. Here, the formula Φ is on n variables $x_1, \dots, x_n$ and m clauses $C_1, \dots, C_m$. (Φ, W) is a 'YES' instance if there exists an assignment $\boldsymbol{a}$ such that $m^+(\boldsymbol{a}) \geq W$.

By assumption, there exist computable $d^* = d^*(p,k) \in \mathbb{Z}^+$, $V = [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_k] \in \mathbb{R}^{d^* \times k}$, and $\boldsymbol{t}^* \in \mathbb{R}^{d^*}$ such that $\|\boldsymbol{t}^*\|_p = (1+\delta)^{1/p}$ for some $\delta > 0$ and $\|V\boldsymbol{z} - \boldsymbol{t}^*\|_p = 1$ for all $\boldsymbol{z} \in \{0,1\}^k \setminus \{\boldsymbol{0}_k\}$.

We define the output CVP_p instance as follows. Let $d := md^* + n$. The basis $B \in \mathbb{R}^{d \times n}$ and target vector $\boldsymbol{t} \in \mathbb{R}^d$ in the output instance have the form

$$B = \begin{pmatrix} B_1 \\ \vdots \\ B_m \\ 2\alpha^{1/p} \cdot I_n \end{pmatrix}, \qquad \boldsymbol{t} = \begin{pmatrix} \boldsymbol{t}_1 \\ \vdots \\ \boldsymbol{t}_m \\ \alpha^{1/p} \cdot \boldsymbol{1}_n \end{pmatrix},$$

with blocks $B_i \in \mathbb{R}^{d^* \times n}$ and $\boldsymbol{t}_i \in \mathbb{R}^{d^*}$ for $1 \le i \le m$ and $\alpha := m + (m-W)\delta$. Note that α is the maximum possible contribution of the clauses $C_1, \ldots, C_m$ to $\|B\boldsymbol{y} - \boldsymbol{t}\|_p^p$ when (Φ, W) is a 'YES' instance. For every $1 \le i \le m$ and $1 \le j \le n$, set the jth column $(B_i)_j$ of block B_i (corresponding to the clause $C_i = \vee_{s=1}^k \ell_{i,s}$) as

$$(B_i)_j := \begin{cases} \boldsymbol{v}_s & \text{if } x_j \text{ is the } s\text{th literal of clause } i, \\ -\boldsymbol{v}_s & \text{if } \neg x_j \text{ is the } s\text{th literal of clause } i, \\ \boldsymbol{0}_{d^*} & \text{otherwise,} \end{cases}$$

and set $\boldsymbol{t}_i := \boldsymbol{t}^* - \sum_{s \in N_i} \boldsymbol{v}_s$. Set $r := (\alpha(n+1))^{1/p}$.

Clearly, the reduction runs in polynomial time. We next analyze for which $\boldsymbol{y} \in \mathbb{Z}^n$ it holds that $\|B\boldsymbol{y} - \boldsymbol{t}\|_p \le r$. Given $\boldsymbol{y} \notin \{0,1\}^n$,

$$\|B\boldsymbol{y} - \boldsymbol{t}\|_p^p \ge \|2\alpha^{1/p} I_n \boldsymbol{y} - \alpha^{1/p}\boldsymbol{1}_n\|_p^p \ge \alpha(n+2) > r^p,$$

so we only need to analyze the case when $\boldsymbol{y} \in \{0,1\}^n$. Consider an assignment $\boldsymbol{y} \in \{0,1\}^n$ to the variables of Φ. Then,

$$\begin{aligned} &\|B_i\boldsymbol{y} - \boldsymbol{t}_i\|_p \\ &= \Big\| \sum_{s \in P_i} y_{\mathrm{ind}(\ell_{i,s})} \cdot \boldsymbol{v}_s + \sum_{s \in N_i} \big(1 - y_{\mathrm{ind}(\ell_{i,s})}\big) \cdot \boldsymbol{v}_s - \boldsymbol{t}^* \Big\|_p \\ &= \Big\| \sum_{s \in S_i(\boldsymbol{a})} \boldsymbol{v}_s - \boldsymbol{t}^* \Big\|_p. \end{aligned}$$

By assumption, the last quantity is equal to 1 if $|S_i(\boldsymbol{y})| \ge 1$, and is equal to $(1+\delta)^{1/p}$ otherwise. Because $|S_i(\boldsymbol{y})| \ge 1$ if and only if C_i is satisfied, it follows that

$$\begin{aligned} \|B\boldsymbol{y} - \boldsymbol{t}\|_p^p &= \Big(\sum_{i=1}^m \|B_i\boldsymbol{y} - \boldsymbol{t}_i\|_p^p \Big) + \alpha n \\ &= m + (m - m^+(\boldsymbol{y}))\delta + \alpha n. \end{aligned}$$

Therefore, $\|B\boldsymbol{y} - \boldsymbol{t}\|_p \le r$ if and only if $m^+(\boldsymbol{y}) \ge W$, and therefore there exists $\boldsymbol{y}$ such that $\|B\boldsymbol{y} - \boldsymbol{t}\|_p \le r$ if and only if (Φ, W) is a 'YES' instance of Max-k-SAT, as needed.

To extend this to a reduction from *weighted* Max-k-SAT to CVP_p, simply multiply each block B_i and the corresponding target vector $\boldsymbol{t}_i$ by $w(C_i)^{1/p}$, where $w(C_i)$ denotes the weight of the clause C_i. Then, by adjusting α to depend on the weights $w(C_i)$ we obtain the desire reduction. ∎

Because the rank n of the output CVP_p instance matches the number of variables in the input SAT formula, we immediately get the following corollary.

Corollary II.3. *For any efficiently computable $p = p(n) \in [1, \infty)$ if there exists a computable (p,k)-isolating parallelepiped for infinitely many $k \in \mathbb{Z}^+$, then, for every constant $\varepsilon > 0$ there is no $2^{(1-\varepsilon)n}$-time algorithm for CVP_p assuming W-Max-SAT-SETH. In particular there is no $2^{(1-\varepsilon)n}$-time algorithm for CVP_p assuming SETH.*

It is easy to construct a (degenerate) family of isolating parallelepipeds for $p = 1$, and therefore we get hardness of CVP_1 as a simple corollary. (See Figure 2.)

Corollary II.4. *For every constant $\varepsilon > 0$ there is no $2^{(1-\varepsilon)n}$-time algorithm for CVP_1 assuming W-Max-SAT-SETH, and in particular there is no $2^{(1-\varepsilon)n}$-time algorithm for CVP_1 assuming SETH.*

Proof: Let $k \in \mathbb{Z}^+$, let $V = [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_k]$ with $\boldsymbol{v}_1 = \cdots = \boldsymbol{v}_k := \frac{1}{k-1}(1,1)^T \in \mathbb{R}^2$, and let $\boldsymbol{t}^* := \frac{1}{k-1}(1,k)^T \in \mathbb{R}^2$. Then, $\|V\boldsymbol{x} - \boldsymbol{t}^*\|_1 = 1$ for every $\boldsymbol{x} \in \{0,1\}^k \setminus \{\boldsymbol{0}_k\}$, and $\|\boldsymbol{t}^*\|_1 = (k+1)/(k-1) > 1$. The result follows by Corollary II.3. ∎

References

[1] A. K. Lenstra, H. W. Lenstra, Jr., and L. Lovász, "Factoring polynomials with rational coefficients," *Math. Ann.*, vol. 261, no. 4, pp. 515–534, 1982. [Online]. Available: http://dx.doi.org/10.1007/BF01457454

[2] H. W. Lenstra, Jr., "Integer Programming with a fixed number of variables," *Math. Oper. Res.*, vol. 8, no. 4, pp. 538–548, 1983. [Online]. Available: http://dx.doi.org/10.1287/moor.8.4.538

[3] R. Kannan, "Minkowski's convex body theorem and Integer Programming," *Math. Oper. Res.*, vol. 12, no. 3, pp. 415–440, 1987. [Online]. Available: http://dx.doi.org/10.1287/moor.12.3.415

[4] D. Dadush, C. Peikert, and S. Vempala, "Enumerative lattice algorithms in any norm via M-ellipsoid coverings," in *FOCS*, 2011.

[5] A. M. Odlyzko, "The rise and fall of knapsack cryptosystems," *Cryptology and Computational Number Theory*, vol. 42, pp. 75–88, 1990.

[6] A. Joux and J. Stern, “Lattice reduction: A toolbox for the cryptanalyst,” *Journal of Cryptology*, vol. 11, no. 3, pp. 161–185, 1998.

[7] P. Q. Nguyen and J. Stern, “The two faces of lattices in cryptology,” in *Cryptography and Lattices*. Springer, 2001, pp. 146–180.

[8] M. Ajtai, “Generating hard instances of lattice problems,” in *Complexity of computations and proofs*, ser. Quad. Mat. Dept. Math., Seconda Univ. Napoli, Caserta, 2004, vol. 13, pp. 1–32, preliminary version in STOC’96.

[9] O. Regev, “On lattices, learning with errors, random linear codes, and cryptography,” *Journal of the ACM*, vol. 56, no. 6, pp. Art. 34, 40, 2009. [Online]. Available: http://dx.doi.org/10.1145/1568318.1568324

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

[11] C. Peikert, “Limits on the hardness of lattice problems in ℓ_p norms,” *Computational Complexity*, vol. 17, no. 2, pp. 300–351, May 2008, preliminary version in CCC 2007.

[12] ——, “A decade of lattice cryptography,” *Foundations and Trends in Theoretical Computer Science*, vol. 10, no. 4, pp. 283–424, 2016.

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

[14] N. Gama and P. Q. Nguyen, “Finding short lattice vectors within Mordell’s inequality,” in *STOC*, 2008.

[15] D. Micciancio and M. Walter, “Practical, predictable lattice basis reduction,” in *Eurocrypt*, 2016.

[16] M. Ajtai, R. Kumar, and D. Sivakumar, “A sieve algorithm for the Shortest Lattice Vector Problem,” in *STOC*, 2001. [Online]. Available: http://doi.acm.org/10.1145/380752.380857

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

[18] X. Pujol and D. Stehlé, “Solving the Shortest Lattice Vector Problem in time $2^{2.465n}$,” *IACR Cryptology ePrint Archive*, vol. 2009, p. 605, 2009.

[19] D. Micciancio and P. Voulgaris, “Faster exponential time algorithms for the Shortest Vector Problem,” in *SODA*, 2010, pp. 1468–1480.

[20] M. Liu, X. Wang, G. Xu, and X. Zheng, “Shortest lattice vectors in the presence of gaps,” *IACR Cryptology ePrint Archive*, vol. 2011, p. 139, 2011.

[21] D. Aggarwal, D. Dadush, O. Regev, and N. Stephens-Davidowitz, “Solving the Shortest Vector Problem in 2^n time via discrete Gaussian sampling,” in *STOC*, 2015.

[22] X. Wang, M. Liu, C. Tian, and J. Bi, “Improved Nguyen-Vidick heuristic sieve algorithm for shortest vector problem,” in *ASIACCS*, 2011.

[23] T. Laarhoven, “Sieving for shortest vectors in lattices using angular locality-sensitive hashing,” in *CRYPTO*, 2015.

[24] A. Becker, L. Ducas, N. Gama, and T. Laarhoven, “New directions in nearest neighbor searching with applications to lattice sieving,” in *SODA*, 2016.

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

[26] V. Arvind and P. S. Joglekar, “Some sieving algorithms for lattice problems,” in *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science*, 2008.

[27] D. Dadush, “A $O(1/\varepsilon^2)^n$-time sieving algorithm for approximate Integer Programming,” in *LATIN*, 2012.

[28] M. Ajtai, R. Kumar, and D. Sivakumar, “Sampling short lattice vectors and the Closest Lattice Vector Problem,” in *CCC*, 2002.

[29] D. Aggarwal, D. Dadush, and N. Stephens-Davidowitz, “Solving the Closest Vector Problem in 2^n time— The discrete Gaussian strikes again!” in *FOCS*, 2015.

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

[31] D. Micciancio and P. Voulgaris, “A deterministic single exponential time algorithm for most lattice problems based on Voronoi cell computations,” *SIAM Journal on Computing*, vol. 42, no. 3, pp. 1364–1391, 2013.

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

[33] N. Bonifas and D. Dadush, "Short paths on the Voronoi graph and the Closest Vector Problem with Preprocessing," in *SODA*, 2015.

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

[35] P. van Emde Boas, "Another NP-complete problem and the complexity of computing short vectors in a lattice," University of Amsterdam, Department of Mathematics, Netherlands, Tech. Rep., 1981, technical Report 8104.

[36] M. Ajtai, "The Shortest Vector Problem in L2 is NP-hard for randomized reductions," in *STOC*, 1998. [Online]. Available: http://doi.acm.org/10.1145/276698.276705

[37] S. Arora, L. Babai, J. Stern, and Z. Sweedyk, "The hardness of approximate optima in lattices, codes, and systems of linear equations," in *FOCS*, 1993.

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

[39] J.-Y. Cai and A. Nerurkar, "Approximating the SVP to within a factor $(1 + 1/\dim^{\varepsilon})$ is NP-hard under randomized conditions," in *CCC*, 1998.

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

[41] S. Khot, "Hardness of approximating the Shortest Vector Problem in lattices," *Journal of the ACM*, vol. 52, no. 5, pp. 789–808, Sep. 2005, preliminary version in FOCS'04.

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

[43] O. Goldreich and S. Goldwasser, "On the limits of nonapproximability of lattice problems," *J. Comput. Syst. Sci.*, vol. 60, no. 3, pp. 540–563, 2000, preliminary version in STOC 1998.

[44] D. Aharonov and O. Regev, "Lattice problems in NP $\cap$ coNP," *J. ACM*, vol. 52, no. 5, pp. 749–765, 2005, preliminary version in FOCS 2004.

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

[46] E. Alkim, L. Ducas, T. Pöppelmann, and P. Schwabe, "Post-quantum key exchange — A new hope," in *USENIX Security Symposium*, 2016.

[47] J. W. Bos, C. Costello, L. Ducas, I. Mironov, M. Naehrig, V. Nikolaenko, A. Raghunathan, and D. Stebila, "Frodo: Take off the ring! Practical, quantum-secure key exchange from LWE," in *CCS*, 2016.

[48] "NIST post-quantum standardization call for proposals," http://csrc.nist.gov/groups/ST/post-quantum-crypto/cfp-announce-dec2016.html, 2016, accessed: 2017-04-02.

[49] M. R. Albrecht, R. Player, and S. Scott, "On the concrete hardness of Learning with Errors," *J. Mathematical Cryptology*, vol. 9, no. 3, pp. 169–203, 2015.

[50] R. Impagliazzo and R. Paturi, "On the complexity of k-SAT," in *CCC*, 1999, pp. 237–240.

[51] S. Chechik, D. H. Larkin, L. Roditty, G. Schoenebeck, R. E. Tarjan, and V. V. Williams, "Better approximation algorithms for the graph diameter," in *SODA*, 2014.

[52] A. Backurs and P. Indyk, "Edit Distance cannot be computed in strongly subquadratic time (unless SETH is false)," in *STOC*, 2015.

[53] A. Abboud, A. Backurs, and V. V. Williams, "Tight hardness results for LCS and other sequence similarity measures," in *FOCS*, 2015.

[54] V. V. Williams, "Hardness of easy problems: Basing hardness on popular conjectures such as the Strong Exponential Time Hypothesis (invited talk)," in *IPEC*, 2015, pp. 17–29.

[55] M. Pătraşcu and R. Williams, "On the possibility of faster SAT algorithms," in *SODA*, 2010.

[56] M. Cygan, H. Dell, D. Lokshtanov, D. Marx, J. Nederlof, Y. Okamoto, R. Paturi, S. Saurabh, and M. Wahlström, "On problems as hard as CNF-SAT," in *CCC*, 2012.

[57] M. Cygan, F. V. Fomin, L. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh, *Parameterized Algorithms*. Springer, 2015. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-21275-3

[58] R. Williams, "Strong ETH breaks with Merlin and Arthur: Short non-interactive proofs of batch evaluation," in *CCC*, 2016.

[59] O. Regev and R. Rosen, "Lattice problems and norm embeddings," in *STOC*, 2006.

[60] H. Bennett, A. Golovnev, and N. Stephens-Davidowitz, "On the quantitative hardness of CVP," 2017, http://arxiv.org/abs/1704.03928.

[61] D. Micciancio, "The hardness of the Closest Vector Problem with Preprocessing," *IEEE Transactions on Information Theory*, vol. 47, no. 3, pp. 1212–1215, 2001.

[62] U. Feige and D. Micciancio, "The inapproximability of lattice and coding problems with preprocessing," *Journal of Computer and System Sciences*, vol. 69, no. 1, pp. 45–67, 2004, preliminary version in CCC 2002.

[63] O. Regev, "Improved inapproximability of lattice and coding problems with preprocessing," *IEEE Transactions on Information Theory*, vol. 50, no. 9, pp. 2031–2037, 2004, preliminary version in CCC'03.

[64] M. Alekhnovich, S. Khot, G. Kindler, and N. K. Vishnoi, "Hardness of approximating the Closest Vector Problem with Pre-processing." *Computational Complexity*, vol. 20, 2011.

[65] S. Khot, P. Popat, and N. K. Vishnoi, "$2^{\log^{1-\varepsilon} n}$ hardness for Closest Vector Problem with Preprocessing," *SIAM Journal on Computing*, vol. 43, no. 3, pp. 1184–1205, 2014.

[66] D. Dadush, O. Regev, and N. Stephens-Davidowitz, "On the Closest Vector Problem with a distance guarantee," in *CCC*, 2014, pp. 98–109. [Online]. Available: http://dx.doi.org/10.1109/CCC.2014.18

[67] R. Williams, "A new algorithm for optimal 2-Constraint Satisfaction and its implications," *Theoretical Computer Science*, vol. 348, no. 2-3, pp. 357–365, 2005.

[68] V. V. Williams, "Multiplying matrices faster than Coppersmith-Winograd," in *STOC*, 2012.

[69] F. Le Gall, "Powers of tensors and fast matrix multiplication," in *ISAAC*, 2014.

[70] G. J. Woeginger, "Open problems around exact algorithms," *Discrete Applied Mathematics*, vol. 156, no. 3, pp. 397–405, 2008.