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 \( \ell_p \) norm (CVP\(_p\)) over rank \( n \) lattices cannot be solved in \( 2^{(1+\epsilon)n} \) time for any constant \( \epsilon > 0 \) 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\(_\infty\); 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 \( \mathbf{b}_1, \ldots, \mathbf{b}_n \in \mathbb{R}^d \),

\[
\mathcal{L} = \mathcal{L}(\mathbf{b}_1, \ldots, \mathbf{b}_n) := \left\{ \sum_{i=1}^{n} z_i \mathbf{b}_i : z_i \in \mathbb{Z} \right\}.
\]

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 \( \mathbf{t} \in \mathbb{R}^d \) to the lattice. Typically, we define shortness and closeness in terms of the \( \ell_p \) norm for some \( 1 \leq p \leq \infty \), given by

\[
\| \mathbf{x} \|_p := (|x_1|^p + |x_2|^p + \cdots + |x_d|^p)^{1/p}
\]

for finite \( p \) and

\[
\| \mathbf{x} \|_\infty := \max_{1 \leq i \leq d} |x_i|.
\]

In particular, the \( \ell_2 \) 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 \( \ell_p \) 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.
[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 \(\ell_p\) 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 \(\ell_p\) 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 \(n^{\tilde{O}(n)}\)-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 \(\ell_2\) norm, and it seems that substantial new ideas would be required to extend them to arbitrary \(\ell_p\) norms.

**B. Hardness of SVP and CVP**

Van Emde Boas showed the NP-hardness of CVP\(_p\) for any \(p\) and SVP\(_\infty\) 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 \(1 \leq p \leq \infty\) 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 NP \(\cap\) 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 \(2^{10^{-1}n}\) under plausible complexity-theoretic assumptions [45].
is asymptotically better than the current best of $2^{n+o(n)}$ \cite{31, 29}. Second, some lattice-based cryptographic constructions are close to deployment \cite{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 \cite{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 \cite{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., \cite{51, 52, 53} and the survey by Vassilevska Williams \cite{54}), and NP-hard problems (see, e.g., \cite{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 \cite{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 \cite{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 \cite{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 $p = \infty$).

D. Our contribution

We now enumerate our results. See also Table 1-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$_\infty$). Formally, we prove the following.

**Theorem 1.1.** 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$_\infty$) 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$_\infty$) 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 necessary fails for even integers $p \leq k - 1$. In spite of this, we actually prove the following result that generalizes Theorem 1.1 to “almost all” $p \geq 1$ (including non-integer $p$).

**Theorem 1.2.** 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 $p = \infty$;

<table>
<thead>
<tr>
<th>Problem</th>
<th>Upper Bound</th>
<th>SETH</th>
<th>Max-2-SAT</th>
<th>ETH</th>
<th>Gap-ETH</th>
</tr>
</thead>
<tbody>
<tr>
<td>CVP$_p$</td>
<td>$n^{\Omega(n)}(2^{O(n)})$</td>
<td>$2^n$</td>
<td>$2^{2n/3}$</td>
<td>$2^{O(n)}$</td>
<td>$2^{\Omega(n^*)}$</td>
</tr>
<tr>
<td>CVP$_{2n}$</td>
<td>$2^{O(n)}$</td>
<td>$2^{\sqrt{n}}$</td>
<td>$2^{O(n)}$</td>
<td>$2^{\Omega(n^*)}$</td>
<td></td>
</tr>
<tr>
<td>CVPP$_p$</td>
<td>$n^{\Omega(n)}(2^{O(n)})$</td>
<td>$2^{\sqrt{n}}$</td>
<td>$2^{O(n)}$</td>
<td>$2^{\Omega(n^*)}$</td>
<td></td>
</tr>
</tbody>
</table>

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 1.2. 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. $\omega$ is the matrix multiplication exponent, satisfying $2 \leq \omega < 2.373$. We have suppressed smaller factors.
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 \( \text{CVP}_p \) 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 \( \varepsilon \)); or
3) \( p = p_0 + \delta(n) \).

Notice that this lower bound (Theorem I.2) comes tantalizingly close to resolving the quantitative complexity of \( \text{CVP}_2 \). In particular, we obtain a \( 2^n \)-time lower bound on \( \text{CVP}_{2+\delta} \) for any \( \delta(n) = o(1) \), and the fastest algorithm for \( \text{CVP}_2 \) run in time \( 2^{n+o(n)} \). But, formally, Theorems I.1 and I.2 say nothing about \( \text{CVP}_2 \). (Indeed, there is at least some reason to believe that \( \text{CVP}_2 \) is easier than \( \text{CVP}_p \) for \( p \neq 2 \).)

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

Finally, we note that in the special case of \( p = \infty \), our reduction works even for approximate \( \text{CVP}_\infty \), or even approximate \( \text{SVP}_\infty \), with an approximation factor of \( \gamma := 1 + 2/(k-1) \). In particular, \( \gamma \) 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 \( \text{SVP}_\infty \) or \( \text{CVP}_\infty \) to within a factor of \( \gamma_{\varepsilon} \) unless SETH fails.

Quantitative hardness of approximate \( \text{CVP} \): As we discussed above, many \( 2^{O(n)} \)-time algorithms for \( \text{CVP}_p \) only work for \( \gamma \)-approximate \( \text{CVP}_p \) for constant approximation factors \( \gamma > 1 \). However, the reduction described above only works for exact \( \text{CVP}_p \) (except when \( p = \infty \)), or at best for \( \gamma \)-approximate \( \text{CVP}_p \) 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 \( \text{Max}-k\text{-SAT} \) 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 \( \text{CVP}_p \) to within some constant factor unless a \( 2^{o(n)} \)-time algorithm exists for approximating \( \text{Max}-2\text{-SAT} \). We also note that a \( 2^{o(n)} \)-time algorithm for approximating \( \text{Max}-2\text{-SAT} \) to within a constant factor would imply one for \( \text{Max}-3\text{-SAT} \) as well.\(^5\)

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

**Theorem I.3.** There is an efficient reduction from approximating a \( \text{Max}-2\text{-SAT} \) on \( n \) variables and \( m \) clauses to within a constant factor to approximating \( \text{CVP}_p \) 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 \( \text{CVP} \) with Preprocessing: \( \text{CVP} \) with Preprocessing (CVPP) is the \( \text{CVPP} \) variant of \( \text{CVP} \) 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 \( \text{CVPP}_2 \) is used as a subroutine in the celebrated Micciancio-Voulgaris algorithm for \( \text{CVP}_2 \) [31], [33]. The complexity of \( \text{CVPP}_p \) 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 \( \text{CVPP}_p \).

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

Additional quantitative hardness results for \( \text{CVPP}_p \): We also observe the following weaker hardness result for \( \text{CVPP}_p \) for any \( 1 \leq p < \infty \) based on different assumptions. The ETH-hardness of \( \text{CVPP}_p \) 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 \( \text{Max}-2\text{-SAT} \) on \( n \) variables to \( \text{CVP}_p \) on a lattice of rank \( n \). (Prior to this work, we were only aware of reductions from \( \text{3-SAT} \) on \( n \) variables to \( \text{CVP}_p \) 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 \( \text{Max}-2\text{-SAT} \) with \( n \) variables
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 $(\text{poly}(n)) \cdot 2^{cn}$-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 poly$(n)$, $2^{2n/3}$-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 $2^{2n/3}$-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 $\Phi$ with $n$ variables and $m$ clauses, we construct the lattice basis

$$B := \left( \begin{array}{c} \Phi \\ 2\alpha I_n \end{array} \right), \tag{1}$$

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

$$\Phi_{i,j} := \left\{ \begin{array}{cl} 2 & \text{if } x_j \text{ is in the } i\text{th clause}, \\ -2 & \text{if } \neg x_j \text{ is in the } i\text{th clause}, \\ 0 & \text{otherwise}. \end{array} \right. \tag{2}$$

I.e., the rows of $\Phi$ 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 $t \in \mathbb{R}^{m+n}$ is given by

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

where

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

where $\eta_i$ is the number of negated variables in the $i$th clause.

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

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

It follows that the distance to the target is exactly $\text{dist}_{p}(t, L)^p = \alpha^p n + 3p(m - S) = \alpha^p n - (3p - 1)S + 3p m$, where $S$ is the maximal number of satisfied clauses on the input. So, the distance $\text{dist}_{p}(t, L)$ 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 $i$th coordinate of $Bz - t$ is $4 - 3$ if two literals in the $i$th clause are satisfied by $z \in \{0,1\}^n$, $2 - 3$ if one literal is satisfied, and $0 - 3$ if the clause is unsatisfied. Since $|4 - 3| = |2 - 3|$, the “contribution” of this $i$th coordinate to the distance $\|Bz - t\|_p^p$ is the same for any satisfied clause. Since $|0 - 3| > |4 - 3|$, the contribution to the $i$th 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 $\Phi \in \mathbb{R}^{m \times n}$ to encode the literals in each clause as in Eq. (2) and construct our lattice basis $B$ as in Eq. (1) and target $t$ as in Eq. (3), perhaps with the number 3 in the definition of $t$ replaced by an arbitrary $t^* \in \mathbb{R}$. Then, the $i$th coordinate of $Bz - t$ would be $2S_i - t^*$, where $S_i$ is the number of literals satisfied in the $i$th 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 $t^* \in \mathbb{R}^d^*$.

We therefore consider modifying the reduction from above by replacing the scalar $\pm 1$ values in our matrix $\Phi$ with vectors in $\mathbb{R}^{d^*}$ for some $d^*$. In particular, for some vectors $V = (v_1, \ldots, v_k) \in \mathbb{R}^{d^*}^k$, we define $\tilde{\Phi} \in \mathbb{R}^{d^*m \times n}$ as

$$\tilde{\Phi}_{i,j} := \begin{cases} v_s & x_j \text{ is } s\text{th literal in } i\text{th clause}, \\ -v_s & x_j \text{ is } s\text{th literal in } i\text{th clause}, \\ 0_d & \text{otherwise}, \end{cases}$$

(5)

where we have abused notation and taken $\tilde{\Phi}_{i,j}$ to be a column vector in $d^*$ dimensions. By defining $t \in \mathbb{R}^{d^*m+n}$ appropriately, we will get that the “contribution of the $i$th clause to the distance” $\|Bz - t\|_p$ is exactly $\|Vy - t^*\|_p$ for some $t^* \in \mathbb{R}^{d^*}$, where $y \in \{0,1\}^k$ such that $y_s = 1$ if and only if $z$ satisfies the $s$th 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 $t^* \in \mathbb{R}^{d^*}$ with the special property that one vertex of the parallelepiped $V\{0,1\}^k - t^*$ is “isolated.” (Here, $V\{0,1\}^k - t^*$ is an affine transformation of the hypercube, i.e., a parallelepiped.)

In particular, every vertex of the parallelepiped, $V y - t^*$ for $y \in \{0,1\}^k$ has unit length $\|Vy - t^*\|_p = 1$ except for the vertex $-t^*$, which is longer, i.e., $\|t^*\|_p > 1$. (See Figure 2.)

In terms of the reduction above, an isolating parallelepiped is exactly what we need. In particular, if we plug $V$ and $t^*$ into the above reduction, then all satisfied clauses (which correspond to non-zero $y$ in the above description) will “contribute” 1 to the distance $\|Bz - t\|_p$, while unsatisfied clauses (which correspond to $y = 0$) will contribute $1 + \delta$ for some $\delta > 0$. Therefore, the total distance will be exactly $\|Bz - t\|_p = \alpha^p n + m^p(z) + (m - m^p(z))(1 + \delta) = \alpha^p n - \delta m(z) + (1 + \delta)m$, where $m(z)$ is the number of clauses satisfied by $z$. So, the distance $\text{dist}_p(t, L)$ 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 $V \in \mathbb{R}^{2^k \times k}, t^* \in \mathbb{R}^{2^k}$ 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 $\|Vy - t^*\|_p$ for $y \in \{0,1\}^k$ depends only on the Hamming weight of $y$ and is linear in the $\alpha_i$ 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 $\|Vy - t^*\|_p$ for each possible Hamming weight of $y \in \{0,1\}^k$. We show that, in order to find weights $\alpha_0, \ldots, \alpha_k \geq 0$ such that $V$ and $t^*$ define a $(p,k)$-isolating parallelepiped, it suffices to find a $t^*$ such that $M_k(p, t^*)$ is invertible. For each odd
In particular, while our result applies to $t = 2$, this follows from above with a constant approximation (except when $p = 1$). We consider $\ell = 2$. A non-explicit result is the extension of our SETH-hardness result to arbitrary $p \geq 1$ 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 1.2.

**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 $2^{\Omega(n)}$ (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\(_\infty\), 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 \(\ell_2\) 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 \(1_n\) and \(0_n\) 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 \(V \in \mathbb{R}^{d^\ast \times k}\) and \(t^\ast \in \mathbb{R}^d\) define a \((p,k)\)-isolation parallelepiped if \(\|t\|_p > 1\) and \(\|Vx - t^\ast\|_p = 1\) for all \(x \in \{0,1\}^k \setminus \{0_k\}\).

In order to give the reduction, we first introduce some notation related to SAT. Let \(\Phi\) be a \(k\)-SAT formula on \(n\) variables \(x_1, \ldots, x_n\) and \(m\) clauses \(C_1, \ldots, C_m\). Let \(\text{ind}(\ell)\) denote the index of the variable underlying a literal \(\ell\). I.e., \(\text{ind}(\ell) = j\) if \(\ell = x_j\) or \(\ell = \neg x_j\). Call a literal \(\ell\) positive if \(\ell = x_j\) and negative if \(\ell = \neg x_j\) for some variable \(x_j\). Given a clause

\[
\begin{align*}
C_1 & \{ \quad v_1 & v_2 & \cdots & 0_d & -v_3 & t^\ast - v_3 \\
& \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\
C_m & 0_d & -v_3 & \cdots & v_2 & v_3 & t^\ast - v_1 \\
x_1 & 2a^{1/p} & 0 & \cdots & 0 & 0 & a^{1/p} \\
x_2 & 0 & 2a^{1/p} & \cdots & 0 & 0 & a^{1/p} \\
& \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\
x_{n-1} & 0 & 0 & \cdots & 2a^{1/p} & 0 & a^{1/p} \\
x_n & 0 & 0 & \cdots & 0 & 2a^{1/p} & a^{1/p}
\end{align*}
\]

Figure 4. A basis \(B\) and target vector \(t\) output by the reduction from Theorem II.2 with some \((p,3)\)-isolating parallelepiped given by \(V = (v_1, v_2, v_3) \in \mathbb{R}^{d^\ast \times 3}\) and \(t^\ast \in \mathbb{R}^d\). In this example, the first clause is \(C_1 \equiv x_1 \vee x_2 \vee \neg x_3\) and the \(m\)th 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 \(\|Bz - t\|_p^p\) will be 1 for any assignment \(z \in \{0,1\}^n\) satisfying \(C_1\), while non-satisfying assignments contribute \((1 + \delta)\) for some \(\delta > 0\). For example, if \(z_1 = 1, z_2 = 0, z_3 = 1\), the clause \(C_1\) is satisfied, and the first \(d\) coordinates will contribute \(\|v_1 - v_3 - (t^\ast - v_3)\|_p^p = \|v_1 - t^\ast\|_p^p = 1\). On the other hand, if \(z_1 = 0, z_2 = 0, z_3 = 1\), then \(C_1\) is not satisfied, and \(\|v_3 - (t^\ast - v_3)\|_p^p = \|t^\ast\|_p^p = 1 + \delta\).

\[C_i = \bigvee_{s=1}^k \ell_{i,s}\text{, let } P_i := \{ s \in [k] : \ell_{i,s} \text{ is positive} \}\text{ and let } N_i := \{ s \in [k] : \ell_{i,s} \text{ is negative} \}\text{ denote the indices of positive and negative literals in } C_i \text{ respectively. Given an assignment } a \in \{0,1\}^n \text{ to the variables of } \Phi, \text{ let } S_i(a) \text{ denote the indices of literals in } C_i \text{ satisfied by } a. \text{ I.e., } S_i(a) := \{ s \in P_i : a_{\text{ind}(\ell_{i,s})} = 1 \} \cup \{ s \in N_i : a_{\text{ind}(\ell_{i,s})} = 0 \}. \text{ Finally, let } m^+(a) \text{ denote the number of clauses of } \Phi \text{ satisfied by the assignment } a, \text{ i.e., the number of clauses } i \text{ for which } |S_i(a)| \geq 1.\]

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 \((\Phi, W)\) to an instance \((B, t^\ast, r)\) of CVP\(_p\). Here, the formula \(\Phi\) is on \(n\) variables \(x_1, \ldots, x_n\) and \(m\) clauses \(C_1, \ldots, C_m\). \((\Phi, W)\) is a ‘YES’ instance if there exists an assignment \(a\) such that \(m^+(a) \geq W\).
By assumption, there exist computable \( d^* = d(p, k) \in \mathbb{Z}^+, V = [v_1, \ldots, v_k] \in \mathbb{R}^{d \times k} \), and \( t^* \in \mathbb{R}^d \) such that \( ||t^*||_p = (1 + \delta)^{1/p} \) for some \( \delta > 0 \) and \( ||Vz - t^*||_p = 1 \) for all \( z \in \{0, 1\}^k \setminus \{0\}_k \).

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

\[
B = \begin{pmatrix} B_1 & \cdots & B_m \\ \cdot & \cdots & \cdot \\ 2\alpha^{1/p} \cdot I_n & \cdot & \cdot \end{pmatrix}, \quad t = \begin{pmatrix} t_1 \\ \vdots \\ t_m \\ \alpha^{1/p} \cdot 1_n \end{pmatrix},
\]

with blocks \( B_i \in \mathbb{R}^{d \times n} \) and \( t_i \in \mathbb{R}^d \) for \( 1 \leq i \leq m \) and \( \alpha := m + (m-W)\delta \). Note that \( \alpha \) is the maximum possible contribution of the clauses \( C_1, \ldots, C_m \) to \( ||B_y - t||_p \) when \( (\Phi, W) \) is a ‘YES’ instance. For every \( 1 \leq i \leq m \) and \( 1 \leq j \leq n \), set the \( j \)th column \( (B_i)_j \) of block \( B_i \) (corresponding to the clause \( C_i = \vee_{s=1}^k \ell_{i,s} \)) as

\[
(B_i)_j := \begin{cases} v_s & \text{if } x_j \text{ is the } j \text{th literal of clause } i, \\ -v_s & \text{if } \neg x_j \text{ is the } j \text{th literal of clause } i, \\ 0 & \text{otherwise}, \end{cases}
\]

and set \( t_i := t^* - \sum_{s \in N_i} v_s \). Set \( r := (\alpha(n+1))^{1/p} \).

Clearly, the reduction runs in polynomial time. We next analyze for which \( \|B_y - t\|_p \leq r \). Given \( y \notin \{0, 1\}^n \),

\[
\|B_y - t\|_p^p = \sum_{s \in P_i} y_{\text{ind}(i,s)} \cdot v_s + \sum_{s \in N_i} (1 - y_{\text{ind}(i,s)}) \cdot v_s - t^* \|_p^p \\
= \sum_{s \in S_i(a)} v_s - t^* \|_p^p.
\]

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

\[
\|B_y - t\|_p = \left( \sum_{i=1}^m \|B_i y - t_i\|_p^p \right) + \alpha n \\
= m + (m - m^+(y))\delta + \alpha n.
\]

Therefore, \( \|B_y - t\|_p \leq r \) if and only if \( m^+(y) \geq W \), and therefore there exists \( y \) such that \( \|B_y - t\|_p \leq r \) if and only if \( (\Phi, W) \) is a ‘YES’ instance of Max-k-SAT, as needed.

To extend this to a reduction from weighted Max-k-SAT to CVP, simply multiply each block \( B_i \) and the corresponding target vector \( t_i \) by \( w(C_i)^{1/p} \), where \( w(C_i) \) denotes the weight of the clause \( C_i \). Then, by adjusting \( \alpha \) to depend on the weights \( w(C_i) \) we obtain the desired reduction.

Because the rank \( n \) of the output CVP 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 assuming W-Max-SAT-SETH. In particular there is no \( 2^{(1-\varepsilon)n} \)-time algorithm for CVP assuming SETH.

It is easy to construct a (degenerate) family of isolating parallelepipeds for \( p = 1 \), and therefore we get hardness of CVP 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 assuming W-Max-SAT-SETH, and in particular there is no \( 2^{(1-\varepsilon)n} \)-time algorithm for CVP assuming SETH.

**Proof:** Let \( k \in \mathbb{Z}^+ \), let \( V = [v_1, \ldots, v_k] \) with \( v_1 = \cdots = v_k := \frac{1}{k-1} (1, 1)^T \in \mathbb{R}^2 \), and let \( t^* := \frac{1}{\varepsilon} (1, k)^T \in \mathbb{R}^2 \). Then, \( ||Vx - t^*||_1 = 1 \) for every \( x \in \{0, 1\}^k \setminus \{0\}_k \), and \( ||t^*||_1 = (k + 1)/(k - 1) > 1 \). The result follows by Corollary II.3.

**References**


[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.


