Search-to-Decision Reductions for Lattice Problems with Approximation Factors (Slightly) Greater Than One*

Noah Stephens-Davidowitz

Courant Institute of Mathematical Sciences, New York University, NY, USA
noahsd@gmail.com

Abstract

We show the first dimension-preserving search-to-decision reductions for approximate SVP and CVP. In particular, for any $\gamma \leq 1 + O(\log n/n)$, we obtain an efficient dimension-preserving reduction from $\gamma^{O(n/\log n)}$-SVP to γ-GapSVP and an efficient dimension-preserving reduction from $\gamma^{O(n)}$-CVP to γ-GapCVP. These results generalize the known equivalences of the search and decision versions of these problems in the exact case when $\gamma = 1$. For SVP, we actually obtain something slightly stronger than a search-to-decision reduction – we reduce $\gamma^{O(n/\log n)}$-SVP to γ-unique SVP, a potentially easier problem than γ-GapSVP.

1998 ACM Subject Classification F.2.2 [Nonnumerical Algorithms and Problems] Geometrical Problems and Computations

Keywords and phrases Lattices, SVP, CVP

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2016.19

1 Introduction

A lattice $\mathcal{L} = \{\sum a_i \mathbf{b}_i : a_i \in \mathbb{Z}\} \subset \mathbb{R}^n$ is the set of all integer linear combinations of linearly independent basis vectors $\mathbf{b}_1, \ldots, \mathbf{b}_n \in \mathbb{R}^n$.

The two most important computational problems on lattices are the Shortest Vector Problem (SVP) and the Closest Vector Problem (CVP). For any approximation factor $\gamma = \gamma(n) \geq 1$, γ-SVP is the search problem that takes as input a lattice and asks us to find a non-zero vector in this lattice whose length is within a factor of γ of the minimal possible value. γ-CVP is the search problem that takes as input both a lattice and a target vector $\mathbf{t} \in \mathbb{R}^n$ and asks us to find a vector in $\mathcal{L}$ whose distance to $\mathbf{t}$ is within a factor of γ of the minimal distance. The natural decisional variants of these problems are called GapSVP and GapCVP respectively. Specifically, γ-GapSVP asks us to approximate the length of the shortest non-zero vector of a lattice up to a factor of γ, and γ-GapCVP asks us to approximate the distance from $\mathbf{t}$ to the lattice up to a factor of γ.

All four of these problems are interesting for a wide range of approximation factors γ. Indeed, algorithms for these problems have found a remarkable number of applications in computer science (e.g., [26, 27, 23, 36, 21, 35, 15]). And, over the past twenty years, many strong cryptographic primitives have been constructed with their security based on the (worst-case) hardness of γ-GapSVP with approximation factors $\gamma = \text{poly}(n)$ that are polynomial in the ambient dimension (e.g., [4, 34, 18, 17, 37, 40, 30, 11, 12]).

* Supported by the National Science Foundation (NSF) under Grant No. CCF-1320188. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

Due to their importance, there has been much work towards understanding the relationship between these problems (and their many close relatives). Since the fastest known algorithms for these problems run in time that is exponential in the dimension n, even with $\gamma = \mathrm{poly}(n)$, *dimension-preserving* reductions between lattice problems are of particular importance [23, 19, 32, 29, 41, 42]. Perhaps the best-known such reduction is the efficient dimension-preserving reduction from γ-SVP to γ-CVP (and from γ-GapSVP to γ-GapCVP) due to Goldreich, Micciancio, Safra, and Seifert [19]. This proves that the time complexity of γ-SVP, as a function of the dimension n, cannot be more than a polynomial factor higher than the time complexity of γ-CVP. We stress that we could *not* reach this conclusion if the reduction increased the dimension significantly, which is why dimension-preserving reductions interest us.

As a much simpler example, we note that there is a trivial dimension-preserving reduction from γ-GapSVP to γ-SVP that works by just finding a short vector in the input lattice and outputting its length. There is of course a similar reduction for CVP as well. More interestingly, there are relatively simple dimension-preserving *search-to-decision* reductions in the special case when $\gamma = 1$ – i.e., finding *exact* shortest vectors is no harder than computing the *exact* lengths of shortest vectors, and finding exact closest vectors to targets is no harder than computing the exact distances between targets and lattices. (See, e.g., [23] or [33], or simply consider the reductions in the sequel with $\gamma = 1$.) However, prior to this work, there were no known search-to-decision reductions for either SVP or CVP for any approximation factor $\gamma > 1$.

This state of affairs was quite frustrating because, with very few exceptions, our best algorithms for the decision problems work by just solving the corresponding search problem. In other words, we don't really know how to "recognize" that a lattice has a short non-zero vector (or a vector close to some target) without just finding such a vector.[1] If there are better techniques, then we would be thrilled to find them! But, if this extremely natural approach is actually optimal, then it would be nice to prove it by formally reducing the search problems to their decision variants. (Of course, it is conceivable that the search and decision problems have the same complexity, even if no search-to-decision reduction exists. One might reasonably argue that this is even the most likely scenario. But, we can at least hope that Nature would not be so unprincipled.)

The ideal positive result would be an efficient dimension-preserving reduction from γ-SVP to γ-GapSVP for all $\gamma \geq 1$, and likewise for CVP. But, this seems completely out of reach at the moment (perhaps because no such reductions exist). So, as a more approachable goal, we can try to find non-trivial reductions that lose in the approximation factor. Indeed, as we mentioned above, we know that search and decision problems are equivalent in the exact case. Can it truly be the case that equivalence holds when $\gamma = 1$, but *nothing* non-trivial holds for any $\gamma > 1$ – even, say, a reduction from n^{100}-CVP to $(1 + 2^{-n})$-GapCVP?!

1.1 Our results

We make some progress towards resolving these issues by presenting dimension-preserving search-to-decision reductions for both approximate SVP and approximate CVP. Our reduc-

[1] The author knows of three rather specific exceptions. There is an efficient algorithm for $\sqrt{n/\log n}$-GapCVP *with preprocessing* [3], while the best efficient algorithm for search CVP with preprocessing only achieves factor of $\gamma = n/\sqrt{\log n}$ [16]. There is a $2^{n/2+o(n)}$-time algorithm for 2-GapSVP for which no analogous search algorithm is known [1]. And, in the special case of ideal lattices in the ring of integers of a number field, γ-GapSVP is trivial for some values of γ for which γ-SVP appears to be hard. (See, e.g., [38].)

tions generalize the known equivalences in the exact case. But, they lose quite a bit in the approximation factor, and their running times depend on the decision approximation factor. They are therefore primarily interesting when the decision approximation factor is very close to one, as we explain below.

▶ **Theorem 1** (SVP reduction). *For any $\gamma = \gamma(n) \geq 1$ and $a = a(n) \geq \log(n+1)$, there is a dimension-preserving (randomized) reduction from $\gamma^{n/a}$-SVP to γ-GapSVP that runs in time $2^{O(a)} \cdot \gamma^{O(n)}$.*

Theorem 1 is primarily interesting for any $\gamma \leq 1 + O(\log n/n)$ and $a = \Theta(\log n)$. For such parameters, the running time is $\mathrm{poly}(n)$ and the search approximation factor is $\gamma^{O(n/\log n)} \leq O(1)$. However, we note that the theorem is non-trivial whenever we have $1 < \gamma \leq 1 + \varepsilon$ and $a \leq \varepsilon n$, where $\varepsilon > 0$ is some small universal constant.[2]

We actually reduce $\gamma^{n/a}$-SVP to γ-unique SVP, which is a potentially easier problem than γ-GapSVP. (See Definition 16 for the formal definition of γ-unique SVP, and Theorem 24 for the reduction.) The reduction described above then follows from this result together with Lyubashevsky and Micciancio's reduction from γ-unique SVP to γ-GapSVP [29]. We obtain a few additional corollaries as well. E.g., this shows a dimension-preserving reduction from $\sqrt{n}$-CVP to γ-unique SVP (and thus to γ-GapSVP as well) that runs in time $\mathrm{poly}(n) \cdot \gamma^{O(n)}$. This also gives an alternative and arguably more natural proof of Aggarwal and Dubey's result that γ-unique SVP is NP-hard (under randomized reductions) for $\gamma \leq 1 + 1/n^{\varepsilon}$ for any constant $\varepsilon > 0$ [2].

With some more work, we are also able to use our SVP reduction to derive the following search-to-decision reduction for CVP.

▶ **Theorem 2** (CVP reduction). *For any $\gamma = \gamma(n) \geq 1$ and $\ell = \ell(n) \geq 1$, there is a dimension-preserving (randomized) reduction from $\gamma^{n/\ell}$-CVP to γ-GapCVP that runs in time $n^{O(\ell)} \cdot \gamma^{O(n)}$.*

This result is primarily interesting when ℓ is any constant and $\gamma \leq 1 + O(\log n/n)$, in which case the reduction runs in polynomial time and the search approximation factor is $\gamma^{O(n)} \leq \mathrm{poly}(n)$. But, it is still non-trivial for $1 < \gamma \leq 1 + \varepsilon$ and $\ell \leq \varepsilon n/\log n$, where $\varepsilon > 0$ is some universal constant.

We actually show a (deterministic) $n^{O(\ell)}$-time reduction from $\gamma^{n/\ell}$-CVP to γ-GapCVP that works in the presence of a $\mathrm{poly}(n)$-SVP oracle. (See Theorem 28.) The above result then follows from instantiating this oracle via our SVP reduction.

Finally, we show deterministic reductions that achieve much worse parameters.

▶ **Theorem 3** (Deterministic SVP reduction). *For any $\gamma = \gamma(n) \geq 1$ and $p = p(n) \geq 2$, there is a deterministic dimension-preserving reduction from γ'-SVP to γ-GapSVP that runs in time $\mathrm{poly}(n) \cdot p$, where $\gamma' := \gamma^{O(n^2/\log p)}$.*

▶ **Theorem 4** (Deterministic CVP reduction). *For any $\gamma = \gamma(n) \geq 1$ and $p = p(n) \geq 2$, there is a deterministic dimension-preserving reduction from γ'-CVP to γ-GapCVP that runs in time $\mathrm{poly}(n) \cdot p$, where $\gamma' := \gamma^{O(n^2/\log p)}$.*

It is easy to see that our randomized reductions always give a better trade-off between the approximation factor and running time for non-trivial parameters. So, these new reductions

[2] In particular, we can choose ε so that, with $a = \varepsilon n$ and $\gamma \leq 1 + \varepsilon$, we get a reduction from $\gamma^{1/\varepsilon}$-SVP to γ-GapSVP that runs in time $O(2^n)$. For larger values of a or γ, the reduction is subsumed by the known $2^{n+o(n)}$-time algorithm for SVP [1].

are primarily interesting because they are deterministic and because they demonstrate additional potential approaches for future work in this area.

We note that all of our reductions are Cook reductions. (They make many oracle calls, sometimes adaptively.)

1.2 Techniques

1.2.1 SVP

Our main SVP reduction works by finding a sublattice of the input lattice that has one relatively short vector but a significantly longer "second-shortest vector." (I.e., we wish to find a sublattice that satisfies the promise of γ-unique SVP. See Definition 16.) To accomplish this, we use lattice sparsification, which was introduced by Khot [24] and refined in a series of works [14, 16, 42].

The idea of sparsification is to consider the "sparsified" sublattice

$$\mathcal{L}' := \{\mathbf{y} \in \mathcal{L} \,:\, \langle \mathbf{z}, \mathbf{B}^{-1}\mathbf{y} \rangle \equiv 0 \bmod p\} \ ,$$

where p is some prime and $\mathbf{z} \in \mathbb{Z}_p^n$ is chosen uniformly at random. We would like to say that each short vector in $\mathcal{L}$ will land in $\mathcal{L}'$ with probability roughly $1/p$, independently of all other short vectors. Of course, if $\mathbf{x}, \mathbf{y} \in \mathcal{L}$ and $\mathbf{x} = k\mathbf{y}$ for some $k \not\equiv 0 \bmod p$, then clearly $\mathbf{x}$ will land in $\mathcal{L}'$ if and only if $\mathbf{y}$ does. So, we cannot have anything close to independence in this case. [42] shows that this is essentially "the only bad case."

Specifically, a lattice vector $\mathbf{x} \in \mathcal{L}$ is *non-primitive* if $\mathbf{x} = k\mathbf{y}$ for some $k \geq 2$ and $\mathbf{y} \in \mathcal{L}$. Otherwise, $\mathbf{x}$ is *primitive.* [42] showed that sparsification behaves very nicely if we restrict our attention to primitive short lattice vectors. In particular, the distribution of short primitive vectors in the sparsified sublattice $\mathcal{L}'$ behaves similarly to the distribution over the short primitive vectors of $\mathcal{L}$ that selects each vector independently with probability $1/p$. (See Theorem 22 for the precise statement, which is taken directly from [42, Theorem 4.1].)

So, let $\xi(\mathcal{L}, r)$ be the number of primitive lattice points of length at most r. Suppose there exists some radius r such that $\xi(\mathcal{L}, \gamma r)$ is not much larger than $\xi(\mathcal{L}, r)$.[3] Then, if we take $p \approx \xi(\mathcal{L}, \gamma r)$, we can expect $\mathcal{L}'$ to contain a primitive vector of length at most r *but no other primitive vectors of length less than* γr with probability $\Theta(\xi(\mathcal{L}, r)/\xi(\mathcal{L}, \gamma r))$. In other words, with this probability, $\mathcal{L}'$ will be a valid instance of γ-unique SVP with $\lambda_1(\mathcal{L}') \leq r$, so that we can use an oracle for γ-unique SVP to find a non-zero lattice vector of length at most r.

The parameter a in the reduction determines how large of a ratio $\xi(\mathcal{L}, \gamma r)/\xi(\mathcal{L}, r)$ we are "willing to tolerate." In particular, a simple proof shows that, for $a \geq n \log \gamma$, there is always a radius $r \leq \gamma^{n/a} \cdot \lambda_1(\mathcal{L})$ such that this ratio is bounded by $2^{O(a)}$. We can therefore obtain a valid γ-unique SVP instance with $\lambda_1(\mathcal{L}') \leq r \leq \gamma^{n/a} \cdot \lambda_1(\mathcal{L})$ with probability at least $2^{-O(a)}$. So, our main reduction essentially works by sampling $\mathcal{L}'$ repeatedly, a total of $2^{O(a)}$ times, and calling its γ-unique SVP oracle on each sampled sublattice $\mathcal{L}'$.

1.2.2 CVP

Our search-to-decision reduction for CVP is a simple "guided" variant of Babai's celebrated nearest-hyperplane algorithm [7]. Babai's algorithm works by dividing the lattice into $(n-1)$-

[3] It might be helpful to think of the heuristic that $\xi(\mathcal{L}, \gamma r)/\xi(\mathcal{L}, r) \approx \gamma^n$. This holds in the limit as $r \to \infty$, and it holds for random lattices in expectation.

dimensional lattice hyperplanes and then searching inside the closest hyperplane to the target. However, it is possible that the closest lattice hyperplane does not actually contain very close lattice points. As a result, Babai's algorithm can lead to quite large approximation factors.

So, we instead use our GapCVP oracle to "test many nearby hyperplanes" to find one that is guaranteed to contain a γ-approximate closest lattice point. By repeating this n times over hyperplanes of progressively lower dimensions, we will find a γ^n-approximate closest vector to the target. To find a $\gamma^{n/\ell}$-approximate closest vector, we do the same thing with all nearby $(n - \ell)$-dimensional hyperplanes.

In order to make this algorithm efficient, we need to limit the number of hyperplanes that we must consider. This amounts to finding a short non-zero vector in the dual lattice. We can find such a vector by using our search-to-decision reduction for SVP (together with the known reduction from GapSVP to GapCVP [19]). Unfortunately, this costs us a factor of $\gamma^{O(n)}$ in the running time.

1.2.3 Deterministic reductions

Our alternative deterministic search-to-decision reductions for SVP and CVP are very similar to the reduction from unique SVP to GapSVP in [29]. They essentially work by finding the coordinates of a short (or close) lattice vector "bit by bit." I.e., in the CVP case, we first use our GapCVP oracle to compare the distance from the target to all lattice vectors whose last coordinate is even with its distance from all lattice vectors whose last coordinate is odd. If, say, the odd estimate is lower, then we restrict our attention to the lattice coset of all lattice vectors whose last coordinate is odd. We choose the remaining bits similarly, eventually obtaining the coordinates of a relatively close lattice vector. Our more general reductions follow from "working in base p instead of base 2."

1.3 Related work

Some efficient dimension-preserving search-to-decision reductions were known for other lattice problems prior to this work. For example, Regev showed such a reduction for Learning with Errors, an important average-case lattice problem with widespread applications in cryptography [40]. (Both the search and decision versions of LWE are average-case problems.) And, Liu, Lyubashevsky, and Micciancio implicitly use a search-to-decision reduction for Bounded Distance Decoding in their work [28]. Finally, Aggarwal and Dubey showed how to use some of the ideas from [29] to obtain a search-to-decision reduction for unique SVP [2]. While all of these works are quite interesting, they are concerned with promise problems, and not the two most important and natural lattice problems, SVP and CVP.

More generally, this work can be seen as part of the ongoing study of the relationships between lattice problems under dimension-preserving reductions. By now, this area has become quite fruitful (e.g., [23, 19, 32, 29, 42]). See [41] for a brief survey of well-known dimension-preserving reductions between various lattice problems.

Most prior work used sparsification to remove a relatively small number of "annoying" short vectors from a lattice without losing too many "good" short vectors (e.g., [24, 14, 16]). In our main SVP reduction, our goal is instead to remove "all but one" short vector. (Independent work of Bai, Wen, and Stehlé used sparsification in a similar way to reduce Bounded Distance Decoding to unique SVP [8].) To obtain our result, we rely heavily on the sparsification analysis of [42], which is tighter and more general than prior work.

Interestingly, Kumar and Sivakumar used a procedure that is very similar to sparsification in their study of unique SVP, published in 2001 [25]. Indeed, they repeatedly sparsify a lattice with $p = 2$ to obtain a sequence of sublattices such that at least one of these sublattices (1) contains a shortest non-zero vector of the original lattice; and (2) contains no other vectors of this length (up to sign, of course). However, the length of the "second-shortest vector" can be arbitrarily close to that of the shortest vector in their construction, even in a fixed dimension n. I.e., they use a restricted form of sparsification to effectively reduce 1-SVP to 1-unique SVP. Our main SVP reduction can be thought of as an updated version of their result. We use tools that were not available fifteen years ago to obtain a lower bound on the ratio between the shortest vector and the "second-shortest vector" that depends only on the dimension n.

To prove hardness of $(1 + 1/\mathrm{poly}(n))$-unique SVP, Aggarwal and Dubey used the result of Kumar and Sivakumar to show a reduction from SVP to γ-unique SVP that works for a restricted subset of lattices [2]. In particular, Aggarwal and Dubey chose a set of lattices such that, over these lattices (1) SVP is NP-hard (as proven by Khot [24]); and (2) this reduction yields $\gamma = 1 + 1/\mathrm{poly}(n)$. In contrast, we directly reduce 2-SVP to $(1 + 1/\mathrm{poly}(n))$-unique SVP over *all* lattices by using a much stronger (and unfortunately more complicated) form of Kumar and Sivakumar's reduction.

While the author knows of no other use of our specific variant of Babai, we feel that it is quite natural and not particularly novel. For example, a similar idea was used in a different context by Micciancio [32, Corollary 7]. Our primary contribution on this front is the observation that this method gives a non-trivial search-to-decision reduction when the decision approximation factor is very small, and when it is combined with our SVP reduction.

We rely heavily on Lyubashevsky and Micciancio's dimension-preserving reduction from γ-unique SVP to γ-GapSVP [29]. Their result is necessary to prove Theorem 1, and our deterministic "bit-by-bit" SVP reduction is very similar to Lyubashevsky and Micciancio's reduction. The main difference between our deterministic SVP reduction and that of Lyubashevsky and Micciancio is that [29] work only with lattices that satisfy the promise of γ-unique SVP. They show that this promise is enough to guarantee that the γ-GapSVP oracle essentially behaves as an *exact* GapSVP oracle. In contrast, our reduction works over general lattices, so we have to worry about accumulating error. (We also use a different method to "reduce the dimension of the lattice.")

1.4 Directions for future work

We view this paper as a first step towards a better understanding of the relationship between the search and decision variants of approximate SVP and CVP. In particular, we show that efficient, dimension-preserving search-to-decision reductions do in fact exist for approximation factors $\gamma > 1$. Prior to this work, one might have reasonably conjectured that such reductions do not exist for non-trivial parameters. But, our reductions lose quite a bit in the approximation factor, and the running times of our main reductions blow up quickly as the approximation factor increases. They are therefore primarily interesting for very small approximation factors $\gamma = 1 + o(1)$.

Results for such low values of γ have sometimes led to similar results for larger approximation factors. For example, hardness of γ-GapSVP was originally proven for $\gamma = 1 + 2^{-\mathrm{poly}(n)}$ [5], and then for $\gamma = 1 + 1/\mathrm{poly}(n)$ [13], before better inapproximability results were found [31, 24, 20]. We therefore ask whether better search-to-decision reductions exist, and in particular, whether non-trivial efficient dimension-preserving reductions exist for larger approximation factors.

More specifically, we note that our main reductions are only efficient when the decision approximation factor is $\gamma = 1 + O(\log n/n)$ because their running time is proportional to

$\gamma^{O(n)}$. This seems inherent to our technique in the case of SVP, and the CVP reduction suffers the same fate because it uses the SVP reduction as a subroutine. However, we see no reason why the running time should necessarily increase with the approximation factor, and this might simply be an artifact of our techniques. So, perhaps we can find reductions that do not have this problem. (One might try, for example, to eliminate the need for the SVP oracle in our CVP reduction.) Indeed, the reductions in Section 5 manage to avoid this pitfall, but they blow up the approximation factor much more and never actually outperform our main reductions.

In the other direction, we ask whether the search and decision versions of SVP and CVP can be separated in any way. I.e., can we show that, for some $\gamma > 1$, there is no efficient dimension-preserving reduction from γ-CVP to γ-GapCVP or no such reduction from γ-SVP to γ-GapSVP (under reasonable complexity-theoretic assumptions or even restrictions on the behavior of the reduction)? Can we find algorithms that solve the decision problems faster than our current search-based techniques allow (something more general than the rather specific examples mentioned in footnote 1)? Of course, any such result would be a major breakthrough.

2 Preliminaries

We write $\log x$ for the logarithm of x in base 2. We write $\|\mathbf{x}\|$ for the Euclidean norm of $\mathbf{x} \in \mathbb{R}^n$. We omit any mention of the bit length of the input throughout. In particular, all of our algorithms take as input vectors in $\mathbb{R}^n$ (with some reasonable representation) and run in time $f(n) \cdot \mathrm{poly}(m)$ for some f, where m is the maximal bit length of an input vector. We are primarily interested in the dependence on n, so we suppress the factor of $\mathrm{poly}(m)$.

2.1 Lattice basics

A rank d lattice $\mathcal{L} \subset \mathbb{R}^n$ is the set of all integer linear combinations of d linearly independent vectors $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_d)$. $\mathbf{B}$ is called a basis of the lattice and is not unique. We write $\mathcal{L}(\mathbf{B})$ to signify the lattice generated by $\mathbf{B}$. By taking the ambient space to be $\mathrm{span}(\mathcal{L})$, we can implicitly assume that a lattice has full rank n, and we therefore will often implicitly assume that $d = n$.

The dual lattice is

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

Similarly, the dual basis $\mathbf{B}^* := \mathbf{B}(\mathbf{B}^T\mathbf{B})^{-1} = (\mathbf{b}_1^*, \ldots, \mathbf{b}_d^*)$ is the unique list of vectors in $\mathrm{span}(\mathcal{L})$ satisfying $\langle \mathbf{b}_i^*, \mathbf{b}_j \rangle = \delta_{i,j}$. $\mathcal{L}^*$ is itself a rank d lattice with basis $\mathbf{B}^*$.

We write $\lambda_1(\mathcal{L}) := \min_{\mathbf{x} \in \mathcal{L} \setminus \{\mathbf{0}\}} \|\mathbf{x}\|$ for the length of the shortest non-zero vector in the lattice. Similarly, we write $\lambda_2(\mathcal{L}) := \min\{r > 0 \; : \; \dim(\mathrm{span}(\mathcal{L} \cap rB_2^n)) \geq 2\}$ for the length of the shortest lattice vector that is linearly independent from a lattice vector of length $\lambda_1(\mathcal{L})$. For any point $\mathbf{t} \in \mathbb{R}^n$, we write $\mathrm{dist}(\mathbf{t}, \mathcal{L}) := \min_{\mathbf{x} \in \mathcal{L}} \|\mathbf{x} - \mathbf{t}\|$ for the distance between $\mathbf{t}$ and $\mathcal{L}$. The covering radius $\mu(\mathcal{L}) := \max_{\mathbf{t} \in \mathrm{span}(\mathcal{L})} \mathrm{dist}(\mathbf{t}, \mathcal{L})$ is the maximal such distance achievable in the span of the lattice.

The following two bounds will be useful.[4]

[4] We note that tighter bounds exist for the number of lattice points in a ball of radius r [22, 39], but we use the bound of [10] because it is simpler. Using a tighter bound here would improve the hidden constants in the exponents of our running times.

▶ **Theorem 5** ([10, Theorem 2.1]). *For any lattice $\mathcal{L} \subset \mathbb{R}^n$ and $r > 0$,*

$$|\{\mathbf{y} \in \mathcal{L} : \|\mathbf{y}\| \le r\lambda_1(\mathcal{L})\}| \le 2\lceil 2r \rceil^n - 1 .$$

▶ **Lemma 6** ([9, Theorem 2.2]). *For any lattice $\mathcal{L} \subset \mathbb{R}^n$, $\lambda_1(\mathcal{L}^*) \cdot \mu(\mathcal{L}) \le n/2$.*

We derive a simple though rather specific corollary of Lemma 6 that we will use twice. The corollary says that a dual vector $\mathbf{w} \in \mathcal{L}^* \setminus \{\mathbf{0}\}$ that is relatively short, $\|\mathbf{w}\| \le \gamma \cdot \lambda_1(\mathcal{L}^*)$, can be used to partition $\mathcal{L}$ into $(n-1)$-dimensional lattice hyperplanes, such that the closest vector to any target $\mathbf{t}$ must lie in one of the $O(\gamma n)$ hyperplanes closest to $\mathbf{t}$.

▶ **Corollary 7.** *For any lattice $\mathcal{L} \subset \mathbb{R}^n$ with basis $(\mathbf{b}_1, \ldots, \mathbf{b}_n)$ and associated dual basis $(\mathbf{b}_1^*, \ldots, \mathbf{b}_n^*)$, $\gamma \ge 1$, and any target $\mathbf{t} \in \mathbb{R}^n$, if $\|\mathbf{b}_1^*\| \le \gamma \cdot \lambda_1(\mathcal{L}^*)$, then any closest lattice vector to $\mathbf{t}$ must lie in a lattice hyperplane $\mathcal{L}' + i\mathbf{b}_1$, where $\mathcal{L}' := \mathcal{L}(\mathbf{b}_2, \ldots, \mathbf{b}_n)$ and i is an integer with $|i - \langle \mathbf{b}_1^*, \mathbf{t} \rangle| \le \gamma n/2$.*

Proof. Let $\mathbf{y} \in \mathcal{L}$ be a closest lattice vector to $\mathbf{t}$. It follows from the definition of a lattice that $\mathbf{y} \in \mathcal{L}' + i\mathbf{b}_1$ for some integer $i = \langle \mathbf{b}_1^*, \mathbf{y} \rangle$. We have

$$|i - \langle \mathbf{b}_1^*, \mathbf{t} \rangle| = |\langle \mathbf{b}_1^*, \mathbf{y} - \mathbf{t} \rangle| \le \|\mathbf{b}_1^*\| \|\mathbf{y} - \mathbf{t}\| \le \gamma \lambda_1(\mathcal{L}^*) \cdot \mu(\mathcal{L}) \le \gamma n/2 ,$$

where we have used Lemma 6. ◀

2.2 LLL-reduced bases

Given a basis, $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$, we define its Gram-Schmidt orthogonalization $(\widetilde{\mathbf{b}}_1, \ldots, \widetilde{\mathbf{b}}_n)$ by

$$\widetilde{\mathbf{b}}_i = \pi_{\{\mathbf{b}_1, \ldots, \mathbf{b}_{i-1}\}^\perp}(\mathbf{b}_i) ,$$

and the Gram-Schmidt coefficients $\mu_{i,j}$ by

$$\mu_{i,j} = \frac{\langle \mathbf{b}_i, \widetilde{\mathbf{b}}_j \rangle}{\|\widetilde{\mathbf{b}}_j\|^2} .$$

Here, π_A represents orthogonal projection onto the subspace A and $\{\mathbf{b}_1, \ldots, \mathbf{b}_{i-1}\}^\perp$ denotes the subspace of vectors in $\mathbb{R}^n$ that are orthogonal to $\mathbf{b}_1, \ldots, \mathbf{b}_{i-1}$.

▶ **Definition 8.** A basis $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ is *LLL-reduced* if

1. for $1 \le j < i \le n$, $|\mu_{i,j}| \le 1/2$; and
2. for $2 \le i \le n$, $\|\widetilde{\mathbf{b}}_i\|^2 \ge (3/4 - \mu_{i,i-1}^2) \cdot \|\widetilde{\mathbf{b}}_{i-1}\|^2$.

▶ **Theorem 9** ([26]). *There exists an efficient algorithm that takes as input a (basis for) a lattice and outputs an LLL-reduced basis for the lattice.*

▶ **Lemma 10.** *For any lattice $\mathcal{L} \subset \mathbb{R}^n$ with LLL-reduced basis $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ and $\mathbf{y} = \sum a_i \mathbf{b}_i \in \mathcal{L}$, we have*

$$|a_i| \le 2^{3n/2-i} \cdot \frac{\|\mathbf{y}\|}{\lambda_1(\mathcal{L})} ,$$

for all i.

Proof. It follows immediately from the definition of an LLL-reduced basis that $\|\widetilde{\mathbf{b}}_i\| \geq \|\mathbf{b}_1\|/2^{i/2} \geq \lambda_1(\mathcal{L})/2^{n/2}$ for all i. For each i, we have

$$\|\mathbf{y}\| \geq \sum_{j=1}^{n} |a_j \mu_{j,i}| \cdot \|\widetilde{\mathbf{b}}_i\| = \Big(|a_i| - \sum_{j=i+1}^{n} |a_j \mu_{j,i}|\Big) \cdot \|\widetilde{\mathbf{b}}_i\| \geq \Big(|a_i| - \frac{1}{2}\sum_{j=i+1}^{n} |a_j|\Big) \cdot 2^{-n/2} \cdot \lambda_1(\mathcal{L}) \; .$$

In particular, $|a_n| \leq 2^{n/2} \cdot \|\mathbf{y}\|/\lambda_1(\mathcal{L})$. We assume for induction that $|a_j| \leq 2^{3n/2-j} \cdot \|\mathbf{y}\|/\lambda_1(\mathcal{L})$ for all j with $i < j \leq n$. Then, plugging in to the above, we have

$$\|\mathbf{y}\| \geq |a_i| \cdot 2^{-n/2} \cdot \lambda_1(\mathcal{L}) - \sum_{j=i+1}^{n} 2^{n-j-1}\|\mathbf{y}\| \geq |a_i| \cdot 2^{-n/2} \cdot \lambda_1(\mathcal{L}) - 2^{n-i-1} \cdot \|\mathbf{y}\| \; .$$

The result follows by rearranging. ◀

▶ **Lemma 11** ([7]). *If $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ is an LLL-reduced basis, then $\mu(\mathcal{L}) \leq \sqrt{n}2^{n/2-1} \cdot \|\widetilde{\mathbf{b}}_n\|$.*

2.3 Lattice problems

We now list the computational problems that concern us. All of the below definitions are standard.

▶ **Definition 12.** For any parameter $\gamma = \gamma(n) \geq 1$, γ-SVP (the Shortest Vector Problem) is the search problem defined as follows: The input is a basis $\mathbf{B}$ for a lattice $\mathcal{L} \subset \mathbb{R}^n$. The goal is to output a lattice vector $\mathbf{x}$ with $0 < \|\mathbf{x}\| \leq \gamma\lambda_1(\mathcal{L})$.

▶ **Definition 13.** For any parameter $\gamma = \gamma(n) \geq 1$, γ-CVP (the Closest Vector Problem) is the search problem defined as follows: The input is a basis $\mathbf{B}$ for a lattice $\mathcal{L} \subset \mathbb{R}^n$ and a target vector $\mathbf{t} \in \mathbb{R}^n$. The goal is to output a lattice vector $\mathbf{x}$ with $\|\mathbf{x} - \mathbf{t}\| \leq \gamma \operatorname{dist}(\mathbf{t}, \mathcal{L})$.

▶ **Definition 14.** For any parameter $\gamma = \gamma(n) \geq 1$, the decision problem γ-GapSVP is defined as follows: The input is a basis $\mathbf{B}$ for a lattice $\mathcal{L} \subset \mathbb{R}^n$ and a number $d > 0$. The goal is to output yes if $\lambda_1(\mathcal{L}) < d$ and no if $\lambda_1(\mathcal{L}) \geq \gamma \cdot d$.

▶ **Definition 15.** For any parameter $\gamma = \gamma(n) \geq 1$, the decision problem γ-GapCVP is defined as follows: The input is a basis $\mathbf{B}$ for a lattice $\mathcal{L} \subset \mathbb{R}^n$, a target $\mathbf{t} \in \mathbb{R}^n$, and a number $d > 0$. The goal is to output yes if $\operatorname{dist}(\mathbf{t}, \mathcal{L}) < d$ and no if $\operatorname{dist}(\mathbf{t}, \mathcal{L}) \geq \gamma \cdot d$.

▶ **Definition 16.** For any parameter $\gamma = \gamma(n) \geq 1$, γ-uSVP (the Unique Shortest Vector Problem) is the search promise problem defined as follows: The input is a basis $\mathbf{B}$ for a lattice $\mathcal{L} \subset \mathbb{R}^n$ with $\lambda_2(\mathcal{L}) \geq \gamma(n) \cdot \lambda_1(\mathcal{L})$. The goal is to output a lattice vector $\mathbf{x}$ with $\|\mathbf{x}\| = \lambda_1(\mathcal{L})$.

2.4 Known results

We will need the following known reductions and hardness results.

▶ **Theorem 17** ([24]). *For any constant $\gamma \geq 1$, γ-GapSVP (and therefore γ-SVP) is NP-hard under randomized reductions.*

▶ **Theorem 18** ([19]). *For any $\gamma \geq 1$, there is an efficient dimension-preserving reduction from γ-GapSVP to γ-GapCVP (and from γ-SVP to γ-CVP).*

▶ **Theorem 19** ([29, Theorem 6.1]). *For any $1 \leq \gamma(n) \leq \mathrm{poly}(n)$, there is an efficient dimension-preserving reduction from γ-uSVP to γ-GapSVP.*

▶ **Theorem 20** ([33, Theorem 4.2]). *There is an efficient reduction from $\sqrt{n}$-CVP to $\sqrt{2}$-SVP. Furthermore, all of the oracle calls of the reduction are made in dimension $n+1$, where n is the input dimension.*

Reductions like that of Theorem 20 that increase the dimension by one are good enough for nearly all applications of perfectly dimension-preserving reductions. But, we can use a simple idea to convert Theorem 20 into a reduction that preserves the dimension exactly. (Micciancio uses essentially the same trick in the proof of [32, Corollary 7].)

▶ **Corollary 21.** *There is a dimension-preserving efficient reduction from $\sqrt{n}$-CVP to $\sqrt{2}$-SVP.*

Proof. On input $\mathbf{t} \in \mathbb{R}^n$ and a lattice $\mathcal{L} \subset \mathbb{R}^n$, the reduction first uses its SVP oracle to find a vector $\mathbf{b}_1^* \in \mathcal{L}^*$ in the dual with $0 < \|\mathbf{b}_1^*\| < 2\lambda_1(\mathcal{L}^*)$. Let $\mathbf{B}^* := (\mathbf{b}_1^*, \ldots, \mathbf{b}_n^*)$ be a basis for $\mathcal{L}^*$, and let $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ be the associated primal basis. (Since $\mathbf{b}_1^*$ is a primitive lattice vector, it is always possible to find such a basis.) Let $\mathcal{L}' := \mathcal{L}(\mathbf{b}_2, \ldots, \mathbf{b}_n)$ be the lattice generated by $\mathbf{B}$ with the first basis vector removed. Finally, let $a := \langle \mathbf{b}_1^*, \mathbf{t} \rangle$. For $i = \lfloor a \rceil - n, \ldots, \lceil a \rceil + n$, the reduction runs the procedure from Theorem 20 on input $\mathbf{t} - i\mathbf{b}_1$ and $\mathcal{L}'$, receiving as output $\mathbf{y}_i \in \mathcal{L}'$. The reduction then simply outputs a closest vector to $\mathbf{t}$ amongst the vectors $\mathbf{y}_i + i\mathbf{b}_1 \in \mathcal{L}$.

It is clear that the reduction is efficient. Furthermore, note that the reduction only uses the procedure from Theorem 20 with $(n-1)$-dimensional input. (Formally, we must project $\mathcal{L}'$ and $\mathbf{t} - i\mathbf{b}_1$ onto $\mathrm{span}(\mathcal{L}') \cong \mathbb{R}^{n-1}$.) Since that procedure increases dimension by one, this new reduction preserves dimension. For correctness, we note that Corollary 7 implies that there is a closest vector to $\mathbf{t}$ in one of the lattice hyperplanes $\mathcal{L}' + i\mathbf{y}_i$. The result then follows from Theorem 20. ◀

2.5 A note on decision and estimation

Formally, we consider *gapped decision problems*, which take as input a number $d > 0$ and some additional input I and require us to output YES if $f(I) \leq d$ and NO if $f(I) > \gamma d$, where f is some function and γ is the approximation factor. (For example, I may be some representation of a lattice and $f(I)$ may be the length of the shortest vector in the lattice.) However, it is sometimes convenient to work with *estimation problems*, which take only I as input and ask for a numerical output $\tilde{d}$ with $f(I) \leq \tilde{d} \leq \gamma f(I)$.

For the specific problems that we consider (and most "sufficiently nice" problems), the estimation variants are equivalent to the gapped decision problems as long as the lattice is "represented reasonably" by the input. For example, if $f(I)$ can be represented as a string of length at most $\mathrm{poly}(|I|)$ (e.g., $f(I)$ might be a rational number with bounded numerator and denominator), then we can use binary search and a gapped decision oracle to estimate $f(I)$ efficiently. This is true, for example, whenever the input is interpreted as a list of vectors with rational coordinates, using the standard representation of rational numbers. (See [33] for a careful discussion of this and related issues in the context of lattice problems.) We therefore make no distinction between gapped decision problems and estimation problems in the sequel, without worrying about the specific form of our input, or more generally, the specific representation of numbers.

3 Reducing SVP to uSVP (and GapSVP) via sparsification

3.1 Sparsification

For a lattice $\mathcal{L} \subset \mathbb{R}^n$, we write $\mathcal{L}_{\mathrm{prim}}$ for the set of all primitive vectors in $\mathcal{L}$, and $\xi(\mathcal{L}, r) := |\mathcal{L}_{\mathrm{prim}} \cap rB_2^n|/2$ for the number of primitive lattice vectors contained in a (closed) ball of radius r around the origin (counting $\mathbf{x}$ and $-\mathbf{x}$ as a single vector). The following theorem from [42] shows that sparsification behaves nicely with respect to primitive vectors, which is enough for our use case.

▶ **Theorem 22** ([42, Theorem 4.1])**.** *For any lattice $\mathcal{L} \subset \mathbb{R}^n$ with basis $\mathbf{B}$, primitive lattice vectors $\mathbf{y}_0, \mathbf{y}_1, \dots, \mathbf{y}_N \in \mathcal{L}_{\mathrm{prim}}$ with $\mathbf{y}_i \neq \pm\mathbf{y}_0$ for all $i > 0$, and prime $p \geq 101$, if $\xi(\mathcal{L}, \|\mathbf{y}_i\|) \leq p/(20\log p)$ for all i, then*

$$\frac{1}{p} - \frac{N}{p^2} \leq \Pr\left[\langle \mathbf{z}, \mathbf{B}^{-1}\mathbf{y}_0\rangle \equiv 0 \bmod p \text{ and } \langle \mathbf{z}, \mathbf{B}^{-1}\mathbf{y}_i\rangle \not\equiv 0 \bmod p \ \forall i > 0\right] \leq \frac{1}{p} \ ,$$

where $\mathbf{z} \in \mathbb{Z}_p^n$ is chosen uniformly at random.

From this, we can immediately derive the following proposition, which is a slight variant of [42, Proposition 4.2].

▶ **Proposition 23.** *There is an efficient (randomized) algorithm that takes as input a basis $\mathbf{B}$ for a lattice $\mathcal{L} \subset \mathbb{R}^n$ and a prime $p \geq 101$ and outputs a full-rank sublattice $\mathcal{L}' \subseteq \mathcal{L}$ such that for any r_1, r_2, with $\lambda_1(\mathcal{L}) \leq r_1 \leq r_2 < p\lambda_1(\mathcal{L})$ and $\xi(\mathcal{L}, r_2) \leq p/(20\log p)$, we have*

$$\Pr[\lambda_1(\mathcal{L}') \leq r_1 \text{ and } \lambda_2(\mathcal{L}') > r_2] \geq \frac{\xi(\mathcal{L}, r_1)}{p} \cdot \left(1 - \frac{\xi(\mathcal{L}, r_2)}{p}\right) .$$

Proof. The algorithm takes as input a basis $\mathbf{B}$ for a lattice $\mathcal{L} \subset \mathbb{R}^n$. It then samples $\mathbf{z} \in \mathbb{Z}_p^n$ uniformly at random and outputs

$$\mathcal{L}' := \{\mathbf{y} \in \mathcal{L} \ : \ \langle \mathbf{z}, \mathbf{B}^{-1}\mathbf{y}\rangle \equiv 0 \bmod p\} .$$

Let $N := \xi(\mathcal{L}, r_2) \leq p/(20\log p)$, and let $\mathbf{y}_1, \dots, \mathbf{y}_N \in \mathcal{L}$ be the N unique primitive vectors in $\mathcal{L}$ satisfying $\|\mathbf{y}_1\| \leq \cdots \leq \|\mathbf{y}_N\| \leq r_2$ (taking only one vector from each pair $\pm\mathbf{y}$). Note that we have $\lambda_1(\mathcal{L}') \leq r_1$ and $\lambda_2(\mathcal{L}') > r_2$ if and only if $\mathbf{y}_i \in \mathcal{L}'$ for some $i \leq \xi(\mathcal{L}, r_1)$ and $\mathbf{y}_j \notin \mathcal{L}'$ for all $j \neq i$. (Here, we have used the fact that $r_2 < p\lambda_1(\mathcal{L})$ to guarantee that vectors of the form $p\mathbf{y}_i \in \mathcal{L}'$ do not cause $\lambda_2(\mathcal{L}')$ to be less than r_2.)

Applying Theorem 22, we see that this happens with probability at least $1/p-(N-1)/p^2 > 1/p - N/p^2$ for any fixed i. The result follows by noting that these are disjoint events, so that the probability that at least one of these events occurs is the sum of their individual probabilities, which is at least

$$\xi(\mathcal{L}, r_1) \cdot \left(\frac{1}{p} - \frac{N}{p^2}\right) = \frac{\xi(\mathcal{L}, r_1)}{p} \cdot \left(1 - \frac{\xi(\mathcal{L}, r_2)}{p}\right) ,$$

as needed. ◀

3.2 The reduction

We now present the main step in our search-to-decision reduction for SVP.

▶ **Theorem 24.** *For any $\gamma = \gamma(n) \geq 1$ and $a = a(n) \geq \log(n+1)$, there is a dimension-preserving (randomized) reduction from $\gamma^{n/a}$-SVP to γ-uSVP that runs in time $2^{O(a)} \cdot \gamma^{O(n)}$.*

Proof. We may assume without loss of generality that $a \le n/2$, since the result is trivial for larger a. (There are known $2^{O(n)}$-time algorithms for SVP [6, 1].) We may also assume that $2^a > \gamma^n$, since this does not affect the asymptotic running time. Let $k := \lceil 4^{an/(n-a)} \rceil = 2^{O(a)}$.

On input a lattice $\mathcal{L} \subset \mathbb{R}^n$, the reduction does the following k times. For $i = 0, \ldots, \ell := \lfloor n/a \rfloor$, let p_i be a prime with $2k^{i+1} < p_i < 4k^{i+1}$. The reduction calls the procedure from Proposition 23 with input $\mathcal{L}$ and p_i, receiving as output $\mathcal{L}_i$. It then calls its uSVP oracle on each $\mathcal{L}_i$, receiving as output $\mathbf{x}_i$. Finally, it simply outputs a shortest non-zero $\mathbf{x}_i$.

It is clear that the reduction runs in time $\mathrm{poly}(n) \cdot k = 2^{O(a)}$, as needed.[5] For each i, let r_i be minimal such that $\xi(\mathcal{L}, r_i) \ge k^i$. In particular, $r_0 = \lambda_1(\mathcal{L})$. And, recalling the definition of ξ, we have

$$|\mathcal{L} \cap r_\ell B_2^n| > 2\xi(\mathcal{L}, r_\ell) \ge 2k^\ell > 2 \cdot (4^{an/(n-a)})^{n/a-1} = 2 \cdot 4^n \; .$$

So, applying Theorem 5, we have that $r_\ell / r_0 = r_\ell / \lambda_1(\mathcal{L}) > 2$.

Therefore, there exists an i such that $r_{i+1}/r_i > 2^{1/\ell} \ge 2^{a/n} > \gamma$. Let j be minimal such that $r_{j+1}/r_j > \gamma$. In particular, this means that $\xi(\mathcal{L}, \gamma r_j) < k^{j+1}$ and $\gamma r_j \le 2\gamma\lambda_1(\mathcal{L}) < p_j \lambda_1(\mathcal{L})$. So, we may apply Proposition 23 to obtain

$$\begin{aligned}\Pr[\lambda_1(\mathcal{L}_j) \le r_j \text{ and } \lambda_2(\mathcal{L}_j) > \gamma r_j] &\ge \frac{\xi(\mathcal{L}, r_j)}{p_j} - \frac{\xi(\mathcal{L}, r_j)\xi(\mathcal{L}, \gamma r_j)}{p_j^2} \\ &> \frac{k^j}{p_j} \cdot \Big(1 - \frac{k^{j+1}}{p_j}\Big) \\ &> \frac{1}{2k} \; .\end{aligned}$$

Therefore, after running the above procedure k times, the algorithm will output a non-zero vector of length at most r_i with at least some positive constant probability.

Finally, by the definition of r_j, we have $r_j/\lambda_1(\mathcal{L}) = r_j/r_0 \le \gamma^j \le \gamma^\ell \le \gamma^{n/a}$. Therefore, the algorithm outputs a $\gamma^{n/a}$-approximate shortest vector with at least constant probability, as needed. ◀

3.3 Corollaries

From this, we derive some immediate corollaries. The first is our main SVP result.

Proof of Theorem 1. We may assume without loss of generality that $\gamma \le 2$, since otherwise the result is trivial as there are known $2^{O(n)}$-time algorithms for SVP. Therefore, by Theorem 19, there is an efficient dimension-preserving reduction from γ-uSVP to γ-GapSVP. The result then follows from Theorem 24. ◀

By combining Theorem 24 with Corollary 21, we obtain the following reduction from $\sqrt{n}$-CVP to γ-uSVP (and therefore γ-GapSVP).

▶ **Corollary 25.** *For any $\gamma = \gamma(n) \ge 1$, there is a dimension-preserving (randomized) reduction from $\sqrt{n}$-CVP to γ-uSVP that runs in time* $\mathrm{poly}(n) \cdot \gamma^{O(n)}$.

Similarly, there is a dimension-preserving (randomized) reduction from $\sqrt{n}$-CVP to γ-GapSVP with the same running time.

[5] The reader might notice that the theorem quotes a running time of $2^{O(a)} \cdot \gamma^{O(n)}$ instead of just $2^{O(a)}$. Note that this looser bound on the running time is exactly what allowed us to assume $2^a > \gamma^n$ above. Equivalently, we could simply require $a > n \log \gamma$ in the theorem statement and achieve a running time of $2^{O(a)}$. But, we wish to avoid misunderstanding by explicitly stating that the running time is at least $\gamma^{O(n)}$ in the theorem statement.

Proof. Setting $a := 2n\log(\gamma) + \log(n+1)$ in Theorem 24 gives a reduction from $\sqrt{2}$-SVP to γ-uSVP with the claimed running time. The first result then follows from Corollary 21. The second result follows from Theorem 19. ◀

We also obtain an alternative proof of the hardness of $(1 + 1/\mathrm{poly}(n))$-uSVP, as originally shown by Aggarwal and Dubey [2].

▶ **Corollary 26.** *For any constant $\varepsilon > 0$, $(1 + 1/n^\varepsilon)$-uSVP is NP-hard (under randomized reductions).*

Proof. For $\gamma \le 1 + O(\log n/n)$, taking $a := n\log(\gamma) + \log(n+1)$ in Theorem 24 gives a polynomial-time reduction from 2-SVP to γ-uSVP. It then follows from Theorem 17 that γ-uSVP is NP-hard (under randomized reductions).

The full result then follows by noting that there is a simple reduction from $(1 + 1/n^\varepsilon)$-uSVP to $(1 + 1/n)$-uSVP for any constant $\varepsilon \in (0, 1)$. In particular, given input $\mathcal{L} \subset \mathbb{R}^n$ with basis $\mathbf{B} := (\mathbf{b}_1, \ldots, \mathbf{b}_n)$, let $N := \lceil n^{1/\varepsilon} \rceil = \mathrm{poly}(n)$, and let $r := 3\|\mathbf{b}_1\| > 2\lambda_1(\mathcal{L})$. Let $\mathcal{L}' := \mathcal{L}(\mathbf{b}_1, \ldots, \mathbf{b}_n, r\mathbf{e}_{n+1}, \ldots, r\mathbf{e}_N) \subset \mathbb{R}^N$ be the rank N lattice obtained by "adding $N - n$ perpendicular vectors of length r to $\mathcal{L}$." The result follows by noting that $N^\varepsilon \ge n$ so that $\mathcal{L}'$ is a valid instance of $(1 + 1/N^\varepsilon)$-uSVP if $\mathcal{L}$ is a valid instance of $(1 + 1/n)$-uSVP, and the two instances have the same solution. ◀

Finally, we note that a reduction to GapSVP immediately implies a reduction to GapCVP, by Theorem 18. We will need this in the next section.

▶ **Corollary 27.** *For any $\gamma = \gamma(n) \ge 1$ and $a = a(n) \ge \log(n+1)$, there is a dimension-preserving (randomized) reduction from $\gamma^{n/a}$-SVP to γ-GapCVP that runs in time $2^{O(a)} \cdot \gamma^{O(n)}$.*

Proof. Combine Theorem 1 with Theorem 18. ◀

4 Reducing CVP to GapCVP

▶ **Theorem 28.** *For any $\gamma = \gamma(n) \ge 1$, $h = h(n) \ge 1$, and integer $\ell = \ell(n) \ge 1$, there is a (deterministic) algorithm with access to a γ-GapCVP oracle and a h-SVP oracle that solves $\gamma^{n/\ell}$-CVP in time $(\mathrm{poly}(n) \cdot h)^\ell$. Furthermore, the dimension of the algorithm's oracle calls never exceeds the dimension of the input lattice.*

Proof. We show how to handle the case $\ell = 1$ and then describe how to extend the result to arbitrary ℓ. On input a lattice $\mathcal{L} \subset \mathbb{R}^n$ and $\mathbf{t} \in \mathbb{R}^n$, the algorithm behaves as follows. If $n = 1$, then it solves the CVP instance directly. Otherwise, it first uses its SVP oracle to find a dual vector $\mathbf{b}_1^* \in \mathcal{L}^*$ with $\|\mathbf{b}_1^*\| \le h \cdot \lambda_1(\mathcal{L}^*)$. Let $\mathbf{b}_2^*, \ldots, \mathbf{b}_n^* \in \mathcal{L}^*$ such that $(\mathbf{b}_1^*, \ldots, \mathbf{b}_n^*)$ is a basis of $\mathcal{L}^*$, and let $(\mathbf{b}_1, \ldots, \mathbf{b}_n) \in \mathcal{L}$ be the associated basis of the primal. (This is always possible if $\mathbf{b}_1^*$ is primitive in $\mathcal{L}^*$. If $\mathbf{b}_1^*$ is not primitive, then we can simply replace it with a primitive vector that is a scalar multiple of $\mathbf{b}_1^*$.)

Next, let $a := \langle \mathbf{b}_1^*, \mathbf{t} \rangle$ and $\mathcal{L}' := \mathcal{L}(\mathbf{b}_2, \ldots, \mathbf{b}_n)$. Then, for $i = \lfloor a - h \cdot n \rfloor, \ldots, \lceil a + h \cdot n \rceil$, the algorithm uses its GapCVP oracle to compute d_i such that $\mathrm{dist}(\mathbf{t} - i \cdot \mathbf{b}_1, \mathcal{L}') \le d_i \le \gamma \cdot \mathrm{dist}(\mathbf{t} - i \cdot \mathbf{b}_1, \mathcal{L}')$. The algorithm then picks an index i such that d_i is minimal and calls itself recursively on input $\mathcal{L}'$ and $\mathbf{t} - i \cdot \mathbf{b}_1$, receiving as output $\mathbf{y} \in \mathcal{L}'$. Finally, it outputs $\mathbf{y} + i\mathbf{b}_1 \in \mathcal{L}$.

It is clear that the running time is as claimed. By Corollary 7, there must exist some j such that $\mathrm{dist}(\mathbf{t} - j \cdot \mathbf{b}_1, \mathcal{L}') = \mathrm{dist}(\mathbf{t}, \mathcal{L})$, so that $d_j \le \gamma\,\mathrm{dist}(\mathbf{t}, \mathcal{L})$. Therefore, $d_i \le \gamma \cdot \mathrm{dist}(\mathbf{t}, \mathcal{L})$, and $\mathrm{dist}(\mathbf{t} - i \cdot \mathbf{b}_1, \mathcal{L}') \le \gamma \cdot \mathrm{dist}(\mathbf{t}, \mathcal{L})$. The result then follows from induction on the dimension n.

To handle arbitrary $\ell \geq 1$, the algorithm simply tries all recursive paths up to depth ℓ and chooses the path that yields the lowest approximate distance according to its γ-GapCVP oracle. Note that there are at most $(2hn+2)^\ell = (\mathrm{poly}(n) \cdot h)^\ell$ such paths, so the running time is as claimed. ◀

We obtain our main CVP reduction by combining the above result with our SVP reduction.

Proof of Theorem 2. We may assume without loss of generality that ℓ is an integer.

We can instantiate the SVP oracle required in Theorem 28 above by using Corollary 27. In particular, taking $a := n \log \gamma + \log(n+1)$ in Corollary 25 gives a reduction from 2-SVP to γ-GapCVP that runs in time $\mathrm{poly}(n) \cdot \gamma^{O(n)}$. By using this reduction to instantiate the h-SVP oracle in Theorem 28 with $h = 2$, we get a dimension-preserving reduction from $\gamma^{n/\ell}$-CVP to γ-CVP that runs in time $n^{O(\ell)} \cdot \gamma^{O(n)}$, as needed. ◀

5 Deterministic reductions

We now show deterministic search-to-decision reductions for SVP and CVP that achieve significantly worse parameters. Both reductions use the same basic idea, which is essentially to "find the coordinates of a short (or close) lattice point bit-by-bit."

5.1 The deterministic CVP reduction

We present the CVP reduction first because it is simpler.

Proof of Theorem 4. We may assume without loss of generality that p is an integer and $\gamma^2 < p$, since the result is trivial for larger γ.

On input a lattice $\mathcal{L} \subset \mathbb{R}^n$ and $\mathbf{t} \in \mathbb{R}^n$, the reduction behaves as follows. If $n = 1$, then it solves the CVP instance directly. Otherwise, it first uses the procedure from Theorem 9, to compute an LLL-reduced basis $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ for $\mathcal{L}$. It then finds the nth coordinate of a close lattice vector to $\mathbf{t}$ "in base p," as follows. Let $\mathbf{t}_0 = \mathbf{t}$, and let $\mathcal{L}_i := \mathcal{L}(\mathbf{b}_1, \ldots, \mathbf{b}_{n-1}, p^i \cdot \mathbf{b}_n)$ for all i. For $i = 0, \ldots, \ell - 1$, with $\ell \geq 1$ to be set in the analysis, the reduction uses its γ-GapCVP oracle to compute $d_{i,0}, \ldots, d_{i,p-1}$ such that $\mathrm{dist}(\mathbf{t}_i - jp^i \cdot \mathbf{b}_n, \mathcal{L}_{i+1}) \leq d_{i,j} \leq \gamma \,\mathrm{dist}(\mathbf{t}_i - jp^i \cdot \mathbf{b}_n, \mathcal{L}_{i+1})$. It then sets $\mathbf{t}_{i+1} = \mathbf{t}_i - j \cdot p^i \cdot \mathbf{b}_n$, where j is chosen such that $d_{i,j}$ is minimal.

Let

$$\mathbf{t}' := \mathbf{t}_\ell - p^\ell \cdot \left\lfloor \frac{\langle \mathbf{t}_\ell, \widetilde{\mathbf{b}}_n \rangle}{p^\ell \cdot \|\widetilde{\mathbf{b}}_n\|^2} \right\rceil \cdot \mathbf{b}_n \; .$$

The reduction then calls itself recursively on input $\mathcal{L}' := \mathcal{L}(\mathbf{b}_1, \ldots, \mathbf{b}_{n-1})$ and $\mathbf{t}'$, receiving as output $\mathbf{y}' \in \mathcal{L}'$. Finally, the reduction outputs $\mathbf{y}' + \mathbf{t} - \mathbf{t}' \in \mathcal{L}$.

Take

$$\ell := \left\lceil \frac{n + \log n + 2}{2 \log(p/\gamma)} \right\rceil = O(n/\log p) \; .$$

It is clear that the running time is as claimed. We first show by induction that $\mathrm{dist}(\mathbf{t}_i, \mathcal{L}_i) \leq \gamma^i \cdot \mathrm{dist}(\mathbf{t}, \mathcal{L})$. For $i = 0$, this is trivial. For any $i > 0$, let $\mathbf{x}$ be a closest vector in $\mathcal{L}_{i-1}$ to $\mathbf{t}_{i-1}$, and assume for induction that $\|\mathbf{x} - \mathbf{t}_{i-1}\| \leq \gamma^{i-1} \,\mathrm{dist}(\mathbf{t}, \mathcal{L})$. Note that we can write $\mathbf{x} = \mathbf{x}' + cp^{i-1} \cdot \mathbf{b}_n$, where $\mathbf{x}' \in \mathcal{L}_i$ and $c \in \{0, \ldots, p-1\}$. In particular,

$$d_{i,c} \leq \gamma \,\mathrm{dist}(\mathbf{t}_{i-1} - cp^{i-1} \cdot \mathbf{b}_n, \mathcal{L}_i) = \gamma \|\mathbf{x} - \mathbf{t}_{i-1}\| \leq \gamma^i \,\mathrm{dist}(\mathbf{t}, \mathcal{L}) \; .$$

It follows from the definition of $\mathbf{t}_i$ that $\operatorname{dist}(\mathbf{t}_i, \mathcal{L}_i) \leq d_{i,c} \leq \gamma^i \operatorname{dist}(\mathbf{t}, \mathcal{L})$, as needed.

We now wish to show that $\operatorname{dist}(\mathbf{t}', \mathcal{L}') = \operatorname{dist}(\mathbf{t}_\ell, \mathcal{L}_\ell)$. Suppose not. Then, clearly we have that $\operatorname{dist}(\mathbf{t}_\ell, \mathcal{L}_\ell) \geq p^\ell \|\widetilde{\mathbf{b}}_n\|/2$. (To see this, consider the "distance in the direction of $\widetilde{\mathbf{b}}_n$.") Combining this with the above, we have $\operatorname{dist}(\mathbf{t}, \mathcal{L}) \geq (p/\gamma)^\ell \cdot \|\widetilde{\mathbf{b}}_n\|/2 \geq \sqrt{n}2^{n/2} \cdot \|\widetilde{\mathbf{b}}_n\|$, contradicting Lemma 11.

Combining everything together, we see that $\operatorname{dist}(\mathbf{t}', \mathcal{L}') \leq \gamma^\ell \cdot \operatorname{dist}(\mathbf{t}, \mathcal{L})$. Finally, we assume for induction that $\|\mathbf{y}' - \mathbf{t}'\| \leq \gamma^{\ell \cdot (n-1)} \operatorname{dist}(\mathbf{t}', \mathcal{L}') \leq \gamma^{\ell \cdot n} \operatorname{dist}(\mathbf{t}, \mathcal{L})$. It follows that $\|(\mathbf{y}' - \mathbf{t}' + \mathbf{t}) - \mathbf{t}\| = \|\mathbf{y}' - \mathbf{t}'\| \leq \gamma^{\ell n} \operatorname{dist}(\mathbf{t}, \mathcal{L}) = \gamma^{O(n^2/\log p)} \operatorname{dist}(\mathbf{t}, \mathcal{L})$, as needed. ◀

5.2 The deterministic SVP reduction

The SVP reduction uses essentially the same idea, but it is a bit more technical because the GapSVP oracle is so much weaker. The reduction is very similar to the reduction from unique SVP to GapSVP in [29].

Proof of Theorem 3. We may assume without loss of generality that p is a prime and $\gamma^3 < p$, since the result is trivial for larger γ. On input a lattice $\mathcal{L} \subset \mathbb{R}^n$, the reduction behaves as follows. If $n = 1$, it solves the SVP instance directly. Otherwise, let $(\mathbf{b}_1, \ldots, \mathbf{b}_n)$ be an LLL-reduced basis for $\mathcal{L}$ (which we can compute efficiently by Theorem 9).

Our goal is to compute a sequence of sublattices $\mathcal{L} = \mathcal{L}^{(0)} \subset \mathcal{L}^{(1)} \subset \cdots \subset \mathcal{L}^{(\ell)}$, with $\ell \geq 1$ to be set in the analysis, such that the index of $\mathcal{L}^{(i+1)}$ over $\mathcal{L}^{(i)}$ is p, and $\lambda_1(\mathcal{L}^{(i+1)}) \leq \gamma \cdot \lambda_1(\mathcal{L}^{(i)})$. In particular, we will take $\mathcal{L}^{(i)} := \mathcal{L}(\mathbf{b}_1, \ldots, \mathbf{b}_{n-2}, a_1^{(i)}\mathbf{b}_{n-1} + a_2^{(i)}\mathbf{b}_n, a_3^{(i)}\mathbf{b}_n)$ for some $a_1^{(i)}, a_2^{(i)}, a_3^{(i)}$, starting with $a_1^{(0)} := 1$, $a_2^{(0)} := 0$, and $a_3^{(0)} := 1$. To compute the remaining coefficients, the reduction behaves as follows for $i = 0, \ldots, \ell - 1$. For $j = 0, \ldots, p-1$, let $\mathcal{L}_{i,j} := \mathcal{L}(\mathbf{b}_1, \ldots, \mathbf{b}_{n-2}, a_1^{(i)}\mathbf{b}_{n-1} + a_2^{(i)}\mathbf{b}_n + ja_3^{(i)}\mathbf{b}_n, pa_3^{(i)}\mathbf{b}_n)$ and let $\mathcal{L}_{i,p} := \mathcal{L}(\mathbf{b}_1, \ldots, \mathbf{b}_{n-2}, p \cdot (a_1^{(i)}\mathbf{b}_{n-1} + a_2^{(i)}\mathbf{b}_n), a_3^{(i)}\mathbf{b}_n)$. For each j, the reduction uses its GapSVP oracle to compute $d_{i,j}$ such that $\lambda_1(\mathcal{L}_{i,j}) \leq d_{i,j} \leq \gamma\lambda_1(\mathcal{L}_{i,j})$. Let j such that $d_{i,j}$ is minimal. The reduction then sets the coefficients so that $\mathcal{L}^{(i+1)} := \mathcal{L}_{i,j}$.[6]

Let k_1 be the largest power of p that divides $a_1^{(\ell)}$, and let k_2 be the largest power of p that divides $a_3^{(\ell)}$. If $k_1 \geq k_2$, the reduction sets $\mathcal{L}' := \mathcal{L}(\mathbf{b}_1, \ldots, \mathbf{b}_{n-2}, \mathbf{b}_n)$ to be "$\mathcal{L}$ without $\mathbf{b}_{n-1}$." Otherwise, it sets $\mathcal{L}' := \mathcal{L}(\mathbf{b}_1, \ldots, \mathbf{b}_{n-2}, a_1^{(\ell)}\mathbf{b}_{n-1} + a_2^{(\ell)}\mathbf{b}_n)$ to be "$\mathcal{L}^{(\ell)}$ without $\mathbf{b}_n$." It then calls itself recursively on $\mathcal{L}'$ and returns the result.

Take

$$\ell := \left\lceil \frac{n+3}{\log(p/\gamma^2)} \right\rceil = O(n/\log p) \; .$$

The running time is clear. We first show that $\lambda_1(\mathcal{L}^{(i+1)}) \leq \gamma \cdot \lambda_1(\mathcal{L}^{(i)})$ for all i. Indeed, let $\mathbf{v} \in \mathcal{L}^{(i)}$ such that $\|\mathbf{v}\| = \lambda_1(\mathcal{L}^{(i)})$. As in the previous proof, it suffices to observe that $\mathbf{v} \in \mathcal{L}_{i,c}$ for some c, as this will imply that

$$\lambda_1(\mathcal{L}^{(i+1)}) \leq \min_i d_{i,j} \leq d_{i,c} \leq \gamma\lambda_1(\mathcal{L}_{i,c}) = \gamma\|\mathbf{v}\| = \gamma\lambda_1(\mathcal{L}^{(i)}) \; ,$$

as needed. To see this, note that we can write $\mathbf{v} = \sum_{i=1}^{n-2} r_i\mathbf{b}_i + r_{n-1} \cdot (a_1^{(i)}\mathbf{b}_{n-1} + a_2^{(i)}\mathbf{b}_n) + r_n \cdot a_3^{(i)}\mathbf{b}_n$ where $r_i \in \mathbb{Z}$. If $r_{n-1} \equiv 0 \bmod p$, then clearly $\mathbf{v} \in \mathcal{L}_{i,p}$. Otherwise, there is a

[6] I.e., if $j < p$, the reduction sets $a_1^{(i+1)} := a_1^{(i)}$, $a_2^{(i+1)} := a_2^{(i)} + ja_3^{(i)}$, and $a_3^{(i+1)} := pa_3^{(i)}$. Otherwise, it sets $a_1^{(i+1)} := pa_1^{(i)}$, $a_2^{(i+1)} := a_2^{(i)}$, and $a_3^{(i+1)} := a_3^{(i)}$.

$c \in \{0, \ldots, p-1\}$ such that $cr_{n-1} \equiv r_n \bmod p$. Then,

$$\begin{aligned}\mathbf{v} &= \sum_{i=1}^{n-2} r_i\mathbf{b}_i + r_{n-1} \cdot (a_1^{(i)}\mathbf{b}_{n-1} + a_2^{(i)}\mathbf{b}_n) + r_n a_3^{(i)}\mathbf{b}_n \\ &= \sum_{i=1}^{n-2} r_i\mathbf{b}_i + r_{n-1} \cdot (a_1^{(i)}\mathbf{b}_{n-1} + a_2^{(i)}\mathbf{b}_n + ca_3^{(i)}\mathbf{b}_n) + (r_n - cr_{n-1})a_3^{(i)}\mathbf{b}_n \ .\end{aligned}$$

Note that by definition $r_n - cr_{n-1} \equiv 0 \bmod p$, so it follows that $\mathbf{v} \in \mathcal{L}_{i,c}$, as needed.

In particular, $\lambda_1(\mathcal{L}^{(\ell)}) \leq \gamma^\ell \lambda_1(\mathcal{L})$. Now, we claim that $\lambda_1(\mathcal{L}') \leq \lambda_1(\mathcal{L}^{(\ell)})$. Note that any point $\mathbf{y} = \sum a_i\mathbf{b}_i \in \mathcal{L}^{(\ell)} \setminus \mathcal{L}'$ has either $|a_n| \geq p^{\max(k_1,k_2)} \geq p^{\ell/2}$ or $|a_{n-1}| \geq p^{\ell/2}$ (depending on whether or not $k_1 \geq k_2$). But, by Lemma 10, this implies that $\|\mathbf{y}\| \geq 2^{-n/2-1} \cdot p^{\ell/2} \cdot \lambda_1(\mathcal{L}) > \gamma^\ell \lambda_1(\mathcal{L})$. Therefore, any vector in $\mathcal{L}^{(\ell)}$ of length at most $\lambda_1(\mathcal{L}^{(\ell)}) \leq \gamma^\ell \cdot \lambda_1(\mathcal{L})$ must be in $\mathcal{L}'$, as needed.

Finally, as in the previous proof, we can show by a simple induction argument that the output vector has length at most $\gamma^{\ell(n-1)}\lambda_1(\mathcal{L}') \leq \gamma^{\ell n}\lambda_1(\mathcal{L}) = \gamma^{O(n^2/\log p)} \cdot \lambda_1(\mathcal{L})$, as needed. ◀

Acknowledgments. I would like to thank Divesh Aggarwal, Huck Bennett, Zvika Brakerski, Daniel Dadush, Daniele Micciancio, and Oded Regev, all of whom provided helpful commentary on earlier drafts of this paper. In particular, Divesh noted that our SVP reduction shows hardness of unique SVP (Corollary 26), and Daniele noted that it implied a reduction from $\sqrt{n}$-CVP to γ-GapSVP (Corollary 21). I also thank the anonymous reviewers for their helpful comments.

References

1 Divesh Aggarwal, Daniel Dadush, Oded Regev, and Noah Stephens-Davidowitz. Solving the Shortest Vector Problem in 2^n time via discrete Gaussian sampling. In *STOC*, 2015.

2 Divesh Aggarwal and Chandan Dubey. Improved hardness results for unique Shortest Vector Problem, 2015. URL: `http://eccc.hpi-web.de/report/2013/076/`.

3 Dorit Aharonov and Oded Regev. Lattice problems in NP intersect coNP. *Journal of the ACM*, 52(5):749–765, 2005. Preliminary version in FOCS'04.

4 Miklós Ajtai. Generating hard instances of lattice problems. In *STOC*, pages 99–108. ACM, 1996.

5 Miklós Ajtai. The Shortest Vector Problem in L2 is NP-hard for randomized reductions. In *STOC*, 1998. `doi:10.1145/276698.276705`.

6 Miklós Ajtai, Ravi Kumar, and D. Sivakumar. A sieve algorithm for the shortest lattice vector problem. In *STOC*, pages 601–610, 2001.

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

8 Shi Bai, Weiqiang Wen, and Damien Stehlé. Improved reduction from the Bounded Distance Decoding problem to the unique Shortest Vector Problem in lattices. In *ICALP*, 2016.

9 W. Banaszczyk. New bounds in some transference theorems in the geometry of numbers. *Mathematische Annalen*, 296(4):625–635, 1993. `doi:10.1007/BF01445125`.

10 U. Betke, M. Henk, and J.M. Wills. Successive-minima-type inequalities. *Discrete & Computational Geometry*, 9(1):165–175, 1993. `doi:10.1007/BF02189316`.

11 Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In *FOCS*, pages 97–106. IEEE, 2011.

12 Zvika Brakerski and Vinod Vaikuntanathan. Lattice-based FHE as secure as PKE. In *ITCS*, pages 1–12, 2014.

13 Jin-Yi Cai and Ajay Nerurkar. Approximating the SVP to within a factor $(1+1/\dim^\varepsilon)$ is NP-hard under randomized reductions. *Journal of Computer and System Sciences*, 59(2):221–239, 1999. `doi:10.1006/jcss.1999.1649`.

14 Daniel Dadush and Gabor Kun. Lattice sparsification and the approximate Closest Vector Problem. In *SODA*, 2013.

15 Daniel Dadush, Chris Peikert, and Santosh Vempala. Enumerative lattice algorithms in any norm via M-ellipsoid coverings. In *FOCS*, pages 580–589. IEEE, 2011.

16 Daniel Dadush, Oded Regev, and Noah Stephens-Davidowitz. On the Closest Vector Problem with a distance guarantee. In *CCC*, pages 98–109, 2014. `doi:10.1109/CCC.2014.18`.

17 Craig Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, pages 169–178. ACM, New York, 2009.

18 Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC*, pages 197–206, 2008.

19 Oded Goldreich, Daniele Micciancio, Shmuel Safra, and Jean-Paul Seifert. Approximating shortest lattice vectors is not harder than approximating closest lattice vectors. *Information Processing Letters*, 71(2):55–61, 1999. `doi:10.1016/S0020-0190(99)00083-6`.

20 Ishay Haviv and Oded Regev. Tensor-based hardness of the Shortest Vector Problem to within almost polynomial factors. *Theory of Computing*, 8(23):513–531, 2012. Preliminary version in STOC'07.

21 Antoine Joux and Jacques Stern. Lattice reduction: A toolbox for the cryptanalyst. *Journal of Cryptology*, 11(3):161–185, 1998.

22 G. A. Kabatjanskiĭ and V. I. Levenšteĭn. Bounds for packings on the sphere and in space. *Problemy Peredači Informacii*, 14(1):3–25, 1978.

23 Ravi Kannan. Minkowski's convex body theorem and integer programming. *Mathematics of Operations Research*, 12(3):pp. 415–440, 1987. URL: `http://www.jstor.org/stable/3689974`.

24 Subhash Khot. Hardness of approximating the Shortest Vector Problem in lattices. *Journal of the ACM*, 52(5):789–808, September 2005. Preliminary version in FOCS'04.

25 S. Ravi Kumar and D. Sivakumar. On the unique shortest lattice vector problem. *Theoretical Computer Science*, 255(1,Äì2):641–648, 2001. `doi:10.1016/S0304-3975(00)00387-X`.

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

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

28 Yi-Kai Liu, Vadim Lyubashevsky, and Daniele Micciancio. On Bounded Distance Decoding for general lattices. In *RANDOM*, 2006.

29 Vadim Lyubashevsky and Daniele Micciancio. On Bounded Distance Decoding, unique shortest vectors, and the minimum distance problem. In *Advances in Cryptology – CRYPTO 2009*, volume 5677 of *LNCS*, pages 577–594. Springer, 2009. `doi:10.1007/978-3-642-03356-8_34`.

30 Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and Learning with Errors over rings. In *EUROCRYPT*, 2010.

31 Daniele Micciancio. The Shortest Vector Problem is NP-hard to approximate to within some constant. *SIAM Journal on Computing*, 30(6):2008–2035, March 2001. Preliminary version in FOCS 1998.

32 Daniele Micciancio. Efficient reductions among lattice problems. In *SODA*, pages 84–93. ACM, New York, 2008.

33 Daniele Micciancio and Shafi Goldwasser. *Complexity of Lattice Problems: a cryptographic perspective*, volume 671 of *The Kluwer International Series in Engineering and Computer Science*. Kluwer Academic Publishers, Boston, Massachusetts, March 2002.

34 Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on Gaussian measures. *SIAM Journal on Computing*, 37(1):267–302, 2007.

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

36 Andrew M Odlyzko. The rise and fall of knapsack cryptosystems. *Cryptology and computational number theory*, 42:75–88, 1990.

37 Chris Peikert. Public-key cryptosystems from the worst-case Shortest Vector Problem. In *STOC*, pages 333–342. ACM, 2009.

38 Chris Peikert and Alon Rosen. Lattices that admit logarithmic worst-case to average-case connection factors. In *STOC*, 2007.

39 Xavier Pujol and Damien Stehlé. Solving the shortest lattice vector problem in time $2^{2.465n}$. *IACR Cryptology ePrint Archive*, 2009:605, 2009.

40 Oded Regev. On lattices, Learning with Errors, random linear codes, and cryptography. *Journal of the ACM*, 56(6):Art. 34, 40, 2009. doi:10.1145/1568318.1568324.

41 Noah Stephens-Davidowitz. Dimension-preserving reductions between lattice problems, 2015. URL: http://noahsd.com/latticeproblems.pdf.

42 Noah Stephens-Davidowitz. Discrete Gaussian sampling reduces to CVP and SVP. In *SODA*, 2016.