Slide Reduction, Revisited—Filling the Gaps in SVP Approximation

Divesh Aggarwal[1], Jianwei Li[2], Phong Q. Nguyen[3,4], and Noah Stephens-Davidowitz[5⋆]

[1] National University of Singapore. dcsdiva@nus.edu.sg
[2] Information Security Group, Royal Holloway, University of London. lijianweithu@sina.com
[3] Inria Paris, France. pnguyen@inria.fr
[4] Département d'informatique de l'ENS, ENS, CNRS, PSL University, Paris, France.
[5] Cornell University. noahsd@gmail.com

Abstract. We show how to generalize Gama and Nguyen's slide reduction algorithm [STOC '08] for solving the approximate Shortest Vector Problem over lattices (SVP) to allow for arbitrary block sizes, rather than just block sizes that divide the rank n of the lattice. This leads to significantly better running times for most approximation factors. We accomplish this by combining slide reduction with the DBKZ algorithm of Micciancio and Walter [Eurocrypt '16].

We also show a different algorithm that works when the block size is quite large—at least half the total rank. This yields the first non-trivial algorithm for sublinear approximation factors.

Together with some additional optimizations, these results yield significantly faster provably correct algorithms for δ-approximate SVP for all approximation factors $n^{1/2+\varepsilon} \leq \delta \leq n^{O(1)}$, which is the regime most relevant for cryptography. For the specific values of $\delta = n^{1-\varepsilon}$ and $\delta = n^{2-\varepsilon}$, we improve the exponent in the running time by a factor of 2 and a factor of 1.5 respectively.

Keywords: Lattice Reduction · Slide Reduction · DBKZ · SVP.

1 Introduction

A lattice $\mathcal{L} \subset \mathbb{R}^m$ is the set of integer linear combinations

$$\mathcal{L} := \mathcal{L}(\mathbf{B}) = \{z_1\boldsymbol{b}_1 + \cdots + z_n\boldsymbol{b}_n \ : \ z_i \in \mathbb{Z}\}$$

⋆ The first author was partially funded by the Singapore Ministry of Education and the National Research Foundation under grant R-710-000-012-135, and supported by the grant MOE2019-T2-1-145 "Foundations of quantum-safe cryptography". The second author was funded by EPSRC grant EP/S020330/1. This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 885394). Parts of this work were done while the fourth author was visiting the Massachusetts Institute of Technology, the Centre for Quantum Technologies at the National University of Singapore, and the Simons Institute in Berkeley.

of linearly independent basis vectors $\mathbf{B} = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n) \in \mathbb{R}^{m \times n}$. We call n the *rank* of the lattice.

The Shortest Vector Problem (SVP) is the computational search problem in which the input is (a basis for) a lattice $\mathcal{L} \subseteq \mathbb{Z}^m$, and the goal is to output a non-zero lattice vector $\boldsymbol{y} \in \mathcal{L}$ with minimal length, $\|\boldsymbol{y}\| = \lambda_1(\mathcal{L}) := \min_{\boldsymbol{x} \in \mathcal{L}_{\neq \mathbf{0}}} \|\boldsymbol{x}\|$. For $\delta \geq 1$, the δ-approximate variant of SVP (δ-SVP) is the relaxation of this problem in which any non-zero lattice vector $\boldsymbol{y} \in \mathcal{L}_{\neq \mathbf{0}}$ with $\|\boldsymbol{y}\| \leq \delta \cdot \lambda_1(\mathcal{L})$ is a valid solution.

A closely related problem is δ-*Hermite SVP* (δ-HSVP, sometimes also called Minkowski SVP), which asks us to find a non-zero lattice vector $\boldsymbol{y} \in \mathcal{L}_{\neq \mathbf{0}}$ with $\|\boldsymbol{y}\| \leq \delta \cdot \mathrm{vol}(\mathcal{L})^{1/n}$, where $\mathrm{vol}(\mathcal{L}) := \det(\mathbf{B}^T\mathbf{B})^{1/2}$ is the covolume of the lattice. *Hermite's constant* γ_n is (the square of) the minimal possible approximation factor that can be achieved in the worst case. I.e.,

$$\gamma_n := \sup \frac{\lambda_1(\mathcal{L})^2}{\mathrm{vol}(\mathcal{L})^{2/n}} ,$$

where the supremum is over lattices $\mathcal{L} \subset \mathbb{R}^n$ with full rank n. Hermite's constant is only known exactly for $1 \leq n \leq 8$ and $n = 24$, but it is known to be asymptotically linear in n, i.e., $\gamma_n = \Theta(n)$. HSVP and Hermite's constant play a large role in algorithms for δ-SVP.

Starting with the celebrated work of Lenstra, Lenstra, and Lovász in 1982 [16], algorithms for solving δ-(H)SVP for a wide range of parameters δ have found innumerable applications, including factoring polynomials over the rationals [16], integer programming [17, 15, 7], cryptanalysis [37, 30, 14, 26], etc. More recently, many cryptographic primitives have been constructed whose security is based on the (worst-case) hardness of δ-SVP or closely related lattice problems [3, 34, 12, 32, 31]. Such lattice-based cryptographic constructions are likely to be used on massive scales (e.g., as part of the TLS protocol) in the not-too-distant future [29], and in practice, the security of these constructions depends on the fastest algorithms for δ-(H)SVP, typically for $\delta = \mathrm{poly}(n)$.

Work on δ-(H)SVP has followed two distinct tracks. There has been a long line of work showing progressively faster algorithms for exact SVP (i.e., $\delta = 1$) [15, 4, 28, 33, 23]. However, even the fastest such algorithm (with proven correctness) runs in time $2^{n+o(n)}$ [2, 1]. So, these algorithms are only useful for rather small n.

This paper is part of a separate line of work on *basis reduction algorithms* [16, 35, 36, 11, 9, 13, 24]. (See [27] and [24] for a much more complete list of works on basis reduction.) At a high level, these are reductions from δ-(H)SVP on lattices with rank n to exact SVP on lattices with rank $k \leq n$. More specifically, these algorithms divide a basis $\mathbf{B}$ into projected blocks $\mathbf{B}_{[i,i+k-1]}$ with *block size* k, where

$$\mathbf{B}_{[i,j]} = (\pi_i(\mathbf{b}_i), \pi_i(\mathbf{b}_{i+1}), \ldots, \pi_i(\mathbf{b}_j))$$

and π_i is the orthogonal projection onto the subspace orthogonal to $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_{i-1}$. Basis reduction algorithms use their SVP oracle to find short vectors in these (low-rank) blocks and incorporate these short vectors into the lattice basis $\mathbf{B}$.

By doing this repeatedly (at most $\mathrm{poly}(n, \log \|\mathbf{B}\|)$ times) with a cleverly chosen sequence of blocks, such algorithms progressively improve the "quality" of the basis $\mathbf{B}$ until $\boldsymbol{b}_1$ is a solution to δ-(H)SVP for some $\delta \geq 1$. The goal, of course, is to take the block size k to be small enough that we can actually run an exact algorithm on lattices with rank k in reasonable time while still achieving a relatively good approximation factor δ.

For HSVP, the DBKZ algorithm due to Micciancio and Walter yields the best proven approximation factor for all ranks n and block sizes k [24], which was previously obtained by [9] only when n is divisible by k. Specifically, the approximation factor corresponds to Mordell's inequality:

$$\delta_{\mathsf{MW},H} := \gamma_k^{\frac{n-1}{2(k-1)}} \ . \tag{1}$$

(Recall that $\gamma_k = \Theta(k)$ is Hermite's constant. Here and throughout the introduction, we have left out low-order factors that can be made arbitrarily close to one.) Using a result due to Lovász [22], this can be converted into an algorithm for $\delta_{\mathsf{MW},H}^2$-SVP. However, the slide reduction algorithm of Gama and Nguyen [9] achieves a better approximation factor for SVP. It yields

$$\delta_{\mathsf{GN},H} := \gamma_k^{\frac{\lceil n \rceil_k - 1}{2(k-1)}} \qquad \delta_{\mathsf{GN},S} := \gamma_k^{\frac{\lceil n \rceil_k - k}{k-1}} \ , \tag{2}$$

for HSVP and SVP respectively, where we write $\lceil n \rceil_k := k \cdot \lceil n/k \rceil$ for n rounded up to the nearest multiple of k. (We have included the result for HSVP in Eq. (2) for completeness, though it is clearly no better than Eq. (1).)

The discontinuous approximation factor in Eq. (2) is the result of an unfortunate limitation of slide reduction: it only works when the block size k divides the rank n. If n is not divisible by k, then we must artificially pad our basis so that it has rank $\lceil n \rceil_k$, which results in the rather odd expressions in Eq. (2). Of course, for $n \gg k$, this rounding has little effect on the approximation factor. But, for cryptographic applications, we are interested in small polynomial approximation factors $\delta \approx n^c$ for relatively small constants c, i.e., in the case when $k = \Theta(n)$. For such values of k and n, this rounding operation can cost us a constant factor in the exponent of the approximation factor, essentially changing n^c to $n^{\lceil c \rceil}$. Such constants in the exponent have a large effect on the theoretical security of lattice-based cryptography.[6]

1.1 Our results

Our first main contribution is a generalization of Gama and Nguyen's slide reduction [9] without the limitation that the rank n must be a multiple of the block size k. Indeed, we achieve exactly the approximation factor shown in Eq. (2) without any rounding, as we show below.

[6] The concrete security of lattice-based cryptography is assessed using HSVP and a heuristic version of Eq. (2) where Hermite's constant is replaced by a Gaussian heuristic estimate. In this work, we restrict our attention to what we can prove, and we focus on SVP rather than HSVP.

As a very small additional contribution, we allow for the possibility that the underlying SVP algorithm for lattices with rank k only solves δ-approximate SVP for some $\delta > 1$. This technique was already known to folklore and used in practice, and the proof requires no new ideas. Nevertheless, we believe that this work is the first to formally show that a δ-SVP algorithm suffices and to compute the exact dependence on δ. (This minor change proves quite useful when we instantiate our δ-SVP subroutine with the $2^{0.802k}$-time δ-SVP algorithm for some large constant $\delta \gg 1$ due to Liu, Wang, Xu, and Zheng [21, 38]. See Table 1 and Figure 1.)

Theorem 1 (Informal, slide reduction for $n \geq 2k$). *For any approximation factor $\delta \geq 1$ and block size $k := k(n) \geq 2$, there is an efficient reduction from δ_H-HSVP and δ_S-SVP on lattices with rank $n \geq 2k$ to δ-SVP on lattices with rank k, where*

$$\delta_H := (\delta^2\gamma_k)^{\frac{n-1}{2(k-1)}} \qquad \delta_S := \delta(\delta^2\gamma_k)^{\frac{n-k}{k-1}} .$$

Notice in particular that this matches Eq. (2) in the case when $\delta = 1$ and k divides n. (This is not surprising, since our algorithm is essentially identical to the original algorithm from [9] in this case.) Theorem 1 also matches the approximation factor for HSVP achieved by [24], as shown in Eq. (1), so that the best (proven) approximation factor for both problems is now achieved by a single algorithm: in other words, we get the best of both algorithms [9] and [24].

However, Theorem 1 only applies for $n \geq 2k$. Our second main contribution is an algorithm that works for $k \leq n \leq 2k$. To our knowledge, this is the first algorithm that provably achieves sublinear approximation factors for SVP and is asymptotically faster than, say, the fastest algorithm for $O(1)$-SVP. (We overcame a small barrier here. See the discussion in Section 3.)

Theorem 2 (Informal, slide reduction for $n \leq 2k$). *For any approximation factor $\delta \geq 1$ and block size $k \in [n/2, n]$, there is an efficient reduction from δ_S-SVP on lattices with rank n to δ-SVP on lattices with rank k, where*

$$\delta_S := \delta^2\sqrt{\gamma_k}(\delta^2\gamma_q)^{\frac{q+1}{q-1}\cdot\frac{n-k}{2k}} \lesssim \delta(\delta^2\gamma_k)^{\frac{n}{2k}} ,$$

and $q := n - k \leq k$.

Together, these algorithms yield the asymptotically fastest proven running times for δ-SVP for all approximation factors $n^{1/2+\varepsilon} \leq \delta \leq n^{O(1)}$—with a particularly large improvement when $\delta = n^c$ for $1/2 < c < 1$ or for any c slightly smaller than an integer. Table 1 and Figure 1 summarize the current state of the art. For example, one can solve $O(n^{1.99})$-SVP in $2^{0.269n+o(n)}$-time and $O(n^{0.99})$-SVP in $2^{0.405n+o(n)}$ instead of the previously best $2^{0.401n+o(n)}$-time and $2^{0.802n+o(n)}$, respectively.

It is worthwhile to mention that, though our focus is on provable algorithms, any heuristic algorithm can be plugged into our reduction giving us the same improvement for these algorithms (see Table 2). Our reduction just shows how to "recycle" one's favourite algorithm for exact (or near-exact) SVP to tackle

Approximation factor	Previous best	Folklore	This work
Exact	2^n [2]	—	—
$\Omega(1) \leq \delta \leq \sqrt{n}$	$2^{0.802n}$ [38]	—	—
n^c for $c \in [\frac{1}{2}, 1)$	$2^{0.802n}$ [38]	—	$2^{\frac{0.802n}{2c}}$ [*]+[38]
n^c for $c \geq 1$	$2^{\frac{n}{\lfloor c+1 \rfloor}}$ [9]+[2]	$2^{\frac{0.802n}{\lfloor c+1 \rfloor}}$ [9]+[38]	$2^{\frac{0.802n}{c+1}}$ [*]+[38]

Table 1: Provable algorithms for solving SVP. We write [A]+[B] to denote the algorithm that uses basis reduction from [A] with the exact/near-exact SVP algorithm from [B], and we write [*] for this work. The "folklore" column represents a result that was likely known to many experts in the field but apparently never published.

Approximation factor	Previous best	This work
$1 \leq \delta \leq \sqrt{n}$	$2^{0.292n}$ [5]	—
n^c for $c \in [\frac{1}{2}, 1)$	$2^{0.292n}$ [5]	$2^{\frac{0.292n}{2c}}$ [*]+[5]
n^c for $c \geq 1$	$2^{\frac{0.292n}{\lfloor c+1 \rfloor}}$ [9]+[5]	$2^{\frac{0.292n}{c+1}}$ [*]+[5]

Table 2: Heuristic algorithms for solving SVP.

higher dimension, provided that one is interested in approximating SVP rather than HSVP. Our results further our understanding of the hardness of SVP but they do not impact usual security estimates, such as those of lattice-based candidates to NIST's post-quantum standardization: this is because current security estimates actually rely on HSVP estimates, following [10]. The problem of approximating SVP is essentially the same as that of approximating HSVP, except for lattices with an extremely small first minimum: such lattices exist but typically do not arise in real-world cryptographic constructions (see [10, §3.2]). For the same reason, implementing our algorithm has limited value in practice at the moment: running the algorithm would only be meaningful if one was interested in approximating SVP on ad-hoc lattices with an extremely small first minimum.

1.2 Our techniques

We first briefly recall some of the details of Gama and Nguyen's slide reduction. Slide reduction divides the basis $\mathbf{B} = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n) \in \mathbb{R}^{m \times n}$ evenly into disjoint "primal blocks" $\mathbf{B}_{[ik+1,(i+1)k]}$ of length k. (Notice that this already requires n to be divisible by k.) It also defines certain "dual blocks" $\mathbf{B}_{[ik+2,(i+1)k+1]}$, which are the primal blocks shifted one to the right. The algorithm then tries to simultaneously satisfy certain primal and dual conditions on these blocks. Namely, it tries to *SVP-reduce* each primal block—i.e., it tries to make the first vector in the block $\boldsymbol{b}^*_{ik+1}$ a shortest vector in $\mathcal{L}(\mathbf{B}_{[ik+1,(i+1)k]})$, where $\boldsymbol{b}^*_j := \pi_j(\boldsymbol{b}_j)$. Simultaneously, it tries to *dual SVP-reduce* (DSVP-reduce) the dual blocks. (See Section 2.3 for the definition of DSVP reduction.) We call a basis that satisfies all of these conditions simultaneously *slide-reduced*.

An SVP oracle for lattices with rank k is sufficient to enforce all primal conditions or all dual conditions separately. (E.g., we can enforce the primal

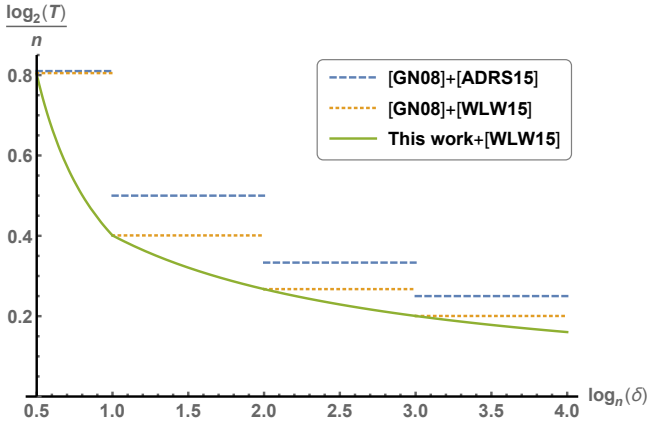

Fig. 1: Provable running time T as a function of approximation factor δ for δ-SVP. The y-axis is $\log_2(T)/n$, and the x-axis is $\log_n \delta$.

conditions by simply finding a shortest non-zero vector in each primal block and including this vector in an updated basis for the block.) Furthermore, if all primal and dual conditions hold simultaneously, then $\|\boldsymbol{b}_1\| \leq \delta_{\mathsf{GN},S}\lambda_1(\mathcal{L})$ with $\delta_{\mathsf{GN},S}$ as in Eq. (2), so that $\|\boldsymbol{b}_1\|$ yields a solution to $\delta_{\mathsf{GN},S}$-SVP. This follows from repeated application of a "gluing" lemma on such bases, which shows how to "glue together" two reduced block to obtain a larger reduced block. (See Lemma 1.) Finally, Gama and Nguyen showed that, if we alternate between SVP-reducing the primal blocks and DSVP-reducing the dual blocks, then the basis will converge quite rapidly to a slide-reduced basis (up to some small slack) [9]. Combining all of these facts together yields the main result in [9]. (See Section 4.)

The case $n > 2k$. We wish to extend slide reduction to the case when $n = pk + q$ for $1 \leq q < k$. So, intuitively, we have to decide what to do with "the extra q vectors in the basis." To answer this, we exploit a "gluing" property, which is implicit in LLL and slide reduction, but which we make explicit: given an integer $\ell \in \{1, \ldots, n\}$, any basis B of a lattice L defines two blocks $B_1 = B_{[1,\ell]}$ and $B_2 = B_{[\ell+1,n]}$. The first block B_1 is a basis of a (primitive) sublattice L_1 of L, and the second block B_2 is a basis of another lattice L_2 which can be thought as the quotient L/L_1. Intuitively, the basis B glues the two blocks B_1 and B_2 together: a gluing property (Lemma 1) provides sufficient conditions on the two blocks B_1 and B_2 to guarantee that the basis B is (H)SVP-reduced. Crucially, the gluing property shows that there is an asymmetry between B_1 and B_2: B can be SVP-reduced without requiring both B_1 and B_2 to be SVP-reduced. Namely, it suffices that B_1 is HSVP-reduced, B_2 is SVP-reduced together with a gluing condition relating the first vectors of B_1 and B_2. [7]

[7] We are ignoring a certain degenerate case here for simplicity. Namely, if all short vectors happen to lie in the span of the first block, and these vectors happen to be very short relative to the volume of the first block, then calling an HSVP oracle on

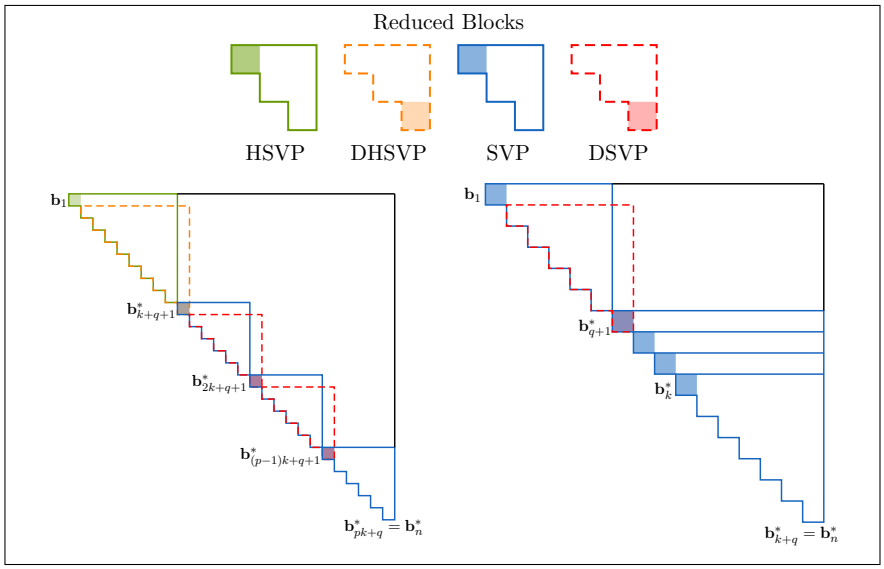

Fig. 2: Slide reduction of an upper-triangular matrix for $n = pk + q \geq 2k$ (left) and $n = k + q \leq 2k$ (right). (The original notion of slide reduction in [9] used only SVP-reduced and DSVP-reduced blocks of fixed size k.)

The HSVP reduction of B_1 can be handled by the algorithm from [24], irrespective of the rank of B_1. The SVP reduction of B_2 can be handled by our SVP oracle if the rank of B_2 is chosen to be k, or by slide reduction [9] if the rank of B_2 is chosen to be a multiple of k. Finally, the gluing condition can be achieved by duality, by reusing the main idea of [9]. Thus, "the extra q vectors in the basis" can simply be included in the first block B_1.

Interestingly, the HSVP approximation factor achieved by [24] (which we use for B_1) and the SVP approximation factor achieved by [9] (which we can use for B_2) are exactly what we need to apply our gluing lemma. (This is not a coincidence, as we explain in Section 4.) The result is Theorem 1.

The case $n < 2k$. For $n = k+q < 2k$, the above idea cannot work. In particular, a "big block" of size $k + q$ in this case would be our entire basis! So, instead of working with one big block and some "regular blocks" of size k, we work with a "small block" of size q and one regular block of size k. We then simply perform slide reduction with (primal) blocks $\mathbf{B}_{[1,q]}$ and $\mathbf{B}_{[q+1,n]} = \mathbf{B}_{[n-k+1,n]}$. If we were to stop here, we would achieve an approximation factor of roughly γ_q (see [20, Th. 4.3.1]), which for $q = \Theta(k)$ is essentially the same as the approximation

the first block might not be sufficient to solve approximate SVP. Of course, if we know a low-dimensional subspace that contains the shortest non-zero vector, then finding short lattice vectors is much easier. This degenerate case is therefore easily handled separately (but it does in fact need to be handled separately).

factor of roughly γ_k that we get when the rank is $2k$. I.e., we would essentially "pay for two blocks of length k," even though one block has size $q < k$.

However, we notice that a slide-reduced basis guarantees more than just a short first vector. It also promises a very strong bound on $\mathrm{vol}(\mathbf{B}_{[1,q]})$. In particular, since $q < k$ and since we have access to an oracle for lattices with rank k, it is natural to try to extend this small block $\mathbf{B}_{[1,q]}$ with low volume to a larger block $\mathbf{B}_{[1,k]}$ of length k that still has low volume. Indeed, we can use our SVP oracle to guarantee that $\mathbf{B}_{[q+1,k]}$ consists of relatively short vectors so that $\mathrm{vol}(\mathbf{B}_{[q+1,k]})$ is relatively small as well. (Formally, we SVP-reduce $\mathbf{B}_{[i,n]}$ for $i \in [q+1,k]$. Again, we are ignoring a certain degenerate case, as in Footnote 7.) This allows us to upper bound $\mathrm{vol}(\mathbf{B}_{[1,k]}) = \mathrm{vol}(\mathbf{B}_{[1,q]}) \cdot \mathrm{vol}(\mathbf{B}_{[q+1,k]})$, which implies that $\lambda_1(\mathcal{L}(\mathbf{B}_{[1,k]}))$ is relatively short. We can therefore find a short vector by making an additional SVP oracle call on $\mathcal{L}(\mathbf{B}_{[1,k]})$. (Micciancio and Walter used a similar idea in [24].)

1.3 Open questions and directions for future work

Table 1 suggests an obvious open question: can we find a non-trivial basis reduction algorithm that provably solves δ-SVP for $\delta \leq O(\sqrt{n})$? More formally, can we reduce $O(\sqrt{n})$-SVP on lattices with rank n to exact SVP on lattices with rank $k = cn$ for some constant $c < 1$. Our current proof techniques seem to run into a fundamental barrier here in that they seem more-or-less incapable of achieving $\delta \ll \sqrt{\gamma_k}$. This setting is interesting in practice, as many record lattice computations use block reduction with $k \geq n/2$ as a subroutine, such as [6]. (One can provably achieve approximation factors $\delta \ll \sqrt{\gamma_k}$ when $k = (1-o(1))n$ with a bit of work,[8] but it is not clear if these extreme parameters are useful.)

Next, we recall that this work shows how to exploit the existing very impressive algorithms for HSVP (in particular, DBKZ [24]) to obtain better algorithms for SVP. This suggests two closely related questions for future work: (1) can we find better algorithms for HSVP (e.g., for δ-HSVP with $\delta \approx \sqrt{\gamma_n}$—i.e., "near-exact" HSVP); and (2) where else can we profitably replace SVP oracles with HSVP oracles? Indeed, most of our analysis (and the analysis of other basis reduction algorithms) treats the δ-SVP oracle as a $\delta\sqrt{\gamma_k}$-HSVP oracle. We identified one way to exploit this to actually get a faster algorithm, but perhaps more can be done here—particularly if we find faster algorithms for HSVP.

Finally, we note that we present two distinct (though similar) algorithms: one for lattices with rank $n \leq 2k$ and one for lattices with rank $n \geq 2k$. It is natural to ask whether there is a single algorithm that works in both regimes. Perhaps work on this question could even lead to better approximation factors.

[8] For example, it is immediate from the proof of Theorem 5 that the (very simple) notion of a slide-reduced basis for $n \leq 2k$ in Definition 1 is already enough to obtain $\delta \approx \gamma_{n-k} \approx n-k$. So, for $n \lesssim k+\sqrt{k}$, this already achieves $\delta \lesssim \sqrt{n}$. With a bit more work, one can show that an extra oracle call like the one used in Corollary 1 can yield a still better approximation factor in this rather extreme setting of $k = (1-o(1))n$.

2 Preliminaries

We denote column vectors $\boldsymbol{x} \in \mathbb{R}^m$ by bold lower-case letters. Matrices $\mathbf{B} \in \mathbb{R}^{m \times n}$ are denoted by bold upper-case letters, and we often think of a matrix as a list of column vectors, $\mathbf{B} = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n)$. For a matrix $\boldsymbol{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ with n linearly independent columns, we write $\mathcal{L}(\boldsymbol{B}) := \{z_1\boldsymbol{b}_1 + \cdots + z_n\boldsymbol{b}_n \ : \ z_i \in \mathbb{Z}\}$ for the lattice generated by $\mathbf{B}$ and $\|\boldsymbol{B}\| = \max\{\|\mathbf{b}_1\|, \ldots, \|\mathbf{b}_n\|\}$ for the maximum norm of a column. We often implicitly assume that $m \geq n$ and that a basis matrix $\mathbf{B} \in \mathbb{R}^{m \times n}$ has rank n (i.e., that the columns of $\mathbf{B}$ are linearly independent). We use the notation $\log := \log_2$ to mean the logarithm with base two.

2.1 Lattices

For any lattice $\mathcal{L}$, its *dual lattice* is

$$\mathcal{L}^\times = \{\mathbf{w} \in \mathrm{span}(\mathcal{L}) : \ \langle \mathbf{w}, \mathbf{y} \rangle \in \mathbb{Z} \text{ for all } \boldsymbol{y} \in \mathcal{L}\} \ .$$

If $\mathbf{B} \in \mathbb{R}^{m \times n}$ is a basis of $\mathcal{L}$, then $\mathcal{L}^\times$ has basis $\mathbf{B}^\times := \mathbf{B}(\mathbf{B}^T\mathbf{B})^{-1}$, called the *dual basis* of $\mathbf{B}$. The *reversed dual basis* $\mathbf{B}^{-s}$ of $\mathbf{B}$ is simply $\mathbf{B}^\times$ with its columns in reversed order [8].

2.2 Gram-Schmidt orthogonalization

For a basis $\mathbf{B} = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n) \in \mathbb{R}^{m \times n}$, we associate a sequence of projections $\pi_i := \pi_{\{\boldsymbol{b}_1, \ldots, \boldsymbol{b}_{i-1}\}^\perp}$. Here, $\pi_{W^\perp}$ means the orthogonal projection onto the subspace $W^\perp$ orthogonal to W. As in [9], $\mathbf{B}_{[i,j]}$ denotes the projected block $(\pi_i(\boldsymbol{b}_i), \pi_i(\boldsymbol{b}_{i+1}), \ldots, \pi_i(\boldsymbol{b}_j))$.

We also associate to $\mathbf{B}$ its Gram-Schmidt orthogonalization (GSO) $\boldsymbol{B}^* := (\mathbf{b}_1^*, \ldots, \mathbf{b}_n^*)$, where $\boldsymbol{b}_i^* := \pi_i(\boldsymbol{b}_i) = \boldsymbol{b}_i - \sum_{j<i} \mu_{i,j}\boldsymbol{b}_j^*$, and $\mu_{i,j} = \langle \boldsymbol{b}_i, \boldsymbol{b}_j^* \rangle / \|\boldsymbol{b}_j^*\|^2$.

We say that $\mathbf{B}$ is *size-reduced* if $|\mu_{i,j}| \leq \frac{1}{2}$ for all $i \neq j$: then $\|\mathbf{B}\| \leq \sqrt{n}\|\mathbf{B}^*\|$. Transforming a basis into this form without modifying $\mathcal{L}(\mathbf{B})$ or $\mathbf{B}^*$ is called *size reduction*, and this can be done easily and efficiently.

2.3 Lattice basis reduction

LLL reduction. Let $\boldsymbol{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ be a size-reduced basis. For $\varepsilon \in [0, 1]$, we say that $\boldsymbol{B}$ is *ε-LLL-reduced* [16] if every rank-two projected block $\boldsymbol{B}_{[i,i+1]}$ satisfies Lovász's condition: $\|\mathbf{b}_i^*\|^2 \leq (1+\varepsilon)\|\mu_{i,i-1}\mathbf{b}_{i-1}^* + \mathbf{b}_i^*\|^2$ for $1 < i \leq n$. For $\varepsilon \geq 1/\mathrm{poly}(n)$, one can efficiently compute an ε-LLL-reduced basis for a given lattice.

SVP reduction and its extensions. Let $\boldsymbol{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ be a basis of a lattice $\mathcal{L}$ and $\delta \geq 1$ be an approximation factor.

We say that $\boldsymbol{B}$ is *δ-SVP-reduced* if $\|\mathbf{b}_1\| \leq \delta \cdot \lambda_1(\mathcal{L})$. Similarly, we say that $\boldsymbol{B}$ is *δ-HSVP-reduced* if $\|\boldsymbol{b}_1\| \leq \delta \cdot \mathrm{vol}(\mathcal{L})^{1/n}$.

$\boldsymbol{B}$ is δ-*DSVP-reduced* [9] (where D stands for dual) if the reversed dual basis $\boldsymbol{B}^{-s}$ is δ-SVP-reduced and $\mathbf{B}$ is $\frac{1}{3}$-LLL-reduced. Similarly, we say that $\boldsymbol{B}$ is δ-*DHSVP-reduced* if $\boldsymbol{B}^{-s}$ is δ-HSVP-reduced.

The existence of such δ-DSVP-reduced bases is guaranteed by a classical property of LLL that $\|\boldsymbol{b}_n^*\|$ never decreases during the LLL-reduction process [16].

We can efficiently compute a δ-(D)SVP-reduced basis for a given rank n lattice $\mathcal{L} \subseteq \mathbb{Z}^m$ with access to an oracle for δ-SVP on lattices with rank at most n. Furthermore, given a basis $\mathbf{B} = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n) \in \mathbb{Z}^{m \times n}$ of $\mathcal{L}$ and an index $i \in [1, n-k+1]$, we can use a δ-SVP oracle for lattices with rank at most k to efficiently compute a size-reduced basis $\mathbf{C} = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_{i-1}, \boldsymbol{c}_i, \ldots, \boldsymbol{c}_{i+k-1}, \boldsymbol{b}_{i+k}, \ldots, \boldsymbol{b}_n)$ of $\mathcal{L}$ such that the block $\mathbf{C}_{[i,i+k-1]}$ is δ-SVP reduced or δ-DSVP reduced:

- If $\mathbf{C}_{[i,i+k-1]}$ is δ-SVP-reduced, the procedures in [9, 24, 19] equipped with δ-SVP-oracle ensure that $\|\mathbf{C}^*\| \leq \|\mathbf{B}^*\|$;
- If $\mathbf{C}_{[i,i+k-1]}$ is δ-DSVP-reduced, the inherent LLL reduction implies $\|\mathbf{C}^*\| \leq 2^k \|\mathbf{B}^*\|$. Indeed, the GSO of $\mathbf{C}_{[i,i+k-1]}$ satisfies

$$\|(\mathbf{C}_{[i,i+k-1]})^*\| \leq 2^{k/2} \lambda_k(\mathcal{L}(\mathbf{C}_{[i,i+k-1]}))$$

(by [16, p. 518, Line 27]) and $\lambda_k(\mathcal{L}(\mathbf{C}_{[i,i+k-1]})) \leq \sqrt{k}\|\mathbf{B}^*\|$. Here, $\lambda_k(\cdot)$ denotes the k-th minimum.

With size-reduction, we can iteratively perform $\mathrm{poly}(n, \log \|\mathbf{B}\|)$ many such operations efficiently. In particular, doing so will not increase $\|\mathbf{B}^*\|$ by more than a factor of $2^{\mathrm{poly}(n, \log \|\mathbf{B}\|)}$, and therefore the same is true of $\|\mathbf{B}\|$. That is, all intermediate entries and the total cost during execution (excluding oracle queries) remain polynomially bounded in the initial input size; See, e.g., [9, 18] for the evidence. Therefore, to bound the running time of basis reduction, it suffices to bound the number of calls to these block reduction subprocedures.

Twin reduction and gluing. We define the following notion, which was implicit in [9] and will arise repeatedly in our proofs. $\boldsymbol{B} = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_{d+1})$ is δ-*twin-reduced* if $\mathbf{B}_{[1,d]}$ is δ-*HSVP-reduced* and $\mathbf{B}_{[2,d+1]}$ is δ-*DHSVP-reduced*. The usefulness of twin reduction is illustrated by the following fact, which is the key idea behind Gama and Nguyen's slide reduction (and is remarkably simple in hindsight).

Fact 3. *If* $\mathbf{B} := (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_{d+1}) \in \mathbb{R}^{m \times (d+1)}$ *is* δ*-twin-reduced, then*

$$\|\boldsymbol{b}_1\| \leq \delta^{2d/(d-1)} \|\boldsymbol{b}_{d+1}^*\| \,. \tag{3}$$

Furthermore,

$$\delta^{-d/(d-1)} \|\boldsymbol{b}_1\| \leq \mathrm{vol}(\mathbf{B})^{1/(d+1)} \leq \delta^{d/(d-1)} \|\boldsymbol{b}_{d+1}^*\| \,. \tag{4}$$

Proof. By definition, we have $\|\boldsymbol{b}_1\|^d \leq \delta^d \mathrm{vol}(\mathbf{B}_{[1,d]})$, which is equivalent to

$$\|\boldsymbol{b}_1\|^{d-1} \leq \delta^d \mathrm{vol}(\mathbf{B}_{[2,d]}) \,.$$

Similarly,

$$\mathrm{vol}(\mathbf{B}_{[2,d]}) \leq \delta^d \|\boldsymbol{b}^*_{d+1}\|^{d-1} .$$

Combining these two inequalities yields Eq. (3).

Finally, we have $\|\boldsymbol{b}_1\|^d \|\boldsymbol{b}^*_{d+1}\| \leq \delta^d \mathrm{vol}(\mathbf{B})$. Applying Eq. (3) implies the first inequality in Eq. (4), and similar analysis yields the second inequality. □

The following gluing lemma, which is more-or-less implicit in prior work, shows conditions on the blocks $\mathbf{B}_{[1,d]}$ and $\mathbf{B}_{[d+1,n]}$ that are sufficient to imply (H)SVP reduction of the full basis $\mathbf{B}$. Notice in particular that the decay of the Gram-Schmidt vectors guaranteed by Eq. (3) is what is needed for Item 2 of the lemma below, when $\eta = \delta^{1/(d-1)}$. And, with this same choice of η, the HSVP reduction requirement on $\mathbf{B}_{[1,d]}$ in Fact 3 is the same as the one in Item 2 of Lemma 1.

Lemma 1 (The gluing lemma). *Let* $\mathbf{B} := (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n) \in \mathbb{R}^{m \times n}$, $\alpha, \beta, \eta \geq 1$, *and* $1 \leq d \leq n$.

1. *If* $\mathbf{B}_{[d+1,n]}$ *is* β*-SVP-reduced,* $\|\boldsymbol{b}_1\| \leq \alpha \|\boldsymbol{b}^*_{d+1}\|$, *and* $\lambda_1(\mathcal{L}(\mathbf{B})) < \lambda_1(\mathcal{L}(\mathbf{B}_{[1,d]}))$, *then* $\mathbf{B}$ *is* $\alpha\beta$*-SVP-reduced.*
2. *If* $\mathbf{B}_{[1,d]}$ *is* η^{d-1}*-HSVP-reduced,* $\mathbf{B}_{[d+1,n]}$ *is* η^{n-d-1}*-HSVP-reduced, and* $\|\boldsymbol{b}_1\| \leq \eta^{2d} \|\boldsymbol{b}^*_{d+1}\|$, *then* $\mathbf{B}$ *is* η^{n-1}*-HSVP-reduced.*

Proof. For Item 1, since $\lambda_1(\mathcal{L}(\mathbf{B})) < \lambda_1(\mathcal{L}(\mathbf{B}_{[1,d]}))$, there exists a shortest non-zero vector $\boldsymbol{u} \in \mathcal{L}(\mathbf{B})$ with $\|\boldsymbol{u}\| = \lambda_1(\mathcal{L}(\mathbf{B}))$ and $\pi_d(\boldsymbol{u}) \neq 0$. Since $\mathbf{B}_{[d+1,n]}$ is β-SVP-reduced, it follows that $\|\boldsymbol{b}^*_{d+1}\|/\beta \leq \|\pi_d(\boldsymbol{u})\| \leq \|\boldsymbol{u}\| = \lambda_1(\mathcal{L}(\mathbf{B}))$. Finally, we have $\|\boldsymbol{b}_1\| \leq \alpha \|\boldsymbol{b}^*_{d+1}\| \leq \alpha\beta\lambda_1(\mathcal{L})$ as needed.

Turning to Item 2, we note that the HSVP conditions imply that $\|\boldsymbol{b}_1\|^d \leq \eta^{d(d-1)} \mathrm{vol}(\mathbf{B}_{[1,d]})$ and $\|\boldsymbol{b}^*_{d+1}\|^{n-d} \leq \eta^{(n-d)(n-d-1)} \mathrm{vol}(\mathbf{B}_{[d+1,n]})$. Using the bound on $\|\boldsymbol{b}_1\|$ relative to $\|\boldsymbol{b}^*_{d+1}\|$, we have

$$\begin{aligned}
\|\boldsymbol{b}_1\|^n &\leq \eta^{2d(n-d)} \|\boldsymbol{b}_1\|^d \cdot \|\boldsymbol{b}^*_{d+1}\|^{n-d} \\
&\leq \eta^{2(n-d)d + d(d-1) + (n-d)(n-d-1)} \mathrm{vol}(\mathbf{B}) \\
&= \eta^{n(n-1)} \mathrm{vol}(\mathbf{B}) ,
\end{aligned}$$

as needed. □

2.4 The Micciancio-Walter DBKZ algorithm

We recall Micciancio and Walter's elegant DBKZ algorithm [24], as we will need it later. Formally, we slightly generalize DBKZ by allowing for the use of a δ-SVP-oracle. We provide only a high-level sketch of the proof of correctness, as the full proof is the same as the proof in [24], with Hermite's constant γ_k replaced by $\delta^2 \gamma_k$.

Theorem 4. *For integers* $n > k \geq 2$, *an approximation factor* $1 \leq \delta \leq 2^k$, *an input basis* $\mathbf{B}_0 \in \mathbb{Z}^{m \times n}$ *for a lattice* $\mathcal{L} \subseteq \mathbb{Z}^m$, *and* $N := \lceil (2n^2/(k-1)^2) \cdot$

Algorithm 1 The Micciancio-Walter DBKZ algorithm [24, Algorithm 1]

Input: A block size $k \geq 2$, number of tours N, a basis $\mathbf{B} = (\mathbf{b}_1, \cdots, \mathbf{b}_n) \in \mathbb{Z}^{m \times n}$, and access to a δ-SVP oracle for lattices with rank k.
Output: A new basis of $\mathcal{L}(\mathbf{B})$.

for $\ell = 1$ **to** N **do**
  for $i = 1$ **to** $n - k$ **do**
    δ-SVP-reduce $\mathbf{B}_{[i,i+k-1]}$.
  end for
  for $j = n - k + 1$ **to** 1 **do**
    δ-DSVP-reduce $\mathbf{B}_{[j,j+k-1]}$
  end for
end for
δ-SVP-reduce $\mathbf{B}_{[1,k]}$.
return $\mathbf{B}$.

$\log(n \log(5\|\mathbf{B}_0\|)/\varepsilon)\rceil$ *for some* $\varepsilon \in [2^{-\mathrm{poly}(n)}, 1]$, *Algorithm 1 outputs a basis* $\mathbf{B}$ *of* $\mathcal{L}$ *in polynomial time (excluding oracle queries) such that*

$$\|\boldsymbol{b}_1\| \leq (1+\varepsilon) \cdot (\delta^2 \gamma_k)^{\frac{n-1}{2(k-1)}} \mathrm{vol}(\mathcal{L})^{1/n}$$

by making $N \cdot (2n - 2k + 1) + 1$ *calls to the* δ*-SVP oracle for lattices with rank* k.

Proof (Proof sketch). We briefly sketch a proof of the theorem, but we outsource the most technical step to a claim from [24], which was originally proven in [25]. Let $\mathbf{B}^{(\ell)}$ be the basis immediately after the ℓth tour, and let $x_i^{(\ell)} := \log \mathrm{vol}(\mathbf{B}^{(\ell)}_{[1,k+i-1]}) - \frac{k+i-1}{n} \log \mathrm{vol}(\mathcal{L})$ for $i = 1, \ldots, n-k$. Let

$$y_i := \frac{(n-k-i+1)(k+i-1)}{k-1} \cdot \log(\delta\sqrt{\gamma_k}) \quad \text{for} \quad i = 1, \ldots, n-k\,.$$

By [24, Claim 3] (originally proven in [25]), we have

$$\max_{1 \leq i \leq n-k} \left| x_i^{(\ell)}/y_i - 1 \right| \leq (1-\xi) \max_{1 \leq i \leq n-k} \left| x_i^{(\ell-1)}/y_i - 1 \right|,$$

where $\xi := 1/(1 + n^2/(4k(k-1))) \geq 4(k-1)^2/(5n^2)$. Furthermore, notice that

$$\max_{1 \leq i \leq n-k} \left| x_i^{(0)}/y_i - 1 \right| \leq \frac{k(n-k)\log(5\|\mathbf{B}^{(0)}\|)}{y_1}\,.$$

It follows that

$$\begin{aligned} \frac{x_1^{(N)} - y_1}{y_1} &\leq (1-\xi)^N \max_{1 \leq i \leq n-k} \left| x_i^{(0)}/y_i - 1 \right| \\ &\leq e^{-4(k-1)^2 N/(5n^2)} \cdot \frac{k(n-k)\log(5\|\mathbf{B}^{(0)}\|)}{y_1} \\ &\leq \frac{k \log(1+\varepsilon)}{y_1}\,. \end{aligned}$$

In other words,

$$\mathrm{vol}\big(\mathbf{B}_{[1,k]}^{(N)}\big) \leq (1+\varepsilon)^k \cdot (\delta^2\gamma_k)^{\frac{(n-k)k}{2(k-1)}} \mathrm{vol}(\mathcal{L})^{k/n} .$$

Notice that the first vector $\boldsymbol{b}_1$ of the output basis is a δ-approximate shortest vector in $\mathcal{L}\big(\mathbf{B}_{[1,k]}^{(N)}\big)$. Therefore,

$$\|\boldsymbol{b}_1\| \leq \delta\sqrt{\gamma_k} \cdot \mathrm{vol}\big(\mathbf{B}_{[1,k]}^{(N)}\big)^{1/k} \leq (1+\varepsilon)(\delta^2\gamma_k)^{\frac{n-1}{2(k-1)}} \mathrm{vol}(\mathcal{L})^{1/n} ,$$

as needed. □

3 Slide reduction for $n \leq 2k$

In this section, we consider a generalization of Gama and Nguyen's slide reduction that applies to the case when $k < n \leq 2k$ [9]. Our definition in this case is not particularly novel or surprising, as it is essentially identical to Gama and Nguyen's except that our blocks are not the same size.[9]

What *is* surprising about this definition is that it allows us to achieve sub-linear approximation factors for SVP when the rank is $n = k + q$ for $q = \Theta(k)$. Before this work, it seemed that approximation factors less than roughly $\gamma_q \approx n$ could not be achieved using the techniques of slide reduction (or, for that matter, any other known techniques with formal proofs). Indeed, our slide-reduced basis only achieves $\|\boldsymbol{b}_1\| \lesssim \gamma_q\lambda_1(\mathcal{L})$ (see [20, Th. 4.3.1]), which is the approximation factor resulting from the gluing lemma, Lemma 1. (This inequality is tight.) We overcome this barrier by using our additional constraints on the primal together with some additional properties of slide-reduced bases (namely, Eq. (4)) to bound $\lambda_1(\mathcal{L}(\mathbf{B}_{[1,k]}))$. Perhaps surprisingly, the resulting bound is much better than the bound on $\|\boldsymbol{b}_1\|$, which allows us to find a much shorter vector with an additional oracle call.

Definition 1 (Slide reduction). *Let $n = k + q$ where $1 \leq q \leq k$ are integers. A basis $\mathbf{B}$ of a lattice with rank n is* (δ, k)-slide-reduced *(with block size $k \geq 2$ and approximation factor $\delta \geq 1$) if it is size-reduced and satisfies the following set of conditions.*

1. *Primal conditions: The blocks $\mathbf{B}_{[1,q]}$ and $\mathbf{B}_{[i,n]}$ for $i \in [q+1, \max\{k, q+1\}]$ are δ-SVP-reduced.*
2. *Dual condition: the block $\mathbf{B}_{[2,q+1]}$ is δ-DSVP-reduced.*

[9] The only difference, apart from the approximation factor δ, is that we use SVP reduction instead of HKZ reduction for the primal. It is clear from the proof in [9] that only SVP reduction is required, as was observed in [24]. We *do* require that additional blocks $\mathbf{B}_{[i,n]}$ for $q+1 \leq i \leq k$ are SVP-reduced, which is quite similar to simply HKZ-reducing $\mathbf{B}_{[q+1,n]}$, but this requirement plays a distinct role in our analysis, as we discuss below.

A reader familiar with the slide reduction algorithm from [9] will not be surprised to learn that such a basis can be found (up to some small slack) using polynomially many calls to a δ-SVP oracle on lattices with rank at most k. Before presenting and analyzing the algorithm, we show that such a slide-reduced basis is in fact useful for approximating SVP with sub-linear factors. (We note in passing that a slight modification of the proof of Theorem 5 yields a better result when $q = o(k)$. This does not seem very useful on its own, though, since when $q = o(k)$, the running times of our best SVP algorithms are essentially the same for rank k and rank $k + q$.)

Theorem 5. *Let $\mathcal{L}$ be a lattice with rank $n = k+q$ where $2 \leq q \leq k$ are integers. For any $\delta \geq 1$, if a basis $\boldsymbol{B}$ of $\mathcal{L}$ is (δ, k)-slide-reduced, then,*

$$\lambda_1(\mathcal{L}(\mathbf{B}_{[1,k]})) \leq \delta\sqrt{\gamma_k}(\delta^2\gamma_q)^{\frac{q+1}{q-1}\cdot\frac{n-k}{2k}}\lambda_1(\mathcal{L}) \ .$$

Proof. Let $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$. We distinguish two cases.

First, suppose that there exists an index $i \in [q+1, \max\{k, q+1\}]$ such that $\|\boldsymbol{b}_i^*\| > \delta\lambda_1(\mathcal{L})$. Let $\boldsymbol{v}$ be a shortest non-zero vector of $\mathcal{L}$. We claim that $\pi_i(\boldsymbol{v}) = 0$, i.e., that $\boldsymbol{v} \in \mathcal{L}(\mathbf{B}_{[1,i-1]})$. If this is not the case, since $\mathbf{B}_{[i,n]}$ is δ-SVP-reduced, we have that

$$\|\boldsymbol{b}_i^*\|/\delta \leq \|\pi_i(\boldsymbol{v})\| \leq \|\boldsymbol{v}\| = \lambda_1(\mathcal{L}),$$

which is a contradiction. Thus, we see that $\boldsymbol{v} \in \mathcal{L}(\mathbf{B}_{[1,i-1]}) \subseteq \mathcal{L}(\mathbf{B}_{[1,k]})$, and hence $\lambda_1(\mathcal{L}(\boldsymbol{B}_{[1,k]})) = \lambda_1(\mathcal{L})$ (which is much stronger than what we need).

Now, suppose that $\|\boldsymbol{b}_i^*\| \leq \delta\lambda_1(\mathcal{L})$ for all indices $i \in [q+1, \max\{k, q+1\}]$. By definition, the primal and dual conditions imply that $\boldsymbol{B}_{[1,q+1]}$ is $\delta\sqrt{\gamma_q}$-twin-reduced. Therefore, by Eq. (4) of Fact 3, we have

$$\begin{aligned}
\mathrm{vol}(\mathbf{B}_{[1,k]}) &= \mathrm{vol}(\mathbf{B}_{[1,q]}) \cdot \prod_{i=q+1}^{k} \|\boldsymbol{b}_i^*\| \\
&\leq (\delta\sqrt{\gamma_q})^{q(q+1)/(q-1))}\|\boldsymbol{b}_{q+1}^*\|^q \cdot \prod_{i=q+1}^{k} \|\boldsymbol{b}_i^*\| \\
&\leq (\delta^2\gamma_q)^{\frac{q+1}{q-1}\cdot\frac{n-k}{2}}(\delta\lambda_1(\mathcal{L}))^k \ ,
\end{aligned}$$

where we have used the assumption that $\|\boldsymbol{b}_i^*\| \leq \delta\lambda_1(L)$ for all indices $i \in [q+1, \max\{k, q+1\}]$ (and by convention we take the product to equal one in the special case when $q = k$). By the definition of Hermite's constant, this implies that

$$\lambda_1(\mathcal{L}(\mathbf{B}_{[1,k]})) \leq \sqrt{\gamma_k}\mathrm{vol}(\mathbf{B}_{[1,k]})^{1/k} \leq \delta\sqrt{\gamma_k}(\delta^2\gamma_q)^{\frac{q+1}{q-1}\cdot\frac{n-k}{2k}}\lambda_1(\mathcal{L}) \ ,$$

as needed. □

Algorithm 2 The slide reduction algorithm for $n \leq 2k$ (adapted from [9, Algorithm 1])

Input: Block size k, slack $\varepsilon > 0$, approximation factor $\delta \geq 1$, a basis $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n) \in \mathbb{Z}^{m \times n}$ of a lattice $\mathcal{L}$ with rank $n = k + q$ where $2 \leq q \leq k$, and access to a δ-SVP oracle for lattices with rank at most k.
Output: A $((1+\varepsilon)\delta, k)$-slide-reduced basis of $\mathcal{L}$.

1: **while** $\mathrm{vol}(\boldsymbol{B}_{[1,q]})^2$ is modified by the loop **do**
2: $\quad\delta$-SVP-reduce $\mathbf{B}_{[1,q]}$.
3: $\quad$**for** $i = q+1$ **to** $\max\{k, q+1\}$ **do**
4: $\quad\quad\delta$-SVP reduce $\mathbf{B}_{[i,n]}$.
5: $\quad$**end for**
6: $\quad$Find a new basis $\boldsymbol{C} := (\boldsymbol{b}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_{q+1}, \boldsymbol{b}_{q+2}, \ldots, \boldsymbol{b}_n)$ of $\mathcal{L}$ by δ-DSVP-reducing $\mathbf{B}_{[2,q+1]}$.
7: $\quad$**if** $(1+\varepsilon)\|\boldsymbol{b}^*_{q+1}\| < \|\boldsymbol{c}^*_{q+1}\|$ **then**
8: $\quad\quad\mathbf{B} \leftarrow \boldsymbol{C}$.
9: $\quad$**end if**
10: **end while**
11: **return** $\boldsymbol{B}$.

3.1 The slide reduction algorithm for $n \leq 2k$

We now present our slight generalization of Gama and Nguyen's slide reduction algorithm that works for all $k + 2 \leq n \leq 2k$.

Our proof that Algorithm 2 runs in polynomial time (excluding oracle calls) is essentially identical to the proof in [9].

Theorem 6. *For $\varepsilon \geq 1/\mathrm{poly}(n)$, Algorithm 2 runs in polynomial time (excluding oracle calls), makes polynomially many calls to its δ-SVP oracle, and outputs a $((1+\varepsilon)\delta, k)$-slide-reduced basis of the input lattice $\mathcal{L}$.*

Proof. First, notice that if Algorithm 2 terminates, then its output must be $((1+\varepsilon)\delta, k)$-slide-reduced. So, we only need to argue that the algorithm runs in polynomial time (excluding oracle calls).

Let $\mathbf{B}_0 \in \mathbb{Z}^{m \times n}$ be the input basis and let $\mathbf{B} \in \mathbb{Z}^{m \times n}$ denote the current basis during the execution of the algorithm. As is common in the analysis of basis reduction algorithms [16, 9, 18], we consider an integral potential of the form

$$P(\mathbf{B}) := \mathrm{vol}(\mathbf{B}_{[1,q]})^2 \in \mathbb{Z}^+ .$$

The initial potential satisfies $\log P(\mathbf{B}_0) \leq 2q \cdot \log \|\mathbf{B}_0\|$, and every operation in Algorithm 2 either preserves or significantly decreases $P(\mathbf{B})$. More precisely, if the δ-DSVP-reduction step (i.e., Step 8) occurs, then the potential $P(\mathbf{B})$ decreases by a multiplicative factor of at least $(1+\varepsilon)^2$. No other step changes $\mathcal{L}(\mathbf{B}_{[1,q]})$ or $P(\mathbf{B})$.

Therefore, Algorithm 2 updates $\mathcal{L}(\mathbf{B}_{[1,q]})$ at most $\frac{\log P(\mathbf{B}_0)}{2\log(1+\varepsilon)}$ times, and hence it makes at most $\frac{qk \log \|\mathbf{B}_0\|}{\log(1+\varepsilon)}$ calls to the δ-SVP-oracle. From the complexity statement in Section 2.3, it follows that Algorithm 2 runs efficiently (excluding the running time of oracle calls). □

Corollary 1. *For any constant $c \in (1/2, 1]$ and $\delta := \delta(n) \geq 1$, there is an efficient reduction from $O(\delta^{2c+1}n^c)$-SVP on lattices with rank n to δ-SVP on lattices with rank $k := \lceil n/(2c) \rceil$.*

Proof. On input (a basis for) an integer lattice $\mathcal{L} \subseteq \mathbb{Z}^m$ with rank n, the reduction first calls Algorithm 2 to compute a $((1+\varepsilon)\delta, k)$-slide-reduced basis $\mathbf{B}$ of $\mathcal{L}$ with, say, $\varepsilon = 1/n$. The reduction then uses its δ-SVP oracle once more on $\mathbf{B}_{[1,k]}$ and returns the resulting nonzero short lattice vector.

It is immediate from Theorem 6 that this reduction is efficient, and by Theorem 5, the output vector is a δ'-approximate shortest vector, where

$$\delta' = \delta^2 \sqrt{\gamma_k}((1+\varepsilon)^2\delta^2\gamma_q)^{\frac{q+1}{q-1}\cdot\frac{n-k}{2k}} \leq O(\delta^{2c+1}n^c) \ ,$$

as needed. □

4 Slide reduction for $n \geq 2k$

We now introduce a generalized version of slide reduction for lattices with any rank $n \geq 2k$. As we explained in Section 1.2, at a high level, our generalization of the definition from [9] is the same as the original, except that (1) our first block $\mathbf{B}_{[1,k+q]}$ is bigger than the others (out of necessity, since we can no longer divide our basis evenly into disjoint blocks of size k); and (2) we only η-HSVP reduce the first block (since we cannot afford to δ-SVP reduce a block with size larger than k). Thus, our notion of slide reduction can be restated as "the first block and the first dual block are η-(D)HSVP reduced and the rest of the basis $\mathbf{B}_{[k+q+1,n]}$ is slide-reduced in the sense of [9]."[10]

However, the specific value of η that we choose in our definition below might look unnatural at first. We first present the definition and then explain where η comes from.

Definition 2 (Slide reduction). *Let n, k, p, q be integers such that $n = pk + q$ with $p, k \geq 2$ and $0 \leq q \leq k-1$, and let $\delta \geq 1$. A basis $\mathbf{B} \in \mathbb{R}^{m\times n}$ is* (δ, k)-slide-reduced *if it is size-reduced and satisfies the following three sets of conditions.*

1. *Mordell conditions: The block $\mathbf{B}_{[1,k+q]}$ is η-HSVP-reduced and the block $\mathbf{B}_{[2,k+q+1]}$ is η-DHSVP-reduced for $\eta := (\delta^2\gamma_k)^{\frac{k+q-1}{2(k-1)}}$.*
2. *Primal conditions: for all $i \in [1, p-1]$, the block $\mathbf{B}_{[ik+q+1,(i+1)k+q]}$ is δ-SVP-reduced.*
3. *Dual conditions: for all $i \in [1, p-2]$, the block $\mathbf{B}_{[ik+q+2,(i+1)k+q+1]}$ is δ-DSVP-reduced.*[11]

[10] Apart from the approximation factor δ, there is one minor difference between our primal conditions and those of [9]. We only require the primal blocks to be SVP-reduced, while [9] required them to be HKZ-reduced, which is a stronger condition. It is clear from the proof in [9] that only SVP reduction is required, as was observed in [24].

[11] When $p = 2$, there are simply no dual conditions.

There are two ways to explain our specific choice of η. Most simply, notice that the output of the DBKZ algorithm—due to [24] and presented in Section 2.4—is η-HSVP reduced when the input basis has rank $k+q$ (up to some small slack ε). In other words, one reason that we choose this value of η is because we actually can η-HSVP reduce a block of size $k+q$ efficiently with access to a δ-SVP oracle for lattices with rank k. If we could do better, then we would in fact obtain a better algorithm, but we do not know how. Second, this value of η is natural in this context because it is the choice that "makes the final approximation factor for HSVP match the approximation factor for the first block." I.e., the theorem below shows that when we plug in this value of η, a slide-reduced basis of rank n is $(\delta^2\gamma_k)^{\frac{n-1}{2(k-1)}}$-HSVP, which nicely matches the approximation factor of $\eta = (\delta^2\gamma_k)^{\frac{k+q-1}{2(k-1)}}$-HSVP that we need for the first block (whose rank is $k+q$). At a technical level, this is captured by Fact 3 and Lemma 1.

Of course, the fact that these two arguments suggest the same value of η is not a coincidence. Both arguments are essentially disguised proofs of Mordell's inequality, which says that $\gamma_n \leq \gamma_k^{(n-1)/(k-1)}$ for $2 \leq k \leq n$. E.g., with $\delta = 1$ the primal Mordell condition says that $\boldsymbol{b}_1$ yields a witness to Mordell's inequality for $\mathbf{B}_{[1,k+q]}$.

Theorem 7. *For any $\delta \geq 1$, $k \geq 2$, and $n \geq 2k$, if $\mathbf{B} = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n) \in \mathbb{R}^{m \times n}$ is a (δ, k)-slide-reduced basis of a lattice $\mathcal{L}$, then*

$$\|\mathbf{b}_1\| \leq (\delta^2\gamma_k)^{\frac{n-1}{2(k-1)}} \operatorname{vol}(\mathcal{L})^{1/n} . \tag{5}$$

Furthermore, if $\lambda_1(\mathcal{L}(\mathbf{B}_{[1,k+q]})) > \lambda_1(\mathcal{L})$, then

$$\|\mathbf{b}_1\| \leq \delta(\delta^2\gamma_k)^{\frac{n-k}{k-1}} \lambda_1(\mathcal{L}) , \tag{6}$$

where $0 \leq q \leq k-1$ is such that $n = pk + q$.

Proof. Let $d := k + q$. Theorem 9 of Appendix A shows that $\mathbf{B}_{[d+1,n]}$ is both $(\delta^2\gamma_k)^{\frac{n-d-1}{2(k-1)}}$-HSVP-reduced and $(\delta^2\gamma_k)^{\frac{n-d-k}{(k-1)}}$-SVP-reduced. (We relegate this theorem and its proof to the appendix because it is essentially just a restatement of [9, Theorem 1], since $\mathbf{B}_{[d+1,n]}$ is effectively just a slide-reduced basis in the original sense of [9].) Furthermore, $\mathbf{B}_{[1,d+1]}$ is $(\delta^2\gamma_k)^{\frac{d-1}{2(k-1)}}$-twin-reduced, so that $\|\boldsymbol{b}_1\| \leq (\delta^2\gamma_k)^{\frac{d}{k-1}} \|\boldsymbol{b}^*_{d+1}\|$. Applying Lemma 1 then yields both Eq. (5) and Eq. (6). □

4.1 The slide reduction algorithm for $n \geq 2k$

We now present our slight generalization of Gama and Nguyen's slide reduction algorithm that works for all $n \geq 2k$. Our proof that the algorithm runs in polynomial time (excluding oracle calls) is essentially identical to the proof in [9].

Theorem 8. *For $\varepsilon \in [1/\mathrm{poly}(n), 1]$, Algorithm 3 runs in polynomial time (excluding oracle calls), makes polynomially many calls to its δ-SVP oracle, and outputs a $((1+\varepsilon)\delta, k)$-slide-reduced basis of the input lattice $\mathcal{L}$.*

Algorithm 3 The slide-reduction algorithm for $n \geq 2k$

Input: Block size $k \geq 2$, slack $\varepsilon > 0$, approximation factor $\delta \geq 1$, basis $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n) \in \mathbb{Z}^{m \times n}$ of a lattice $\mathcal{L}$ of rank $n = pk + q \geq 2k$ for $0 \leq q \leq k-1$, and access to a δ-SVP oracle for lattices with rank k.

Output: A $((1+\varepsilon)\delta, k)$-slide-reduced basis of $\mathcal{L}(\mathbf{B})$.

1: **while** $\mathrm{vol}(\mathbf{B}_{[1,ik+q]})^2$ is modified by the loop for some $i \in [1, p-1]$ **do**
2: $(1+\varepsilon)\eta$-HSVP-reduce $\mathbf{B}_{[1,k+q]}$ using Alg. 1 for $\eta := (\delta^2\gamma_k)^{\frac{k+q-1}{2(k-1)}}$.
3: **for** $i = 1$ **to** $p-1$ **do**
4: δ-SVP-reduce $\mathbf{B}_{[ik+q+1,(i+1)k+q]}$.
5: **end for**
6: **if** $\mathbf{B}_{[2,k+q+1]}$ is not $(1+\varepsilon)\eta$-DHSVP-reduced **then**
7: $(1+\varepsilon)^{1/2}\eta$-DHSVP-reduce $\mathbf{B}_{[2,k+q+1]}$ using Alg. 1.
8: **end if**
9: **for** $i = 1$ **to** $p-2$ **do**
10: Find a new basis $\boldsymbol{C} := (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_{ik+q+1}, \boldsymbol{c}_{ik+q+2}, \ldots, \boldsymbol{c}_{(i+1)k+q+1}, \boldsymbol{b}_{ik+q+2}, \ldots, \boldsymbol{b}_n)$ of $\mathcal{L}$ by δ-DSVP-reducing $\mathbf{B}_{[ik+q+2,(i+1)k+q+1]}$.
11: **if** $(1+\varepsilon)\|\boldsymbol{b}^*_{(i+1)k+q+1}\| < \|\boldsymbol{c}^*_{(i+1)k+q+1}\|$ **then**
12: $\mathbf{B} \leftarrow \boldsymbol{C}$.
13: **end if**
14: **end for**
15: **end while**
16: **return** $\mathbf{B}$.

Proof. First, notice that if Algorithm 3 terminates, then its output is $((1+\varepsilon)\delta, k)$-slide-reduced. So, we only need to argue that the algorithm runs in polynomial time (excluding oracle calls).

Let $\mathbf{B}_0 \in \mathbb{Z}^{m \times n}$ be the input basis and let $\mathbf{B} \in \mathbb{Z}^{m \times n}$ denote the current basis during the execution of Algorithm 3. As is common in the analysis of basis reduction algorithms [16, 9, 18], we consider an integral potential of the form

$$P(\mathbf{B}) := \prod_{i=1}^{p-1} \mathrm{vol}(\mathbf{B}_{[1,ik+q]})^2 \in \mathbb{Z}^+.$$

The initial potential satisfies $\log P(\mathbf{B}_0) \leq 2n^2 \cdot \log \|\mathbf{B}_0\|$, and every operation in Algorithm 3 either preserves or significantly decreases $P(\mathbf{B})$. In particular, the potential is unaffected by the primal steps (i.e., Steps 2 and 4), which leave $\mathrm{vol}(\mathbf{B}_{[1,ik+q]})$ unchanged for all i. The dual steps (i.e., Steps 7 and 12) either leave $\mathrm{vol}(\mathbf{B}_{[1,ik+q]})$ for all i or decrease $P(\mathbf{B})$ by a multiplicative factor of at least $(1+\varepsilon)$.

Therefore, Algorithm 2 updates $\mathrm{vol}(\mathbf{B}_{[1,ik+q]})$ for some i at most $\log P(\mathbf{B}_0)/\log(1+\varepsilon)$ times. Hence, it makes at most $4pn^2 \log \|\mathbf{B}_0\|/\log(1+\varepsilon)$ calls to the SVP oracle in the SVP and DSVP reduction steps (i.e., Steps 4 and 12), and similarly at most $4n^2 \log \|\mathbf{B}_0\|/\log(1+\varepsilon)$ calls to Algorithm 1. From the complexity statement in Section 2.3, it follows that Algorithm 2 runs efficiently (excluding the running time of oracle calls), as needed.

□

Corollary 2. *For any constant $c \geq 1$ and $\delta := \delta(n) \geq 1$, there is an efficient reduction from $O(\delta^{2c+1}n^c)$-SVP on lattices with rank n to δ-SVP on lattices with rank $k := \lfloor n/(c+1) \rfloor$.*

Proof. On input (a basis for) an integer lattice $\mathcal{L} \subseteq \mathbb{Z}^m$ with rank n, the reduction first calls Algorithm 3 to compute a $((1+\varepsilon)\delta, k)$-slide-reduced basis $\mathbf{B} = (\mathbf{b}_1, \cdots, \mathbf{b}_n)$ of $\mathcal{L}$ with, say, $\varepsilon = 1/n$. Then, the reduction uses the procedure from Corollary 1 on the lattice $\mathcal{L}(\mathbf{B}_{[1,2k]})$ with $c = 1$ (i.e., slide reduction on a lattice with rank $2k$), to find a vector $\boldsymbol{v} \in \mathcal{L}(\mathbf{B}_{[1,2k]})$ with $0 < \|\boldsymbol{v}\| \leq O(\delta^3 n)\lambda_1(\mathcal{L}(\mathbf{B}_{[1,2k]}))$. Finally, the reduction outputs the shorter of the two vectors $\mathbf{b}_1$ and $\mathbf{v}$.

It is immediate from Corollary 1 and Theorem 8 that this reduction is efficient. To prove correctness, we consider two cases.

First, suppose that $\lambda_1(\mathcal{L}(\mathbf{B}_{[1,k+q]})) = \lambda_1(\mathcal{L})$. Then,

$$\|\boldsymbol{v}\| \leq O(\delta^3 n)\lambda_1(\mathcal{L}(\mathbf{B}_{[1,2k]})) \leq O(\delta^{2c+1}n^c)\lambda_1(\mathcal{L}) \ ,$$

so that the algorithm will output a $O(\delta^{2c+1}n^c)$-approximate shortest vector.

On the other hand, if $\lambda_1(\mathcal{L}(\mathbf{B}_{[1,k+q]})) > \lambda_1(\mathcal{L})$, then by Theorem 7, we have

$$\|\boldsymbol{b}_1\| \leq (1+\varepsilon)\delta((1+\varepsilon)^2\delta^2\gamma_k)^{\frac{n-k}{k-1}}\lambda_1(\mathcal{L}) \leq O(\delta^{2c+1}n^c) \ ,$$

so that the algorithm also outputs a $O(\delta^{2c+1}n^c)$-approximate shortest vector in this case. □

References

[1] D. Aggarwal and N. Stephens-Davidowitz. "Just take the average! An embarrassingly simple 2^n-time algorithm for SVP (and CVP)". In: *SOSA*. http://arxiv.org/abs/1709.01535. 2018.

[2] D. Aggarwal et al. "Solving the Shortest Vector Problem in 2^n time via Discrete Gaussian Sampling". In: *STOC*. http://arxiv.org/abs/1412.7994. 2015.

[3] M. Ajtai. "Generating hard instances of lattice problems". In: *STOC*. 1996. ISBN: 978-0-89791-785-8.

[4] M. Ajtai, R. Kumar, and D. Sivakumar. "A sieve algorithm for the Shortest Lattice Vector Problem". In: *STOC*. 2001.

[5] A. Becker et al. "New directions in nearest neighbor searching with applications to lattice sieving". In: *SODA*. 2016.

[6] Y. Chen and P. Q. Nguyen. "Faster algorithms for approximate common divisors: Breaking fully-homomorphic-encryption challenges over the integers". In: *EUROCRYPT*. 2012. ISBN: 978-3-642-29011-4.

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

[8] N. Gama, N. Howgrave-Graham, and P. Q. Nguyen. "Symplectic lattice reduction and NTRU". In: *EUROCRYPT*. 2006. ISBN: 978-3-540-34547-3.

[9] N. Gama and P. Q. Nguyen. "Finding short lattice vectors within Mordell's inequality". In: *STOC*. 2008. ISBN: 978-1-60558-047-0.
[10] N. Gama and P. Q. Nguyen. "Predicting Lattice Reduction". In: *EUROCRYPT '08*. 2008.
[11] N. Gama et al. "Rankin's constant and blockwise lattice reduction". In: *CRYPTO*. 2006. ISBN: 978-3-540-37433-6.
[12] C. Gentry, C. Peikert, and V. Vaikuntanathan. "Trapdoors for hard lattices and new cryptographic constructions". In: *STOC*. https://eprint.iacr.org/2007/432. 2008.
[13] G. Hanrot, X. Pujol, and D. Stehlé. "Analyzing blockwise lattice algorithms using dynamical systems". In: *CRYPTO*. 2011. ISBN: 978-3-642-22792-9.
[14] A. Joux and J. Stern. "Lattice reduction: A toolbox for the cryptanalyst". In: *J. Cryptology* 11.3 (1998).
[15] R. Kannan. "Improved algorithms for integer programming and related lattice problems". In: *STOC*. 1983.
[16] A. K. Lenstra, H. W. Lenstra Jr., and L. Lovász. "Factoring polynomials with rational coefficients". In: *Mathematische Annalen* 261.4 (1982).
[17] H. W. Lenstra Jr. "Integer programming with a fixed number of variables". In: *Mathematics of Operations Research* 8.4 (1983).
[18] J. Li and P. Q. Nguyen. "Approximating the densest sublattice from Rankin's inequality". In: *LMS Journal of Computation and Mathematics* 17.Special Issue A (2014). Contributed to ANTS-XI, 2014.
[19] J. Li and P. Q. Nguyen. "Computing a Lattice Basis Revisited". In: *ISSAC*. 2019.
[20] J. Li and W. Wei. "Slide reduction, successive minima and several applications". In: *Bulletin of the Australian Mathematical Society* 88 (03 2013), pp. 390–406.
[21] M. Liu et al. "Shortest lattice vectors in the presence of gaps". http://eprint.iacr.org/2011/139. 2011.
[22] L. Lovász. *An algorithmic theory of numbers, graphs and convexity*. Society for Industrial and Applied Mathematics, 1986. ISBN: 978-0-89871-203-2.
[23] D. Micciancio and P. Voulgaris. "A deterministic single exponential time algorithm for most lattice problems based on Voronoi cell computations". In: *SIAM J. on Computing* 42.3 (2013).
[24] D. Micciancio and M. Walter. "Practical, predictable lattice basis reduction". In: *Eurocrypt*. http://eprint.iacr.org/2015/1123. 2016.
[25] A. Neumaier. "Bounding basis reduction properties". In: *Designs, Codes and Cryptography* 84.1 (2017).
[26] P. Q. Nguyen and J. Stern. "The two faces of lattices in cryptology". In: *CaLC*. 2001.
[27] P. Q. Nguyen and B. Vallée, eds. *The LLL algorithm: Survey and applications*. Springer-Verlag, 2010. ISBN: 978-3-642-02294-4.
[28] P. Q. Nguyen and T. Vidick. "Sieve algorithms for the Shortest Vector Problem are practical". In: *J. Mathematical Cryptology* 2.2 (2008).

[29] C. S. D. NIST. *Post-quantum cryptography*. https://csrc.nist.gov/Projects/Post-Quantum-Cryptography. 2018.
[30] A. M. Odlyzko. "The rise and fall of knapsack cryptosystems". In: *Cryptology and Computational Number Theory* 42 (1990).
[31] C. Peikert. "A decade of lattice cryptography". In: *Foundations and Trends in Theoretical Computer Science* 10.4 (2016).
[32] C. Peikert. "Public-key cryptosystems from the worst-case Shortest Vector Problem". In: *STOC*. 2009.
[33] X. Pujol and D. Stehlé. *Solving the Shortest Lattice Vector Problem in time* $2^{2.465n}$. http://eprint.iacr.org/2009/605. 2009.
[34] O. Regev. "On lattices, learning with errors, random linear codes, and cryptography". In: *J. ACM* 56.6 (2009).
[35] C.-P. Schnorr. "A hierarchy of polynomial time lattice basis reduction algorithms". In: *Theor. Comput. Sci.* 53.23 (1987).
[36] C.-P. Schnorr and M. Euchner. "Lattice basis reduction: Improved practical algorithms and solving subset sum problems". In: *Mathmatical Programming* 66 (1994).
[37] A. Shamir. "A polynomial-time algorithm for breaking the basic Merkle-Hellman cryptosystem". In: *IEEE Trans. Inform. Theory* 30.5 (1984).
[38] W. Wei, M. Liu, and X. Wang. "Finding shortest lattice vectors in the presence of gaps". In: *CT-RSA*. 2015. ISBN: 978-3-319-16715-2.

A Properties of Gama and Nguyen's slide reduction

In the theorem below, $\mathbf{B}_{[d+1,n]}$ is essentially just a slide-reduced basis in the sense of [9]. So, the following is more-or-less just a restatement of [9, Theorem 1].

Theorem 9. *Let* $\mathbf{B} = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n) \in \mathbb{R}^{m \times n}$ *with* $n = pk + d$ *for some* $p \geq 1$ *and* $d \geq k$ *be* (δ, k)*-slide reduced in the sense of Definition 2. Then,*

$$\|\mathbf{b}^*_{d+1}\| \leq (\delta^2 \gamma_k)^{ik/(k-1)} \|\mathbf{b}^*_{ik+d+1}\| \text{ for } 0 \leq i \leq p-1 \,, \tag{7}$$

$$\|\mathbf{b}^*_{d+1}\| \leq (\delta^2 \gamma_k)^{\frac{n-d-1}{2(k-1)}} \mathrm{vol}(\mathbf{B}_{[d+1,n]})^{1/(n-d)} \,, \text{ and} \tag{8}$$

$$\|\mathbf{b}^*_{d+1}\| \leq \delta (\delta^2 \gamma_k)^{\frac{n-d-k}{k-1}} \lambda_1(\mathcal{L}(\mathbf{B}_{[d+1,n]})) \,. \tag{9}$$

Proof. By definition, for each $i \in [0, p-2]$, the block $\mathbf{B}_{[ik+d+1,(i+1)k+d+1]}$ is $\delta\sqrt{\gamma_k}$-twin-reduced. By Eq. (3) of Fact 3, we see that

$$\|\boldsymbol{b}^*_{ik+d+1}\| \leq (\delta\sqrt{\gamma_k})^{2k/(k-1)} \|\boldsymbol{b}^*_{(i+1)k+d+1}\| \,,$$

which implies (7) by induction.

We prove (8) and (9) by induction over p. If $p = 1$, then both inequalities hold as $\mathbf{B}_{[d+1,n]}$ is δ-SVP reduced by the definition of slide reduction. Now, assume that Eqs. (8) and (9) hold for some $p \geq 1$. Let $n = (p+1)k + d$. Then $\mathbf{B}$ satisfies

the requirements of the theorem with $d' := d + k$. Therefore, by the induction hypothesis, we have

$$\|\mathbf{b}^*_{d+k+1}\| \leq (\delta^2\gamma_k)^{\frac{n-d-k-1}{2(k-1)}} \mathrm{vol}(\mathbf{B}_{[d+k+1,n]})^{1/(n-d-k)} \text{ , and}$$
$$\|\mathbf{b}^*_{d+k+1}\| \leq \delta(\delta^2\gamma_k)^{\frac{n-d-2k}{k-1}} \lambda_1(\mathcal{L}(\mathbf{B}_{[d+k+1,n]})) \text{ .}$$

Since $\mathbf{B}_{[d+1,d+k]}$ is $\delta\sqrt{\gamma_k}$-HSVP reduced, we may apply Lemma 1.2 with $\eta = (\delta^2\gamma_k)^{\frac{1}{2(k-1)}}$, which proves (8) for $\mathbf{B}_{[d+1,n]}$.

Furthermore, if $\lambda_1(\mathcal{L}(\mathbf{B}_{[d+1,n]})) < \lambda_1(\mathcal{L}(\mathbf{B}_{[d+1,d+k]}))$, it follows from Lemma 1.1 that $\mathbf{B}_{[d+1,n]}$ is δ'-SVP-reduced for

$$\delta' = (\delta^2\gamma_k)^{k/(k-1)} \cdot \delta(\delta^2\gamma_k)^{\frac{n-d-2k}{k-1}} = \delta(\delta^2\gamma_k)^{\frac{n-d-k}{k-1}} \text{ ,}$$

as needed. If not, then $\lambda_1(\mathcal{L}(\mathbf{B}_{[d+1,n]})) = \lambda_1(\mathcal{L}(\mathbf{B}_{[d+1,d+k]}))$, and $\|\boldsymbol{b}^*_{d+1}\| \leq \delta\lambda_1(\mathcal{L}(\mathbf{B}_{[d+1,n]}))$ because $\mathbf{B}_{[d+1,d+k]}$ is δ-SVP reduced. In all cases, we proved (9). This completes the proof of Theorem 9. □