LMS J. Comput. Math. 17 (Special issue A) (2014) 92–111
doi:10.1112/S1461157014000333

Approximating the densest sublattice from Rankin's inequality

Jianwei Li and Phong Q. Nguyen

Abstract

We present a higher-dimensional generalization of the Gama–Nguyen algorithm (STOC '08) for approximating the shortest vector problem in a lattice. This generalization approximates the densest sublattice by using a subroutine solving the exact problem in low dimension, such as the Dadush–Micciancio algorithm (SODA '13). Our approximation factor corresponds to a natural inequality on Rankin's constant derived from Rankin's inequality.

1. *Introduction*

Lattices are discrete subgroups of $\mathbb{R}^m$. Any lattice L has a basis: a set of linearly independent vectors $\mathbf{b}_1, \ldots, \mathbf{b}_n$ in $\mathbb{R}^m$ such that L is equal to the set $L(\mathbf{b}_1, \ldots, \mathbf{b}_n) = \{\sum_{i=1}^n x_i \mathbf{b}_i, x_i \in \mathbb{Z}\}$ of all integer linear combinations of the $\mathbf{b}_i$. All the bases of L have the same number n of elements, called the dimension of L, and they all have the same n-dimensional volume, called the volume $\mathrm{vol}(L)$ or determinant of L. Let $\lambda_1(L)$ be the first minimum of L, that is, the minimal Euclidean distance between two lattice points. In the 19th century, Hermite [**7**] proved the inequality $\lambda_1(L)/\mathrm{vol}(L)^{1/n} \leqslant (\frac{4}{3})^{(n-1)/4}$, which gave rise to Hermite's constant γ_n, defined as the supremum of $(\lambda_1(L)/\mathrm{vol}(L)^{1/n})^2$ over all n-dimensional lattices L, and satisfying

$$\text{(Hermite's inequality)} \quad \gamma_n \leqslant (\sqrt{\tfrac{4}{3}})^{n-1} = \gamma_2^{n-1}. \tag{1.1}$$

Hermite's inequality has an efficient algorithmic version: the celebrated LLL algorithm [**10**]. Given a basis B_0 of an n-dimensional integer lattice $L \subseteq \mathbb{Z}^m$ and a reduction factor $\varepsilon > 0$, LLL outputs (in time polynomial in $(\mathrm{size}(B_0), 1/\varepsilon)$) a reduced basis $(\mathbf{b}_1, \ldots, \mathbf{b}_n)$ whose first vector is provably short, namely

$$\|\mathbf{b}_1\|/\mathrm{vol}(L)^{1/n} \leqslant ((1+\varepsilon)\sqrt{\tfrac{4}{3}})^{(n-1)/2}, \tag{1.2}$$

$$\|\mathbf{b}_1\|/\lambda_1(L) \leqslant ((1+\varepsilon)\sqrt{\tfrac{4}{3}})^{n-1}. \tag{1.3}$$

Hermite's inequality (1.1) is implied by (1.2) with $\varepsilon = 0$. The second inequality (1.3) means that LLL approximates the shortest vector problem (SVP) within an exponential factor.

In 1944, Mordell [**15**] generalized Hermite's inequality (1.1) as follows

$$\text{(Mordell's inequality)} \quad \gamma_n \leqslant \gamma_k^{(n-1)/(k-1)} \quad \text{for any } 2 \leqslant k \leqslant n. \tag{1.4}$$

Gama and Nguyen [**5**] gave an efficient algorithmic version of Mordell's inequality: given a basis B_0 of an n-dimensional integer lattice $L \subseteq \mathbb{Z}^m$, a blocksize k dividing n, a reduction factor $\varepsilon > 0$, and an SVP-subroutine computing shortest vectors in any lattice of

Received 27 February 2014; revised 23 May 2014.

2010 Mathematics Subject Classification 11H06, 68W25 (primary), 11Y16 (secondary).

Contributed to the Algorithmic Number Theory Symposium XI, GyeongJu, Korea, 6–11 August 2014.

The work is supported in part by China's 973 program (No. 2013CB834205) and the National Natural Science Foundation of China (No. 61133013).

dimension $\leqslant k$, their slide reduction algorithm outputs (in time polynomial in $(\mathrm{size}(B_0), 1/\varepsilon)$) a basis whose first vector $\mathbf{b}_1$ satisfies

$$\|\mathbf{b}_1\|/\mathrm{vol}(L)^{1/n} \leqslant (\sqrt{1+\varepsilon}\gamma_k)^{(n-1)/2(k-1)}, \tag{1.5}$$

$$\|\mathbf{b}_1\|/\lambda_1(L) \leqslant (\sqrt{1+\varepsilon}\gamma_k)^{(n-k)/(k-1)}, \tag{1.6}$$

and the number of calls to the SVP-subroutine is polynomial in $(\mathrm{size}(B_0), 1/\varepsilon)$. As an SVP-subroutine, one can take the Micciancio–Voulgaris algorithm [14], which runs in 2^{2k} polynomial-time operations and exponential space. The exponential cost of the SVP-subroutine can be kept polynomial in the size of the basis if the blocksize k is sufficiently small, namely, $k = O(\log n)$ for the MV algorithm [14], in which case the Gama–Nguyen algorithm achieves in polynomial time a subexponential approximation factor $2^{O(n \log\log n/\log n)}$.

In 1953, Rankin [17] introduced a higher-dimensional generalization of Hermite's constant. For any n-dimensional lattice L and $1 \leqslant r \leqslant n$, its Rankin invariant $\gamma_{n,r}(L)$ is defined as

$$\gamma_{n,r}(L) = \min_{\substack{\mathbf{x}_1,\ldots,\mathbf{x}_r \in L \\ \mathrm{vol}(\mathbf{x}_1,\ldots,\mathbf{x}_r)\neq 0}} \left(\frac{\mathrm{vol}(\mathbf{x}_1,\ldots,\mathbf{x}_r)}{\mathrm{vol}(L)^{r/n}}\right)^2 = \min_{\substack{S \text{ sublattice of } L \\ \dim S = r}} \left(\frac{\mathrm{vol}(S)}{\mathrm{vol}(L)^{r/n}}\right)^2. \tag{1.7}$$

Rankin's constant is $\gamma_{n,r} = \max \gamma_{n,r}(L)$ over all n-dimensional lattices L, which satisfies $\gamma_{n,r} \leqslant \gamma_n^r$. Hermite's constant is the special case $r = 1$: $\gamma_n = \gamma_{n,1}$. Gama *et al.* [3] introduced a natural generalization of the shortest vector problem, which is to Rankin's constant what SVP is to Hermite's constant. The so-called *r-dimensional densest sublattice problem* (r-DSP) asks to find a sublattice reaching the Rankin invariant: given an n-dimensional lattice L and an integer $r \in \{1, \ldots n\}$, find an r-dimensional sublattice S of L such that $\sqrt{\gamma_{n,r}(L)} = \mathrm{vol}(S)/\mathrm{vol}(L)^{r/n}$. There is a reduction from SVP to r-DSP for $1 \leqslant r < n$, then r-DSP is NP-hard under randomized reductions [2]. In the special case $r = 1$, the densest sublattice problem is simply SVP. Dadush and Micciancio [2] showed how to solve r-DSP exactly with running time $r^{O(r\cdot n)}$ and $2^n \mathrm{poly}(n)$ space for any $r < n$. In its approximate version introduced by [3], one is asked to find an r-dimensional sublattice S of L such that $\mathrm{vol}(S)/\mathrm{vol}(L)^{r/n} \leqslant f(n,r)$ for some approximation factor $f(n,r)$. The case $(n,r) = (2k,k)$ is known as the *half-volume problem* (see [3]), and is used in blocksize reduction algorithms [3, 5, 19] for approximating SVP, where one needs to approximate DSP using an SVP-oracle in dimensions $\leqslant k+1$ (see transference reduction in [3]). Approximating DSP also arises in enumeration algorithms for SVP [6, 20, 21]: the cost of enumeration depends on the quality of the basis with respect to DSP; the better the approximation factor, the faster the enumeration.

Our results. The Dadush–Micciancio algorithm for exact DSP suggests finding a DSP approximation algorithm using an exact algorithm in low dimension, similar to blocksize approximation algorithms for SVP, which rely on an exact algorithm in low dimension. Such a framework was briefly discussed in [2, § 5.2]. However, one may wonder what should be the approximation factor.

We first show a simple generalization of Mordell's inequality for Rankin's constant.

THEOREM 1.1. *For all integers r and k such that $1 \leqslant r < k$, Rankin's constant satisfies*

$$\gamma_{n,r} \leqslant \gamma_{k,r}^{(n-r)/(k-r)} \tag{1.8}$$

for any $n \geqslant r$ of the form $n = pk + \ell r$ for some integers $p \geqslant 1$ and $\ell \geqslant 0$.

The inequality (1.8) with $r = 1$ becomes Mordell's inequality (1.4). Yet, for $r \geqslant 2$, (1.8) appears to be new, though we derive it from elementary inequalities due to Rankin.

https://doi.org/10.1112/S1461157014000333 Published online by Cambridge University Press

Next, we show that (1.8) has an efficient algorithmic version like Mordell's inequality, by generalizing the Gama–Nguyen SVP-approximation algorithm using a DSP-oracle instead of a SVP-oracle. Given a basis B_0 of an n-dimensional integer lattice $L \subseteq \mathbb{Z}^m$, a blocksize k dividing n, an index $r < k$, a reduction factor $\varepsilon > 0$, and a DSP-subroutine computing the r-dimensional densest sublattice in any lattice of dimension k, one can compute (in time polynomial in $(\mathrm{size}(B_0), 1/\varepsilon)$) a basis $(\mathbf{b}_1, \ldots, \mathbf{b}_n)$ such that

$$\mathrm{vol}(\mathbf{b}_1, \ldots, \mathbf{b}_r) < (\sqrt{1+\varepsilon}\gamma_{k,r})^{(n-r)/2(k-r)}\mathrm{vol}(L)^{r/n} \tag{1.9}$$

where the number of calls to the DSP-subroutine is polynomial in $(\mathrm{size}(B_0), 1/\varepsilon)$, and the input to the subroutine always has polynomial size.

Roadmap. In § 2, we provide background on lattices and lattice reduction. In § 3, we prove Theorem 1.1. In § 4, we present an efficient algorithmic version of Theorem 1.1, which we call block–Rankin reduction. In Appendices A–D, we provide missing proofs.

2. *Background*

In this paper, we use bold lower case letters to denote vectors in column notation, and use column-representation for matrices: the matrix is denoted by upper case letters, and its coefficients are denoted by lower case letters. The ring of $m \times n$ matrices with coefficients in the ring $\mathbb{A}$ is denoted by $\mathbb{A}^{m\times n}$. The $n \times n$ identity matrix is denoted by I_n. For a matrix $B = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ of n columns, we denote $\|B\| = \max\{\|\mathbf{b}_1\|, \ldots, \|\mathbf{b}_n\|\}$, while $\|\cdot\|_F$ denotes the Frobenius norm: $\|B\|_F = \sqrt{\sum_{i=1}^n \|\mathbf{b}_i\|^2}$. The size of an object corresponds to the length of its binary representation, for example, if $B = (b_{i,j}) \in \mathbb{Q}^{m\times n}$, then $\mathrm{size}(B) = \sum_{i\in[1,m],j\in[1,n]} \mathrm{size}(b_{i,j})$.

2.1. *Lattices*

Hermite's constant. The Hermite invariant of an n-dimensional lattice L is defined by $\gamma_n(L) = (\lambda_1(L)/\mathrm{vol}(L)^{1/n})^2$, where $\lambda_1(L) = \min_{\mathbf{v}\in L\setminus\{\mathbf{0}\}} \|\mathbf{v}\|$ is the first minimum of L. Hermite's constant is the maximum $\gamma_n = \max \gamma_n(L)$ over all n-dimensional lattices L. We have $\gamma_n < 1 + n/4$ for $n \geqslant 1$; see [12]. It is known that $\gamma_n \leqslant \gamma_{n-1}^{(n-1)/(n-2)}$; see [15].

Rankin's constant. For any n-dimensional lattice L and $1 \leqslant r \leqslant n$, Rankin [17] introduced the Rankin invariant $\gamma_{n,r}(L)$ defined as

$$\gamma_{n,r}(L) = \min_{\substack{\mathbf{x}_1,\ldots,\mathbf{x}_r\in L\\ \mathrm{vol}(\mathbf{x}_1,\ldots,\mathbf{x}_r)\neq 0}} \left(\frac{\mathrm{vol}(\mathbf{x}_1, \ldots, \mathbf{x}_r)}{\mathrm{vol}(L)^{r/n}}\right)^2 = \min_{S \text{ sublattice of } L \atop \dim S = r} \left(\frac{\mathrm{vol}(S)}{\mathrm{vol}(L)^{r/n}}\right)^2.$$

Rankin's constant is the maximum $\gamma_{n,r} = \max \gamma_{n,r}(L)$ over all n-dimensional lattices L. Clearly, $\gamma_{n,1}(L) = \gamma_n(L)$ and $\gamma_{n,1} = \gamma_n$.

Primitive set. Let L be an n-dimensional lattice. If $1 \leqslant i \leqslant n$, then i linearly independent vectors $\mathbf{b}_1, \ldots, \mathbf{b}_i \in L$ form a *primitive set* for L if and only if $\mathbf{b}_1, \ldots, \mathbf{b}_i$ can be extended to a basis of L. In particular, the vectors $\mathbf{b}_1, \ldots, \mathbf{b}_i \in \mathbb{Z}^n$ form a primitive set for $\mathbb{Z}^n$ if and only if $\mathbf{b}_1, \ldots, \mathbf{b}_i$ can be extended to a unimodular matrix.

Orthogonalization. Given a basis $B = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$, consider the orthogonal projections

$$\pi_i : \mathrm{span}(\mathbf{b}_1, \ldots, \mathbf{b}_n) \mapsto \mathrm{span}(\mathbf{b}_1, \ldots, \mathbf{b}_{i-1})^\perp, \quad \text{for } i = 1, \ldots, n,$$

https://doi.org/10.1112/S1461157014000333 Published online by Cambridge University Press

where $\mathrm{span}(\mathbf{b}_1, \ldots, \mathbf{b}_n)$ denotes the space spanned by $\mathbf{b}_1, \ldots, \mathbf{b}_n$. Then $L_i := \pi_i(L)$ is a lattice of dimension $n - i + 1$. We will use the notation $B_{[i,j]}$ for the projected block $(\pi_i(\mathbf{b}_i), \pi_i(\mathbf{b}_{i+1}), \ldots, \pi_i(\mathbf{b}_j))$. In particular, $B_{[i,n]}$ is a basis of L_i.

The vectors $\mathbf{b}_i^* = \pi_i(\mathbf{b}_i)$ for $1 \leqslant i \leqslant n$ are the Gram–Schmidt vectors of B. Its Gram–Schmidt coefficients are $\mu_{i,j} = \langle \mathbf{b}_i, \mathbf{b}_j^* \rangle / \langle \mathbf{b}_j^*, \mathbf{b}_j^* \rangle$ which can be defined recursively by: $\mathbf{b}_1^* = \mathbf{b}_1, \mathbf{b}_i^* = \mathbf{b}_i - \sum_{j=1}^{i-1} \mu_{i,j} \mathbf{b}_j^*$, for $i = 2, \ldots, n$. We have $\mu_{i,i} = 1$ and $\mu_{i,j} = 0$ for $i < j$. Then for $1 \leqslant i < j \leqslant n$, we have $\pi_i(\mathbf{b}_j) = \sum_{l=i}^{j} \mu_{j,l} \mathbf{b}_l^*$. There is a unique Gram–Schmidt decomposition $B = QD\mu$, where $Q = (\mathbf{b}_1^*/\|\mathbf{b}_1^*\|, \ldots, \mathbf{b}_n^*/\|\mathbf{b}_n^*\|)$ is an orthogonal set, $D = \mathrm{Diag}(\|\mathbf{b}_1^*\|, \ldots, \|\mathbf{b}_n^*\|)$, and $\mu = (\mu_{i,j})_{1 \leqslant i,j \leqslant n}^t$ is upper triangular. The triplet (Q, D, μ) is called the GSO of B. The GSO can also be defined for linearly dependent vectors, in which case some of the $\mathbf{b}_i^*$ can be zero.

Duality. For any n-dimensional lattice L with basis $B \in \mathbb{R}^{m \times n}$, the dual lattice of L is defined as $L^\times = \{\mathbf{y} \in \mathrm{span}(B) : \langle \mathbf{x}, \mathbf{y} \rangle \in \mathbb{Z}, \ \forall \ \mathbf{x} \in L\}$; $L^\times$ has basis $B^{-t} \triangleq B(B^tB)^{-1}$, which is called the dual basis of B. The reversed dual basis [4] is defined as $B^{-s} = R_m B^{-t} R_n$ where R_n is the reversed identity matrix: $R_n(i,j) = \delta_{i,n-j+1}$ where $\delta_{i,j}$ denotes Kronecker's symbol. Write $\widehat{\mathbf{b}} := R_m \mathbf{b} = (b_m, b_{m-1}, \ldots, b_1)^t$ where $\mathbf{b} = (b_1, b_2, \ldots, b_m)^t$, and let $\widehat{L} := \{\widehat{\mathbf{x}} : \mathbf{x} \in L\}$. Then $\widehat{L}^\times := \widehat{(L^\times)} = (\widehat{L})^\times$ is the reversed dual lattice of L, which has basis B^{-s}. In lattice reduction, it is more convenient to consider B^{-s} than to consider B^{-t}. The main advantage is that the reversed duality preserves upper triangular, lower triangular, diagonal and orthogonal matrices; it is fully compatible with the matrix product, for example, $(QD\mu)^{-s} = Q^{-s} D^{-s} \mu^{-s}$; the reversed dual lattice $\widehat{L}^\times$ is isometric to the standard dual lattice $L^\times$, and has therefore the same mathematical properties; the projected blocks satisfy $(B^{-s})_{[i,j]} = (B_{[n-j+1,n-i+1]})^{-s}$ for $1 \leqslant i < j \leqslant n$ (see Appendix A for the proof).

Quadratic forms. Any basis $B = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ defines a positive definite quadratic form over $\mathbb{R}^n$ by $q(x_1, \ldots, x_n) = \| \sum_{i=1}^n x_i \mathbf{b}_i \|^2$. Reciprocally, any positive definite quadratic form over $\mathbb{R}^n$ is of this form for at least one basis of some lattice, which follows from classical matrix decompositions.

2.2. *Lattice reduction*

Size reduction. A basis $B = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ is size-reduced if its GSO satisfies: $|\mu_{i,j}| \leqslant \frac{1}{2}$ for all $1 \leqslant j < i \leqslant n$. The basis B is size-reduced starting from $\mathbf{b}_{i_0}$ if $|\mu_{i,j}| \leqslant \frac{1}{2}$ for $i_0 \leqslant i \leqslant n$ and $1 \leqslant j < i$. The basis vector $\mathbf{b}_i$ is size-reduced if $|\mu_{i,j}| \leqslant \frac{1}{2}$ for all $1 \leqslant j < i$.

LLL reduction. A basis $B = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ is LLL-reduced [10] with factor $\varepsilon \in [0, 3)$ if it is size-reduced and every 2×2 block $B_{[i,i+1]}$ satisfies Lovász's condition: $\|\mathbf{b}_i^*\|^2 \leqslant (1 + \varepsilon)(\|\mathbf{b}_{i+1}^*\|^2 + \mu_{i+1,i}^2 \|\mathbf{b}_i^*\|^2)$. This implies Siegel's condition: $\|\mathbf{b}_i^*\|^2 \leqslant (4(1+\varepsilon)/(3-\varepsilon)) \|\mathbf{b}_{i+1}^*\|^2$. Given as input $B \in \mathbb{Z}^{m \times n}$ and $\varepsilon > 0$, the LLL algorithm [10] outputs an LLL-reduced basis in time polynomial in $(\mathrm{size}(B), 1/\varepsilon)$.

HKZ reduction. A basis $B = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ is HKZ-reduced [7, 8] if it is size-reduced and $\mathbf{b}_i^*$ is a shortest non-zero vector of the projected lattice L_i for $i = 1, \ldots, n$.

SVP-oracle. Given as input a symmetric matrix $G \in \mathbb{Q}^{k \times k}$, an SVP-oracle outputs $\mathbf{x} = (x_1, \ldots, x_k)^t \in \mathbb{Z}^k \backslash \{\mathbf{0}\}$ minimizing the positive definite quadratic form $q(x_1, \ldots, x_k) = \mathbf{x}^t G \mathbf{x}$. In the particular case of a lattice with the known GSO (μ, D) of a basis, we may take $G = \mu^t D^2 \mu$. From a theoretical point of view, the best SVP algorithm known is the

https://doi.org/10.1112/S1461157014000333 Published online by Cambridge University Press

Micciancio–Voulgaris algorithm [14], which is a deterministic single exponential time algorithm with 2^{2k} polynomial-time operations and 2^k polynomial-size registers.

SVP reduction. A basis B is SVP-reduced if the first basis vector $\mathbf{b}_1$ satisfies $\|\mathbf{b}_1\| = \lambda_1(L(B))$. There is a natural relaxation: a basis B is $(1+\varepsilon)$-SVP-reduced for $\varepsilon \geqslant 0$ if the first basis vector satisfies $\|\mathbf{b}_1\| \leqslant \sqrt{1+\varepsilon} \cdot \lambda_1(L(B))$.

DSVP reduction. For $\varepsilon \geqslant 0$, a basis B is $(1+\varepsilon)$-DSVP-reduced [5] (where D stands for dual) if the reversed dual basis B^{-s} is $(1+\varepsilon)$-SVP-reduced.

Rankin reduction. A basis B of a lattice L of dimension n is r-Rankin reduced [3] if the first r basis vectors satisfy $\mathrm{vol}(\mathbf{b}_1,\ldots,\mathbf{b}_r) = \sqrt{\gamma_{n,r}(L)}\mathrm{vol}(L)^{r/n}$. There exist r-Rankin reduced bases for any given lattice.

All lattice reduction notions can naturally be adapted to positive definite quadratic forms, using the correspondence described in the previous subsection.

2.3. *The Gama–Nguyen slide reduction algorithm*

A basis B of a lattice L of dimension $n = pk$ is *slide reduced* [5, Definition 1] with blocksize k and factor $\varepsilon \geqslant 0$ if it is size-reduced and satisfies the following two sets of conditions.

(1) Primal conditions: for all $i \in [0, p-1]$, the block $B_{[ik+1,ik+k]}$ is HKZ-reduced.
(2) Dual conditions: for all $i \in [0, p-2]$, the block $B_{[ik+2,ik+k+1]}$ is $(1+\varepsilon)$-DSVP-reduced.

Slide reduced bases achieve Mordell's inequality (1.4).

THEOREM 2.1 [5, Theorem 1]. *A slide reduced basis $B = (\mathbf{b}_1,\ldots,\mathbf{b}_n)$ of a lattice L of dimension $n = pk$ with blocksize k and factor $\varepsilon \geqslant 0$ satisfies*

$$\|\mathbf{b}_1\| \leqslant (\gamma_k\sqrt{1+\varepsilon})^{(n-1)/2(k-1)}\mathrm{vol}(L)^{1/n}, \tag{2.1}$$
$$\|\mathbf{b}_1\| \leqslant (\gamma_k\sqrt{1+\varepsilon})^{(n-k)/(k-1)}\lambda_1(L). \tag{2.2}$$

Gama and Nguyen [5] showed how to compute slide-reduced bases in polynomial time, using a polynomial number of calls to the SVP-oracle in dimension $\leqslant k$ (see Algorithm 1). Algorithm 1 uses two local algorithms which use an SVP-oracle in dimension $\leqslant k$.

- Algorithm 2 [5, Algorithm 2], which performs an HKZ reduction of a given block, using an SVP-oracle in dimension $\leqslant k$. This algorithm calls a Projected-LLL subroutine (see [5, Algorithm 4] for details), which given as input an index $j \in \mathbb{N}$, a factor $\varepsilon > 0$, and a family $B = (\mathbf{b}_1,\ldots,\mathbf{b}_{\ell+1})$ of dimension ℓ where $(\mathbf{b}_1,\ldots,\mathbf{b}_{j-1})$ are linearly independent and $j \leqslant \ell$, outputs a basis $B' = (\mathbf{b}_1,\ldots,\mathbf{b}_{j-1},\mathbf{v}_j,\ldots,\mathbf{v}_\ell)$ of $L(B)$ such that B' is size-reduced starting from $\mathbf{v}_j$ and the projection $(\pi_j(\mathbf{v}_j),\ldots,\pi_j(\mathbf{v}_\ell))$ is an LLL-reduced basis of $L(\pi_j(\mathbf{b}_j),\ldots,\pi_j(\mathbf{b}_{\ell+1}))$. In particular, $B'_{[j,ik+k]}$ is SVP-reduced after Step 4 in Algorithm 2.
- A DSVP-algorithm (see [5, Algorithm 3]), which performs a $(1+\varepsilon)$-DSVP reduction of a given block. Given as input a factor ε, a basis $B \in \mathbb{Z}^{m\times n}$ whose block $B_{[ik+2,ik+k+1]}$ is LLL-reduced, and the GSO matrices $\mu_{[ik+2,ik+k+1]}, D^2_{[ik+2,ik+k+1]}$ of the block $B_{[ik+2,ik+k+1]}$, this algorithm outputs a basis B' such that the block $B'_{[ik+2,ik+k+1]}$ becomes $(1+\varepsilon)$-DSVP-reduced, but none of the basis vectors outside the block are modified.

https://doi.org/10.1112/S1461157014000333 Published online by Cambridge University Press

Algorithm 1 The Gama–Nguyen slide reduction algorithm [5, Algorithm 1]

Input: A blocksize k, a factor $\varepsilon > 0$, and a basis $B = (\mathbf{b}_1, \ldots, \mathbf{b}_n) \in \mathbb{Z}^{m \times n}$ in dimension $n = pk$.
Output: A slide reduced basis of $L(B)$ with blocksize k and factor ε.

while B is not slide reduced **do**
  while one of the primal conditions does not hold **do**
    LLL-reduce B with factor ε and update the GSO matrices $\mu, D^2 \in \mathbb{Q}^{n \times n}$
    for $i \in [0, p-1]$ **do**
      HKZ-reduce $B_{[ik+1, ik+k]}$ using Algorithm 2
    end for
  end while
  for $i \in [0, p-2]$ **do**
    $(1+\varepsilon)$-DSVP-reduce $B_{[ik+2, ik+k+1]}$ using DSVP-algorithm (see [5, Algorithm 3])
  end for
end while
return B.

3. *A new inequality for Rankin's constant*

In this section, we prove Theorem 1.1 and discuss what an algorithmic version of (1.8) would mean.

3.1. *Proof of Theorem 1.1*

We recall a few elementary inequalities on Rankin's constant proved in [17]:

$$\forall n \in \mathbb{N},\ \gamma_{n,n} = 1,\ \gamma_{n,1} = \gamma_n, \tag{3.1}$$

$$\forall n, r \text{ with } r < n,\ \gamma_{n,r} = \gamma_{n,n-r}, \tag{3.2}$$

$$\forall n, r \text{ and } s \in [r, n],\ \gamma_{n,r} \leqslant \gamma_{s,r}(\gamma_{n,s})^{r/s}. \tag{3.3}$$

Rankin's inequality (3.3) and the duality relation (3.2) imply Mordell's inequality $\gamma_{n+1} \leqslant \gamma_n^{n/(n-1)}$ and the generalized Mordell's inequality $\gamma_{n+r,r} \leqslant \gamma_{n,r}^{n/(n-r)}$ (see [12, Corollary 2.8.9]).

Proof of (1.8). Let $1 \leqslant r < k \leqslant n$ where n is of the form $n = pk + \ell r$ with $p \geqslant 1$ and $\ell \geqslant 0$; we want to prove

$$\gamma_{n,r} \leqslant \gamma_{k,r}^{(n-r)/(k-r)} \quad (1.8).$$

The generalized Mordell's inequality $\gamma_{n+r,r} \leqslant \gamma_{n,r}^{n/(n-r)}$ implies

$$\gamma_{pk+\ell r, r} \leqslant \gamma_{pk+\ell r-r, r}^{(pk+\ell r-r)/(pk+\ell r-2r)} \leqslant \ldots \leqslant \gamma_{pk,r}^{(pk+\ell r-r)/(pk-r)},\ \text{for } p \geqslant 1 \text{ and } \ell \geqslant 0.$$

To complete the proof, it suffices to show that $\gamma_{pk,r} \leqslant \gamma_{k,r}^{(pk-r)/(k-r)}$ for $p \geqslant 1$, which is done by induction over p.

Assume that it holds for some p. Let L be a lattice of dimension $n = (p+1)k$. There exists a basis B of L such that B is r-Rankin reduced, $B_{[r+1,n]}$ is $(k-r)$-Rankin reduced and $B_{[k+1,n]}$ is r-Rankin reduced, because $r < r+1$ and $r+1+(k-r-1) < k+1$. Then $B_{[1,k]}$ is r-Rankin reduced, therefore

$$\mathrm{vol}(B_{[1,r]}) \leqslant \gamma_{k,r}^{1/2} \mathrm{vol}(B_{[1,k]})^{r/k}. \tag{3.4}$$

https://doi.org/10.1112/S1461157014000333 Published online by Cambridge University Press

Algorithm 2 HKZ-reduction of the block $B_{[ik+1,ik+k]}$ [5, Algorithm 2]

Input: A factor ε, a basis $B \in \mathbb{Z}^{m\times n}$ whose block $B_{[ik+1,ik+k]}$ is LLL-reduced, and the GSO matrices $\mu_{[ik+1,ik+k]}, D^2_{[ik+1,ik+k]}$ of the block $B_{[ik+1,ik+k]}$.
Output: The block $B_{[ik+1,ik+k]}$ becomes HKZ-reduced, but none of the basis vectors outside the block are modified.

for $j = ik+1$ to $ik+k-1$ **do**
  Call the SVP-oracle on the quadratic form defined by $\mu_{[j,ik+k]}$ and $D^2_{[j,ik+k]}$, let $(\alpha_j,\ldots,\alpha_{ik+k})^t$ be the output linear combination, and compute $\mathbf{b} = \sum_{l=j}^{ik+k} \alpha_l \mathbf{b}_l$
  if $\|\mathbf{b}_j^*\| > \|\pi_j(\mathbf{b})\|$ **then**
    Projected-LLL-reduce [5, Algorithm 4] with factor ε the family $(\mathbf{b}_1,\ldots,\mathbf{b}_{j-1},\mathbf{b},\mathbf{b}_j,\ldots,\mathbf{b}_{ik+k})$ starting at index j
    Store the result of Projected-LLL in column j to $ik+k$ of B, update $\mu_{[ik+1,ik+k]}$ and $D^2_{[ik+1,ik+k]}$
  else
    $j \leftarrow j+1$
  end if
end for
return B.

And $B_{[r+1,k+r]}$ is $(k-r)$-Rankin reduced, therefore $\mathrm{vol}(B_{[r+1,k]}) \leqslant \gamma_{k,r}^{1/2}\mathrm{vol}(B_{[r+1,k+r]})^{(k-r)/k}$. This yields

$$\mathrm{vol}(B_{[r+1,k]}) \leqslant \gamma_{k,r}^{k/(2r)}\mathrm{vol}(B_{[k+1,k+r]})^{(k-r)/r}.$$

Together with (3.4) and $\mathrm{vol}(B_{[1,k]}) = \mathrm{vol}(B_{[1,r]}) \times \mathrm{vol}(B_{[r+1,k]})$, this implies that

$$\mathrm{vol}(B_{[1,r]}) \leqslant \gamma_{k,r}^{k/(k-r)}\mathrm{vol}(B_{[k+1,k+r]}). \tag{3.5}$$

Since $B_{[k+1,n]}$ is r-Rankin reduced, by the inductive hypothesis, we have

$$\mathrm{vol}(B_{[k+1,k+r]}) \leqslant \gamma_{k,r}^{(pk-r)/2(k-r)}\mathrm{vol}(B_{[k+1,n]})^{r/pk}. \tag{3.6}$$

Combining (3.4)–(3.6), we obtain $\mathrm{vol}(B_{[1,r]}) \leqslant \gamma_{k,r}^{(n-r)/2(k-r)}\mathrm{vol}(B_{[1,n]})^{r/n}$, which yields $\gamma_{(p+1)k,r} \leqslant \gamma_{k,r}^{((p+1)k-r)/(k-r)}$. Thus, we proved $\gamma_{pk,r} \leqslant \gamma_{k,r}^{(pk-r)/(k-r)}$ by induction over $p \geqslant 1$, which completes the proof. □

We note that (1.8) implies both $\gamma_{n+r,r} \leqslant \gamma_{n,r}^{n/(n-r)}$ and $\gamma_n \leqslant \gamma_k^{(n-1)/(k-1)}$ as special cases.

3.2. *Approximating the densest sublattice from Theorem 1.1*

Theorem 1.1 upper bounds Rankin's constant $\gamma_{n,r}$ in high dimension using Rankin's constant $\gamma_{k,r}$ in low dimension. This is reminiscent of Mordell's inequality, which upper bounds Hermite's constant γ_n in high dimension using Hermite's constant γ_k in low dimension. Mordell's inequality inspired the Gama–Nguyen reduction algorithm [5] which, given a subroutine to solve SVP in low dimensions $\leqslant k$, approximates SVP in high dimension n, within an approximation factor corresponding to Mordell's inequality. This suggests that similarly, there might exist an algorithm corresponding to Theorem 1.1 for approximating the densest sublattice, which we will show in the next section.

The r-DSP is to Rankin's constant what the SVP is to Hermite's constant. It was introduced by Gama *et al.* [3] under the name '*the smallest volume problem*': given a basis of an

https://doi.org/10.1112/S1461157014000333 Published online by Cambridge University Press

n-dimensional lattice L, r-DSP asks to find an r-dimensional sublattice S of L such that $\mathrm{vol}(S)$ is minimal. This is equivalent to finding a basis $(\mathbf{b}_1, \ldots, \mathbf{b}_n)$ of L minimizing $\mathrm{vol}(\mathbf{b}_1, \ldots, \mathbf{b}_r)$. In its approximate version, given an approximation factor $f(n, r)$, one is asked to find an r-dimensional sublattice S of L such that $\mathrm{vol}(S) \leqslant f(n, r)\mathrm{vol}(L)^{r/n}$.

By analogy with Mordell's inequality, Theorem 1.1 suggests to look for an algorithm to approximate the densest sublattice within a factor essentially $\sqrt{\gamma_{k,r}}^{(n-r)/(k-r)}$, using polynomially many calls to a subroutine solving k-DSP exactly.

Like in the Gama–Nguyen reduction algorithm [5], it is more convenient if the subroutine works with positive definite quadratic forms instead of lattices.

We define an (r, k)-DSP-oracle as any algorithm which, given a positive definite quadratic form represented by a (positive definite) symmetric matrix $G \in \mathbb{Q}^{k\times k}$, outputs an integer matrix $H \in \mathbb{Z}^{k\times r}$ such that

$$\det(H^t G H) = \min_{\substack{\mathbf{x}_1,\ldots,\mathbf{x}_r \in \mathbb{Z}^k \\ \mathrm{vol}(\mathbf{x}_1,\ldots,\mathbf{x}_r)\neq 0}} \det((\mathbf{x}_i^t G \mathbf{x}_j)_{1\leqslant i,j\leqslant r}),$$

and the quadratic form represented by $H^t G H$ is HKZ-reduced.

The first condition corresponds to the classical interpretation of Rankin's constant for positive definite quadratic forms. We explain the HKZ requirement. If $B = (\mathbf{b}_1, \ldots, \mathbf{b}_k)$ is an r-Rankin reduced basis, then the $\mathbf{b}_i$ may be arbitrarily long for $i \geqslant 2$. Indeed, if we replace $\mathbf{b}_i$ by $\mathbf{b}_i + \sum_{j=1}^{i-1} x_j \mathbf{b}_j$, where the x_js are integers, then the basis is still r-Rankin reduced. Forcing the output to be HKZ-reduced allows us to bound the coefficients of H. By minimality and [13, Lemma 3.1], H can be extended to a $k \times k$ unimodular matrix, which is done by Algorithm B.2 (see Appendix B).

It can be checked that the Dadush–Micciancio algorithm [2] is an (r, k)-DSP-oracle, running in time $r^{O(r\cdot k)}$ and space $2^k \mathrm{poly}(k)$ for any $r < k$. By duality, an (r, k)-DSP-oracle implies a $(k - r, k)$-DSP-oracle.

4. *Block–Rankin reduction*

In this section, we present an efficient algorithmic version of Theorem 1.1, which is to (1.8) what the Gama–Nguyen reduction algorithm is to Mordell's inequality. In fact, our algorithm is very similar to the Gama–Nguyen reduction algorithm. First, we introduce a new reduction notion, called block–Rankin reduction, which achieves (1.8) by generalizing Gama–Nguyen's slide reduction. Then we present a deterministic polynomial-time reduction algorithm to output block–Rankin reduced bases, which can be viewed as a higher-dimensional generalization of the Gama–Nguyen reduction algorithm.

The main result of this paper is the following: there exists a deterministic algorithm which, given as input a basis B_0 of an n-dimensional lattice L, a reduction factor $\varepsilon > 0$, an (r, k)-DSP-oracle such that $n = pk + r$ or $n = pk$ for some p, outputs r linearly independent lattice vectors $\mathbf{b}_1, \ldots, \mathbf{b}_r$ such that

$$\mathrm{vol}(\mathbf{b}_1, \ldots, \mathbf{b}_r) < (\sqrt{1+\varepsilon}\gamma_{k,r})^{(n-r)/2(k-r)}\mathrm{vol}(L)^{r/n},$$

where the number of calls to the oracle and the size of the input are polynomial, and apart from the running time of the oracle, the algorithm runs in polynomial time. If the blocksize is $k \leqslant (\log_r \mathrm{poly}(n))/r$ and one selects the Dadush–Micciancio algorithm [2] as the oracle, then the whole algorithm runs in polynomial time.

4.1. *Reduction definitions*

In order to present our block–Rankin reduction, we first generalize the SVP reduction and DSVP reduction used in slide reduction. These generalizations are straightforward.

https://doi.org/10.1112/S1461157014000333 Published online by Cambridge University Press

DEFINITION 1 (Rankin reduction). A basis $B = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ of an n-dimensional lattice L is r-Rankin reduced [3] if its first r vectors satisfy $\mathrm{vol}(\mathbf{b}_1, \ldots, \mathbf{b}_r) = \sqrt{\gamma_{n,r}(L)}\mathrm{vol}(L)^{r/n}$. There is a natural relaxation: a basis B is $(1+\varepsilon, r)$-Rankin reduced for $\varepsilon \geqslant 0$ if its first r vectors satisfy

$$\mathrm{vol}(\mathbf{b}_1, \ldots, \mathbf{b}_r) \leqslant \sqrt{(1+\varepsilon)\gamma_{n,r}(L)}\mathrm{vol}(L)^{r/n}.$$

If the basis is a projected block $B_{[i,j]}$ of size $k = j - i + 1$, this implies that

$$\mathrm{vol}(B_{[i,i+r-1]}) \leqslant \sqrt{(1+\varepsilon)\gamma_{k,r}}\mathrm{vol}(B_{[i,j]})^{r/k}.$$

DEFINITION 2 (Dual-Rankin reduction). A basis B is $(1+\varepsilon, r)$-dual-Rankin reduced if the reversed dual basis B^{-s} is $(1+\varepsilon, r)$-Rankin reduced.

If the basis is a projected block $B_{[i,j]}$ of size $k = j - i + 1$, this implies that

$$\mathrm{vol}(B_{[i,j]}) \leqslant \sqrt{(1+\varepsilon)\gamma_{k,r}}^{k/r}\mathrm{vol}(B_{[j-r+1,j]})^{k/r},$$

which is equivalent to $\mathrm{vol}(B_{[i,j-r]}) \leqslant \sqrt{(1+\varepsilon)\gamma_{k,r}}\mathrm{vol}(B_{[i,j]})^{(k-r)/k}$ because $\mathrm{vol}(B_{[j-r+1,j]}) = \mathrm{vol}(B_{[i,j]})/\mathrm{vol}(B_{[i,j-r]})$.

We now present our higher-dimensional generalization of slide reduction for lattices of dimension n, when n is of the form $n = pk + r$ or $n = pk$ and k is a blocksize satisfying $1 \leqslant r < k$.

DEFINITION 3 (Block–Rankin reduction). A basis B of an n-dimensional lattice L is $(1+\varepsilon, r)$-Block–Rankin reduced with blocksize k where $n = pk + r$ (respectively $n = pk$) with $p \geqslant 1$ and $\varepsilon \geqslant 0$ if it is size-reduced and satisfies the following two sets of conditions.

(i) Primal conditions: the blocks $B_{[ik+1,ik+k]}$ for $i \in [0, p-1]$ are r-Rankin reduced.

(ii) Dual conditions: the blocks $B_{[ik+r+1,ik+k+r]}$ for $i \in [0, p-1]$ (respectively $i \in [0, p-2]$) are $(1+\varepsilon, r)$-dual-Rankin reduced.

Block–Rankin reduction achieves the new Rankin's inequality (1.8), like slide reduction achieved Mordell's inequality.

THEOREM 4.1. *If a basis B of an n-dimensional lattice L is $(1+\varepsilon, r)$-block–Rankin reduced with blocksize k, where n is of the form $n = pk + r$ or $n = pk$ with $p \geqslant 1$ and $\varepsilon \geqslant 0$, then*

$$\mathrm{vol}(B_{[1,r]}) \leqslant (\sqrt{1+\varepsilon}\gamma_{k,r})^{(n-r)/2(k-r)}\mathrm{vol}(L)^{r/n}. \tag{4.1}$$

Proof. The main idea is the same as in Theorem 1.1. We first prove the case $n = pk$. By Definition 3, each primal condition implies

$$\mathrm{vol}(B_{[ik+1,ik+r]}) \leqslant \gamma_{k,r}^{k/2(k-r)}\mathrm{vol}(B_{[ik+r+1,ik+k]})^{r/k-r}, \quad \text{for } 0 \leqslant i \leqslant p-1.$$

Due to the dual conditions, we have

$$\mathrm{vol}(B_{[ik+r+1,ik+k]})^{r/(k-r)} \leqslant ((1+\varepsilon)\gamma_{k,r})^{k/2(k-r)}\mathrm{vol}(B_{[(i+1)k+1,(i+1)k+r]}), \quad \text{for } 0 \leqslant i \leqslant p-2.$$

For each $i \in [0, p-2]$, by multiplying the above two inequalities together, we obtain

$$\mathrm{vol}(B_{[ik+1,ik+r]}) \leqslant (\sqrt{1+\varepsilon}\gamma_{k,r})^{k/(k-r)}\mathrm{vol}(B_{[(i+1)k+1,(i+1)k+r]}).$$

https://doi.org/10.1112/S1461157014000333 Published online by Cambridge University Press

In particular, the first r basis vectors satisfy $\mathrm{vol}(B_{[1,r]}) \leqslant (\sqrt{1+\varepsilon}\gamma_{k,r})^{ik/(k-r)}\mathrm{vol}(B_{[ik+1,ik+r]})$. This yields

$$\mathrm{vol}(B_{[1,r]}) \leqslant (\sqrt{1+\varepsilon}\gamma_{k,r})^{ik/(k-r)+1/2}\mathrm{vol}(B_{[ik+1,ik+k]})^{r/k}, \quad \text{for } 0 \leqslant i \leqslant p-1.$$

The product of the above p inequalities for $i \in [0, p-1]$ gives rise to

$$\mathrm{vol}(B_{[1,r]}) \leqslant (\sqrt{1+\varepsilon}\gamma_{k,r})^{(n-r)/2(k-r)}\mathrm{vol}(L)^{r/n}.$$

For the case $n = pk + r$, it is easy to deduce the desired result by combining the inequalities on $\mathrm{vol}(B_{[1,r]})/\mathrm{vol}(B_{[1,pk]})^{r/pk}$ and $\mathrm{vol}(B_{[1,r]})/\mathrm{vol}(B_{[pk+1,pk+r]})$. This completes the proof. □

The inequality (4.1) is a higher-dimensional generalization of (2.1). Similarly, we will present an algorithm, which is a higher-dimensional generalization of slide reduction.

4.2. *A reduction algorithm*

Our block–Rankin reduction algorithm is Algorithm 3, which has the same structure as Algorithm 1, and uses two local algorithms based on an (r, k)-DSP-oracle:

- Algorithm 4 performs an r-Rankin reduction of a given block;
- Algorithm 5 performs a $(1+\varepsilon, r)$-dual-Rankin reduction of a given block.

Without loss of generality, we may assume that $r \leqslant k/2$ holds in Algorithm 4 and Algorithm 5 by duality. Both Algorithm 4 and Algorithm 5 call Algorithm B.2 (see Appendix B) for extending the output matrix of the (r, k)-DSP-oracle to a unimodular matrix and do not modify any of the basis vectors outside the block: furthermore, the size of the block vectors is always polynomial in the size of the input basis and $1/\varepsilon$. The analysis of the two local algorithms and Algorithm B.2 can be found in Appendices C and D.

We first show correctness of Algorithm 3.

Algorithm 3 Block–Rankin reduction of an integer lattice

Input: Parameters $0 < r < k$, a factor $\varepsilon > 0$, and a basis $B = (\mathbf{b}_1, \ldots, \mathbf{b}_n) \in \mathbb{Z}^{m\times n}$ in dimension $n = pk$.
Output: A $(1+\varepsilon, r)$-block–Rankin reduced basis of $L(B)$ with blocksize k.

```
while B is modified by the loop do
    // ⇔ While B is not block Rankin reduced
    LLL-reduce B with factor ε and update the GSO matrices μ, D² ∈ ℚ^{n×n}
    for i ∈ [0, p − 1] do
        //ensure primal conditions
        r-Rankin reduce B_{[ik+1,ik+k]} using Algorithm 4
    end for
    LLL-reduce B with factor ε and update the GSO matrices μ, D² ∈ ℚ^{n×n}
    for i ∈ [0, p − 2] do
        //ensure dual conditions
        (1 + ε, r)-dual-Rankin reduce B_{[ik+r+1,ik+k+r]} using Algorithm 5
    end for
end while
return B.
```

https://doi.org/10.1112/S1461157014000333 Published online by Cambridge University Press

Algorithm 4 Rankin reduction of the block $B_{[ik+1,ik+k]}$

Input: Parameters $r < k$, a factor ε, a basis $B = (\mathbf{b}_1, \ldots, \mathbf{b}_n) \in \mathbb{Z}^{m \times n}$ whose block $B_{[ik+1,ik+k]}$ is LLL-reduced, and the GSO matrices $\mu_{[ik+1,ik+k]}, D^2_{[ik+1,ik+k]}$ of the block $B_{[ik+1,ik+k]}$.

Output: The block $B_{[ik+1,ik+k]}$ becomes r-Rankin reduced, but none of the basis vectors outside the block are modified.

Call the (r,k)-DSP-oracle on the quadratic form defined by $\mu_{[ik+1,ik+k]}$ and $D^2_{[ik+1,ik+k]}$, let $H \in \mathbb{Z}^{k \times r}$ be the output linear combination matrix, and compute the corresponding minimum $m_r := \det(H^t \mu^t_{[ik+1,ik+k]} D^2_{[ik+1,ik+k]} \mu_{[ik+1,ik+k]} H)$
  // note that $B_{[ik+1,ik+k]}H$ reaches $\gamma_{k,r}(L(B_{[ik+1,ik+k]}))$ and is HKZ-reduced
if $m_r \neq \mathrm{vol}(B_{[ik+1,ik+r]})^2$ **then**
  Apply Algorithm B.2 on H and I_k to obtain a unimodular matrix $U \in \mathbb{Z}^{k \times k}$
    // note that $U_{[1,r]} = H$ and therefore $B_{[ik+1,ik+k]}U$ is r-Rankin reduced
  Compute $(\mathbf{b}_{ik+1}, \ldots, \mathbf{b}_{ik+k}) \leftarrow (\mathbf{b}_{ik+1}, \ldots, \mathbf{b}_{ik+k})U$
end if
return B.

Algorithm 5 Dual-Rankin reduction of the block $B_{[ik+r+1,ik+k+r]}$

Input: Parameters $r < k$, a factor ε, a basis $B \in \mathbb{Z}^{m \times n}$ whose block $B_{[ik+r+1,ik+k+r]}$ is LLL-reduced, and the GSO matrices $\mu_{[ik+r+1,ik+k+r]}, D^2_{[ik+r+1,ik+k+r]}$ of the block $B_{[ik+r+1,ik+k+r]}$.

Output: The block $B_{[ik+r+1,ik+k+r]}$ becomes $(1+\varepsilon, r)$-dual-Rankin reduced, but none of the basis vectors outside the block are modified.

Compute $\mu' \leftarrow \sum_{j=0}^{k-1} N^j$ where $N = I_k - (\mu_{[ik+r+1,ik+k+r]})^s$
Compute $D'^2 \leftarrow \mathrm{Diag}\left(\frac{1}{\|\mathbf{b}^*_{ik+k+r}\|^2}, \ldots, \frac{1}{\|\mathbf{b}^*_{ik+r+1}\|^2}\right)$
  //Note that μ', D' are the GSO matrices of $(B_{[ik+r+1,ik+k+r]})^{-s}$
Call the (r,k)-DSP-oracle on the quadratic form defined by μ' and D'^2, let $H \in \mathbb{Z}^{k \times r}$ be the output linear combination matrix, and compute the corresponding minimum $m_r := \det(H^t \mu'^t D'^2 \mu' H)$
  //Note that $(B_{[ik+r+1,ik+k+r]})^{-s}H$ reaches $\gamma_{k,r}(L((B_{[ik+r+1,ik+k+r]})^{-s}))$ and is HKZ-reduced
if $(B_{[ik+r+1,ik+k+r]})^{-s}$ is not $(1+\varepsilon, r)$-Rankin reduced, that is, $(1+\varepsilon)m_r \mathrm{vol}(B_{[ik+k+1,ik+k+r]})^2 < 1$, **then**
  Apply Algorithm B.2 on H and I_k to obtain a unimodular matrix $U \in \mathbb{Z}^{k \times k}$
    //Note that $U_{[1,r]} = H$ and therefore $(B_{[ik+r+1,ik+k+r]})^{-s}U$ is r-Rankin reduced
  Compute $U^{-s} \leftarrow R_k U^{-t} R_k$
  Compute $(\mathbf{b}_{ik+r+1}, \ldots, \mathbf{b}_{ik+k+r}) \leftarrow (\mathbf{b}_{ik+r+1}, \ldots, \mathbf{b}_{ik+k+r})U^{-s}$
end if
return B.

THEOREM 4.2. *For all $\varepsilon \geqslant 0$, Algorithm 3 terminates, and outputs a $(1+\varepsilon, r)$-block–Rankin reduced basis with blocksize k.*

Proof. We only consider the case $n = pk$ (the case $n = pk + r$ is similar). Let $B_0 = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ be the input integer lattice basis and let B denote the current basis during the execution. Following the standard analysis of LLL [10], we consider the following integral

https://doi.org/10.1112/S1461157014000333 Published online by Cambridge University Press

potential

$$P(B) = \prod_{j=1}^{p-1} \mathrm{vol}(B_{[1,jk]})^2 = \prod_{j=0}^{p-2} \mathrm{vol}(B_{[jk+1,jk+k]})^{2(p-1-j)} \in \mathbb{Z}.$$

Then, the initial potential satisfies $\log P(B_0) \leqslant n(p-1) \cdot \log \|B_0\|$ and every operation in Algorithm 3 either preserves or strictly decreases $P(B)$. More precisely, if each $(1+\varepsilon, r)$-dual-Rankin reduction modifies the block $B_{[ik+r+1,ik+k+r]}$ for some $i \in [0; p-2]$, or each special swap of LLL (in lines 3 and 8) between two indexes $(ik+k, ik+k+1)$ occurs, then the integer number $P(B)$ is reduced by a factor $< 1/(1+\varepsilon)$. Therefore, there is a bounded number of such $(1+\varepsilon, r)$-dual-Rankin reductions and special swaps even if $\varepsilon = 0$. The operations which preserve $P(B)$ cannot modify the basis indefinitely. Hence, Algorithm 3 terminates for all $\varepsilon \geqslant 0$.

It is easy to see that Algorithm 3 finally outputs a $(1+\varepsilon, r)$-block Rankin reduced basis with blocksize k. Indeed, the LLL reduction ensures that the output basis is size-reduced. The use of Algorithm 4 ensures that each primal condition is satisfied, while the use of Algorithm 5 ensures that each dual condition is satisfied. This completes the proof. □

Algorithm 3 follows virtually the same structure as Algorithm 1, and Algorithm 5 follows the same principle as the DSVP-algorithm [**5**, Algorithm 3]. Algorithm B.2 as a subroutine in both Algorithm 4 and Algorithm 5 only performs rational operations on the integer matrices consisting of k-dimensional vectors instead of on the basis consisting of m-dimensional vectors. However, we need to bound the size of the matrices that appear in steps 4–8 of Algorithm 4 and in steps 7–12 of Algorithm 5 (see Appendices C and D for details).

4.3. *Complexity analysis*

We now show that Algorithm 3 is in fact polynomial, in the same sense as blockwise reduction algorithms [**3**, **5**, **19**] to approximate SVP. More precisely, we have the following result (see Appendices C and D for the technical lemmas).

THEOREM 4.3. *Given as input parameters $1 \leqslant r < k$, a basis $B_0 \in \mathbb{Z}^{m \times n}$ of dimension $n = pk + r$ or $n = pk$, and a reduction factor $\varepsilon \in (0, 1]$, then any execution of Algorithm 3 satisfies the following.*

(i) *The number of calls to the (r, k)-DSP-oracle is $O(np^2 \log \|B_0\|/\varepsilon)$.*
(ii) *The size of the coefficients passed to the (r, k)-DSP-oracle is polynomial in $\mathrm{size}(B_0)$.*
(iii) *Apart from the calls to the (r, k)-DSP-oracle, the algorithm only performs arithmetic operations on rational numbers such that the number of arithmetic operations is polynomial in $(\mathrm{size}(B_0), 1/\varepsilon)$, and the size of the rational numbers remains polynomial in $\mathrm{size}(B_0)$.*

We consider again the integral potential $P(B)$ defined in § 4.2. Since $\varepsilon > 0$, the number of calls to $(1+\varepsilon, r)$-dual-Rankin reduction subroutine and special swap is at most $(\log P(B_0))/\log(1+\varepsilon)$. That is, Algorithm 3 terminates after at most $(\log P(B_0))/\log(1+\varepsilon)$ loops, and every loop has pr-Rankin reductions and $p-1$ $(1+\varepsilon, r)$-dual-Rankin reductions (for the case $n = pk$). Therefore the total number of calls to the (r, k)-DSP-oracle is at most $(2np^2 \log \|B_0\|)/\log(1+\varepsilon)$.

It remains to upper bound the size of the variables and the cost of the operations (apart from oracle queries) used by Algorithm 3. The main issue is to upper bound $\|B\|$ during the algorithm, with respect to $\|B_0\|$.

https://doi.org/10.1112/S1461157014000333 Published online by Cambridge University Press

First, consider the current basis B, right after Steps 3 or 8. Then B is LLL-reduced. By classical properties of LLL-reduced bases [10], this implies (for $\varepsilon \in (0,1]$) that $\|B\| \leqslant 2^n \lambda_n(L)$, where $\lambda_n()$ denotes the nth minimum, and L is the lattice spanned by B_0. Notice that $\lambda_n(L) \leqslant \|B_0\|$ because B_0 is a basis. Hence, after each Step 3 or 8, we have

$$\|B\| \leqslant 2^n \|B_0\|,$$

which implies that size(B) is polynomial in size(B_0).

Now, the only other operations which may increase $\|B\|$ are Steps 6 and 11, where Algorithms 4 and 5 are called: these algorithms apply some unimodular transformation U locally on B. Fortunately, we are able to suitably upper bound $\|U\|$ (see Lemmas C.2 and C.3 in Appendix C), that is, size(U) is polynomial in size(B): this is because our oracles have an HKZ constraint, and the way we extend a primitive set to a basis cannot increase the norms much (see Theorem B.2). Hence, before each Step 3 or 8, size(B) is polynomial in size(B_0) (see Lemmas D.1 and D.3), and therefore the LLL reductions run in time polynomial in size(B_0).

We proved that size(B) is always polynomial in size(B_0) during the whole algorithm: by the same arguments as in the classical analysis of the LLL algorithm, it follows that all the rational numbers used by the algorithm (such as Gram–Schmidt coefficients and the oracle inputs) have size polynomial in size(B_0) and can be computed in time polynomial in size(B_0).

This completes the proof of Theorem 4.3. And we deduce that the running time of Algorithm 3 can be upper bounded by a polynomial factor times the cost DSP(r,k) of the (r,k)-DSP-oracle. Hence, Algorithm 3 is polynomial in the same sense as Schnorr's algorithm [19] and its transference variant [3], and slide reduction [5]. In particular, if $k \leqslant (\log_r \mathrm{poly}(n))/r$ and we use the Dadush–Micciancio algorithm [2] as the (r,k)-DSP-oracle, then Algorithm 3 runs in polynomial time.

Appendix A. *Properties of projected blocks*

PROPOSITION A.1. *Let $B = (\mathbf{b}_1, \ldots, \mathbf{b}_n) \in \mathbb{R}^{m \times n}$ be an n-dimensional lattice basis. Then the projected blocks satisfy $(B^{-s})_{[i,j]} = (B_{[n-j+1,n-i+1]})^{-s}$ for $1 \leqslant i < j \leqslant n$.*

Proof. Since $(B_{[n-j+1,n-i+1]})^{-s} = R_m (B_{[n-j+1,n-i+1]})^{-t} R_{j-i+1}$, it suffices to prove that $R_m (B^{-s})_{[i,j]} R_{j-i+1} = (B_{[n-j+1,n-i+1]})^{-t}$, which is equivalent to

$$\mathrm{span}(R_m (B^{-s})_{[i,j]} R_{j-i+1}) = \mathrm{span}(B_{[n-j+1,n-i+1]}), \tag{A.1}$$

$$R_m (B^{-s})_{[i,j]} R_{j-i+1} (B_{[n-j+1,n-i+1]})^t = I_m. \tag{A.2}$$

Let $\mathbf{b}_1^*, \ldots, \mathbf{b}_n^*$ denote the Gram–Schmidt orthogonalization of $\mathbf{b}_1, \ldots, \mathbf{b}_n$, and let $\mathbf{d}_n, \ldots, \mathbf{d}_1$ be the dual basis of $\mathbf{b}_1, \ldots, \mathbf{b}_n$ in reverse order with Gram–Schmidt orthogonalization $\mathbf{d}_n^*, \ldots, \mathbf{d}_1^*$ (using this order). Then $B^{-s} = R_m B^{-t} R_n = (\widehat{\mathbf{d}}_n, \ldots, \widehat{\mathbf{d}}_1)$ and $\mathbf{b}_l^* = \mathbf{d}_l^* / \|\mathbf{d}_l^*\|^2$ for $1 \leqslant l \leqslant n$ (see [9, 18]). We define $\tau : \mathrm{span}(\mathbf{d}_n, \ldots, \mathbf{d}_1) \mapsto \mathrm{span}(\mathbf{d}_n, \ldots, \mathbf{d}_{n-i+2})^\perp$, then

$$R_m (B^{-s})_{[i,j]} R_{j-i+1} = (\tau(\mathbf{d}_{n-j+1}), \ldots, \tau(\mathbf{d}_{n-i+1})).$$

Note that $\mathrm{span}(\tau(\mathbf{d}_{n-j+1}), \ldots, \tau(\mathbf{d}_{n-i+1})) = \mathrm{span}(\mathbf{d}_{n-j+1}^*, \ldots, \mathbf{d}_{n-i+1}^*)$ and $\mathrm{span}(B_{[n-j+1,n-i+1]}) = \mathrm{span}(\mathbf{b}_{n-j+1}^*, \ldots, \mathbf{b}_{n-i+1}^*)$, these imply that (A.1) holds. For $i \leqslant s,t \leqslant j$, since $\langle \mathbf{d}_{n-s+1} - \tau(\mathbf{d}_{n-s+1}), \pi_{n-j+1}(\mathbf{b}_{n-t+1}) \rangle = 0$ and $\langle \mathbf{d}_{n-s+1}, \mathbf{b}_{n-t+1} - \pi_{n-j+1}(\mathbf{b}_{n-t+1}) \rangle = 0$, then

$$\langle \tau(\mathbf{d}_{n-s+1}), \pi_{n-j+1}(\mathbf{b}_{n-t+1}) \rangle = \langle \mathbf{d}_{n-s+1}, \mathbf{b}_{n-t+1} \rangle = \delta_{s,t}$$

which proves (A.2). This completes the proof. □

https://doi.org/10.1112/S1461157014000333 Published online by Cambridge University Press

Algorithm B.1 Computing a basis of a lattice given by generators

Input: An integer matrix $B = (\mathbf{b}_1, \ldots, \mathbf{b}_n) \in \mathbb{Z}^{m \times n}$ such that $\mathbf{b}_1 \neq \mathbf{0}$. The columns of B might be linearly dependent.

Output: A basis of the lattice $L = L(\mathbf{b}_1, \ldots, \mathbf{b}_n)$ spanned by the columns of B.

$z \leftarrow 0$, $j \leftarrow 2$
while $j \leqslant n$ **do**
  if $\mathbf{b}_j^* \neq \mathbf{0}$ **then**
    $j \leftarrow j + 1$
  else
    Size-reduce $\mathbf{b}_j$ {with respect to the previous vectors $(\mathbf{b}_{z+1}, \ldots, \mathbf{b}_{j-1})$}
    if $\mathbf{b}_j = 0$ **then**
      for $i = j$ downto $z + 2$ {move $\mathbf{b}_j$ to the front, and shift the rest} **do**
        $\mathbf{b}_i \leftarrow \mathbf{b}_{i-1}$
      end for
      $\mathbf{b}_{z+1} \leftarrow 0$, $j \leftarrow j + 1$; $z \leftarrow z + 1$ {we have found one more zero vector}
    else
      Swap $\mathbf{b}_{j-1}$ and $\mathbf{b}_j$
      $j \leftarrow \max\{z + 2, j - 1\}$
    end if
  end if
end while
return $(\mathbf{b}_{z+1}, \ldots, \mathbf{b}_n)$

Appendix B. *Extending a primitive set to a lattice basis*

Our block–Rankin reduction algorithm requires a subroutine to extend a given primitive set of a lattice L to a basis of L, in order to use the DSP-oracle output (for both the primal and dual blocks): similarly, the Gama–Nguyen algorithm [5] required the simplest case where the primitive set is a single primitive vector to be tackled, which was done in Algorithms 3 and 4 of [5] (for both the primal and dual blocks). This general problem was solved by Magliveras *et al.* [11], by reducing the problem to the computation of a Hermite normal form and a matrix inverse. However, for the analysis of the reduction algorithm, it is also convenient to have good bounds on the output basis, which are not provided in [11].

In this section, we present a (simple) direct polynomial-time algorithm (see Algorithm B.2) to extend a primitive set, together with good bounds on the output basis. This algorithm simply applies a special algorithm (Algorithm B.1) which, given (possibly linear dependent) integer vectors, outputs a basis of the lattice spanned by the input vectors, without increasing the norms much. This special algorithm is essentially a 'cheap' version of the modified LLL algorithm for linearly dependent vectors, where we ignore Lovász' conditions: apparently, the same idea is used in the `image()` subroutine of Shoup's NTL library, and the algorithms might be identical.

We first describe our special algorithm to compute a lattice basis from generators: this is Algorithm B.1, which iteratively modifies the sequence $(\mathbf{b}_1, \ldots, \mathbf{b}_n)$ of generators given as input. It uses two indices j and z such that: $\mathbf{b}_1, \ldots, \mathbf{b}_z$ are zero vectors, but $\mathbf{b}_{z+1}, \ldots, \mathbf{b}_{j-1}$ are linearly independent. At the end of the algorithm, $j - 1 = n$, which implies that $(\mathbf{b}_{z+1}, \ldots, \mathbf{b}_n)$ is a basis. It can be checked that if the input matrix is actually a basis, then the algorithm returns the same basis.

https://doi.org/10.1112/S1461157014000333 Published online by Cambridge University Press

The main result on Algorithm B.1 is the following.

THEOREM B.1. *Given as input an $m \times n$ integer matrix B_0, Algorithm B.1 runs in time polynomial in the size of B_0, and outputs a basis B of the lattice L spanned by the n columns of B_0 such that $\|B\| \leqslant \sqrt{\dim(L)} \times \|B_0\|$.*

Proof. Let L be the lattice spanned by the n columns of the input matrix B_0. It can easily be checked that during the algorithm, we always have $L = L(\mathbf{b}_1, \ldots, \mathbf{b}_n)$, that is, the b_i's generate L.

We first prove the correctness of the algorithm. To do so, note that the following invariants hold at Steps 3–16:

(i) $j \geqslant z + 2$;
(ii) if $1 \leqslant i \leqslant z$, then $\mathbf{b}_i = 0$;
(iii) if $z < i < j$, then $\mathbf{b}_i^* \neq 0$.

Property (i) is obvious. Property (ii) holds initially when $z = 0$, and the only operation which increases z is Step 11, where a zero vector is inserted at index $z + 1$. The third invariant (iii) also holds initially when $j = 2$ because the input $\mathbf{b}_1 \neq 0$: the only operations which can change j are Steps 4 and 14. Step 4 preserves (iii) by definition. For Step 14, $\mathbf{b}_{j-1}^*$ and $\mathbf{b}_j^*$ have been changed: either j decreases which preserves (iii) no matter, or $j = z + 2$, which implies that $\mathbf{b}_{z+1}$ has been replaced by $\mathbf{b}_j \neq 0$, and therefore $\mathbf{b}_{z+1}^* = \mathbf{b}_j \neq 0$ which proves (iii). Now if the algorithm terminates, then $j = n + 1$: (iii) implies that $\mathbf{b}_{z+1}, \ldots, \mathbf{b}_n$ are linearly independent, and together with (ii), it implies that they form a basis of L: hence, the output returned by Step 18 is indeed a basis of L.

Next, we study the running time of the algorithm. We consider the same potential $D = D_L \times D_R$ as in the analysis of the modified LLL algorithm for linearly dependent vectors (see [**16**, § 6]), where

$$D_L = \prod_{i=1}^{\dim(L)} d_i \quad \text{and} \quad D_R = \prod_{\|\mathbf{b}_i^*\|=0} 2^i,$$

with d_i the product of the i first non-zero $\|\mathbf{b}_i^*\|^2$. It is known that the d_i are strictly positive integers, and so are therefore D_L, D_R and D.

We use D to upper bound the number of iterations of the `while` loop, and we study the evolution of $\|B^*\|$. Initially, we have

$$D \leqslant \|B_0\|^{2\dim(L)^2} 2^{n^2} \quad \text{and} \quad \|B^*\| \leqslant \|B_0\|.$$

Since size-reduction does not change the Gram–Schmidt vectors, the only operations which can change D or $\|B^*\|$ are Steps 8–11 (`then`) and Step 13.

Steps 8–11 do not change D_L, but they decrease D_R by a multiplicative factor $\geqslant 2$: indeed, (i) implies that at least one non-zero vector is moved, and a zero vector is inserted at index $z + 1$. Thus, Steps 8–11 decrease D by a multiplicative factor $\geqslant 2$, and they do not change $\|B^*\|$.

Now, consider the effect of Step 13 on the Gram–Schmidt vectors.

- The new $\mathbf{b}_{j-1}^*$ is $\mu_{j,j-1}\mathbf{b}_{j-1}^*$ where $|\mu_{j,j-1}| \leqslant \frac{1}{2}$ by size-reduction.
- The new $\mathbf{b}_j^*$ is either zero if $\mu_{j,j-1} \neq 0$, or $\mathbf{b}_{j-1}^*$ otherwise.
- All the other Gram–Schmidt vectors are preserved.

This implies that Step 13 cannot increase $\|B^*\|$, and there are two cases.

- If $\mu_{j,j-1} \neq 0$, then D_L decreases by a multiplicative factor $\geqslant 4$ (because $0 < \mu_{j,j-1}^2 \leqslant \frac{1}{4}$) and D_R remains.
- Otherwise $\mu_{j,j-1} = 0$, which preserves D_L, but decreases D_R by a multiplicative factor 2.

https://doi.org/10.1112/S1461157014000333 Published online by Cambridge University Press

To summarize, $\|B^*\|$ never increases, and D decreases by a multiplicative factor $\geqslant 2$ at each Step 5. This proves that the number of loop iterations is upper bounded by $O(\log \|B_0\|^{2\dim(L)^2} 2^{n^2})$, which is polynomial in size(B_0). It remains to bound the cost of each iteration.

The only operation which can change $\|B\|$ is the size-reduction (Step 6): at this point, we know that $\mathbf{b}_j^* = 0$, therefore $\mathbf{b}_j = \sum_{i=1}^{j-1} \mu_{j,i}\mathbf{b}_i^*$ where $|\mu_{j,i}| \leqslant \frac{1}{2}$ and the $\mathbf{b}_i^*$ are pairwise orthogonal, thus $\|\mathbf{b}_j\| \leqslant \|B^*\|_F/2$ and $\|\mathbf{b}_j\| \leqslant \|B^*\|\sqrt{\dim(L)}/2$ because at most $\dim(L)$ coefficients $\mu_{j,i}$ are non-zero. Hence, we always have

$$\|B\| \leqslant \sqrt{\dim(L)} \times \|B_0\|.$$

This shows that the size of the $\mathbf{b}_i$ is always polynomial in size(B_0), and it follows that each loop iteration runs in time polynomial in size(B_0). □

We now explain how Algorithm B.1 gives rise to Algorithm B.2 for extending a primitive set to a lattice basis.

Algorithm B.2 Extending a primitive set to a lattice basis

Input: A primitive set $P = (\mathbf{p}_1, \ldots, \mathbf{p}_r)$ of a lattice $L \in \mathbb{Z}^m$, and a basis B_0 of L.
Output: A basis B of L such that $B_{[1,r]} = P$.
1: **return** the output of Algorithm B.1 on the column-concatenation of P and B_0.

Our main result is as follows.

THEOREM B.2. *Given as input a primitive set $P = (\mathbf{p}_1, \ldots, \mathbf{p}_r)$ of a lattice $L \subseteq \mathbb{Z}^m$, and a basis B_0 of L, Algorithm B.2 runs in time polynomial in the size of B_0, and outputs a basis B of the lattice L such that $B_{[1,r]} = P$ and $\|B\| \leqslant \sqrt{\dim(L)} \times \max(\|P\|, \|B_0\|)$.*

Proof. From Theorem B.1, the only thing to prove is $B_{[1,r]} = P$. This certainly holds at the start of Algorithm B.1. We claim that the following invariant holds at Steps 3–16: if $z < i \leqslant z + r$, then $\mathbf{b}_i = \mathbf{p}_{i-z}$. Now, the only way to break the invariant is if Step 13 occurs when $j = z + r + 1$. So let us assume that $j = z + r + 1$. If $\mathbf{b}_j^* \neq 0$, then Step 13 does not occur. Otherwise, $\mathbf{b}_j \in \text{span}(\mathbf{b}_1, \ldots, \mathbf{b}_{j-1}) = \text{span}(\mathbf{p}_1, \ldots, \mathbf{p}_r)$. However, $\text{span}(\mathbf{p}_1, \ldots, \mathbf{p}_r) \bigcap L = L(\mathbf{p}_1, \ldots, \mathbf{p}_r)$ because $P = (\mathbf{p}_1, \ldots, \mathbf{p}_r)$ is a primitive set of L. Therefore $\mathbf{b}_j \in L(\mathbf{p}_1, \ldots, \mathbf{p}_r)$, which implies that the size-reduction (Step 6) makes $\mathbf{b}_j$ equal to zero, and therefore Step 13 does not occur. Hence, we proved that Step 13 never occurs when $j = z + r + 1$, and the invariant always holds. □

Algorithm B.2 can be natively used to extend the output matrix H of the (r, k)-DSP-oracle to a unimodular matrix: simply feed H and I_k to Algorithm B.2, which returns a basis U of $\mathbb{Z}^k$ such that $U_{[1,r]} = H$ and $\|U\| \leqslant \sqrt{k}\|H\|$.

Appendix C. *Bounding the size of unimodular transformations*

We bound the size of the matrices of our algorithm, using the notation $\|\cdot\|_F$ and $\|\cdot\|$.

LEMMA C.1. *The following properties hold for the Frobenius norm:*
(i) $\|A_1 A_2\|_F \leqslant \|A_1\|_F \|A_2\|_F$ *for* $A_1 \in \mathbb{R}^{m\times n}$ *and* $A_2 \in \mathbb{R}^{n\times k}$;
(ii) $\|A\| \leqslant \|A\|_F \leqslant \sqrt{k}\|A\|$ *for* $A \in \mathbb{R}^{m\times k}$;

https://doi.org/10.1112/S1461157014000333 Published online by Cambridge University Press

(iii) $\|QA\|_F = \|A\|_F$ *for* $A \in \mathbb{R}^{k\times k}$, *where* $Q \in \mathbb{R}^{m\times k}$ *satisfies* $Q^tQ = I_k$;
(iv) $\|\widetilde{A}\|_F \leqslant k\|A\|_F^{k-1}$ *for* $A \in \mathbb{R}^{k\times k}$ *and* $k \geqslant 2$, *where* $\widetilde{A} = \det(A)A^{-1}$ *denotes the adjunct matrix of* A.

Proof. Properties (i)–(iii) are obvious. Let $\widetilde{A} = (\tilde{a}_{i,j})_{1\leqslant i,j\leqslant k}$, then $\|\widetilde{A}\|_F = \sqrt{\sum_{i,j}\tilde{a}_{i,j}^2} \leqslant k\max_{i,j}|\tilde{a}_{i,j}|$. Note that $\tilde{a}_{i,j}$ is the (i,j)th cofactor of A. From Hadamard's inequality $\det(B) \leqslant \prod_{i=1}^n \|\mathbf{b}_i\|$ for $B = (\mathbf{b}_i)_{1\leqslant i\leqslant n}$ and the mean inequality $\prod_{i=1}^n x_i \leqslant ((\sum_{i=1}^n x_i^2)/n)^{n/2}$, we deduce that $|\tilde{a}_{i,j}| \leqslant (\|A\|_F^2/(k-1))^{(k-1)/2}$. This proves Property (iv) since $k \geqslant 2$. □

LEMMA C.2. *Let* $B \in \mathbb{Z}^{m\times n}$ *be the input basis of Algorithm 4 and* $\alpha = 4(1+\varepsilon)/(3-\varepsilon)$. *If* $H \in \mathbb{Z}^{k\times r}$ *and* $U \in \mathbb{Z}^{k\times k}$ *are the matrices defined in Line 1 and Line 4 of Algorithm 4 respectively, then*

$$\|H\|_F \leqslant \alpha^{k-1}k^{k+1}\|B\|^{2n},$$
$$\|U\|_F \leqslant \alpha^{k-1}k^{k+2}\|B\|^{2n}.$$

Proof. We use the notation of Algorithm 4 and let $K = B_{[ik+1,ik+k]}$. Note that there exists a $k\times k$ unimodular matrix V such that KV is LLL-reduced (with factor ε) and $V_{[1,r]} = H$, therefore it suffices to bound $\|V\|_F$. Let $W = KV$, then $V = (K^tK)^{-1}K^tW$. By Lemma C.1(i), we obtain

$$\|V\|_F \leqslant \|(K^tK)^{-1}\|_F\|K\|_F\|W\|_F.$$

By Lemma C.1(ii), we have

$$\|K\|_F \leqslant \sqrt{k}\|B\|. \tag{C.1}$$

Since both W and K are LLL-reduced, $\|W\| \leqslant \alpha^{k-1}\|K\|$. Then, by Lemma C.1(ii), we obtain

$$\|W\|_F \leqslant \alpha^{k-1}\sqrt{k}\|B\|.$$

Since $(K^tK)^{-1} = (\det(K^tK))^{-1}\widetilde{K^tK}$, Lemma C.1(iv) implies that $\|(K^tK)^{-1}\|_F \leqslant k(\det(K^tK))^{-1}\|K^tK\|_F^{k-1}$. Using $\det(K^tK) = \mathrm{vol}(B_{[ik+1,ik+k]})^2 = \mathrm{vol}(B_{[1,ik+k]})^2/\mathrm{vol}(B_{[1,ik]})^2$ and $\mathrm{vol}(B_{[1,j]})^2 \in \mathbb{Z}$ for $j \in [1,n]$, we have

$$\|(K^tK)^{-1}\|_F \leqslant k\|B\|^{2ik}\|K\|_F^{2(k-1)} \leqslant k^k\|B\|^{2(n-1)}. \tag{C.2}$$

Putting the above together, we obtain that $\|V\|_F \leqslant \alpha^{k-1}k^{k+1}\|B\|^{2n}$. Since $\|H\|_F \leqslant \|V\|_F$ and $\|U\|_F \leqslant k\|H\|_F$ (by Theorem B.2 and Lemma C.1(ii)), the conclusion follows. □

LEMMA C.3. *Let* $B \in \mathbb{Z}^{m\times n}$ *be the input basis of Algorithm 5 and* $\alpha = 4(1+\varepsilon)/(3-\varepsilon)$. *If* $H \in \mathbb{Z}^{k\times r}$ *and* $U \in \mathbb{Z}^{k\times k}$ *are the matrices defined in Line 4 and Line 7 of Algorithm 5 respectively, then*

$$\|H\|_F \leqslant \alpha^{k-1}k^{k+3}\|B\|^{3n},$$
$$\|U\|_F \leqslant \alpha^{k-1}k^{k+4}\|B\|^{3n},$$
$$\|U^{-s}\|_F \leqslant \alpha^{(k-1)^2}k^{k^2+3k-3}\|B\|^{3n(k-1)}.$$

Proof. We use the notation of Algorithm 5 and let $M = (B_{[ik+r+1,ik+k+r]})^{-s}$. Note that there exists a $k\times k$ unimodular matrix V such that $J := MV$ is LLL-reduced (with factor ε) and $V_{[1,r]} = H$, therefore it suffices to bound $\|V\|_F$. Since $V = (M^tM)^{-1}M^tJ$, then

$$\|V\|_F \leqslant \|(M^tM)^{-1}\|_F\|M\|_F\|J\|_F.$$

https://doi.org/10.1112/S1461157014000333 Published online by Cambridge University Press

We first bound $\|J\|_F$. Clearly, $\|J\|_F \leqslant \sqrt{k}\|J\|$. Since J is LLL-reduced, then $\|J\| \leqslant \sqrt{\alpha}^{k-1}\lambda_k(L(J))$ (see [10]) where the $\lambda_j(L)$ denote the successive minima of L. Using the transference theorem $\lambda_k(L(J))\lambda_1(L(J)^\times) \leqslant k$ (see [1]), we obtain

$$\|J\| \leqslant \sqrt{\alpha}^{k-1} k \lambda_1(L(J)^\times)^{-1}.$$

Note that the lattice $L(J)^\times$ is isometric to the projected lattice $L(B_{[ik+r+1,ik+k+r]})$ where $B_{[ik+r+1,ik+k+r]}$ is LLL-reduced, then

$$\lambda_1(L(J)^\times)^{-2} = \lambda_1(L(B_{[ik+r+1,ik+k+r]}))^{-2} \leqslant \frac{\alpha^{k-1}}{\|\mathbf{b}^*_{ik+r+1}\|^2} = \frac{\alpha^{k-1}\mathrm{vol}(B_{[1,ik+r]})^2}{\mathrm{vol}(B_{[1,ik+r+1]})^2}.$$

Since $\mathrm{vol}(B_{[1,ik+r]})^2 \leqslant \|B\|^{ik+r}$ and $\mathrm{vol}(B_{[1,ik+r+1]})^2 \in \mathbb{Z}$, then $\lambda_1(L(J)^\times)^{-2} \leqslant \alpha^{k-1}\|B\|^{2(n-k)}$. Therefore,

$$\|J\|_F \leqslant \alpha^{k-1} k^{\frac{3}{2}} \|B\|^{n-k}.$$

By Lemma C.1(iii), we have

$$\|M\|_F = \|(B_{[ik+r+1,ik+k+r]})^{-t}\|_F = \|B_{[ik+r+1,ik+k+r]}((B_{[ik+r+1,ik+k+r]})^t B_{[ik+r+1,ik+k+r]})^{-1}\|_F.$$

Hence, applying (C.1) and (C.2) on the block $B_{[ik+r+1,ik+k+r]}$, we obtain

$$\|M\|_F \leqslant k^{k+1/2}\|B\|^{2n-1}.$$

Since $M = (B_{[ik+r+1,ik+k+r]})^{-s}$, then $(M^tM)^{-1} = ((B_{[ik+r+1,ik+k+r]})^t B_{[ik+r+1,ik+k+r]})^s$. Thus

$$\|(M^tM)^{-1}\|_F = \|(B_{[ik+r+1,ik+k+r]})^t B_{[ik+r+1,ik+k+r]}\|_F \leqslant \|B_{[ik+r+1,ik+k+r]}\|_F^2 \leqslant k\|B\|^2.$$

Putting the above together, we obtain

$$\|H\|_F \leqslant \|V\|_F \leqslant \alpha^{k-1}k^{k+3}\|B\|^{3n}.$$

Using Theorem B.2 and Lemma C.1(ii), we have

$$\|U\|_F \leqslant k\|H\|_F \leqslant \alpha^{k-1}k^{k+4}\|B\|^{3n}.$$

Lemma C.1(iv) implies $\|U^{-s}\|_F = \|U^{-1}\|_F = \|\widetilde{U}\|_F \leqslant k\|U\|_F^{k-1}$, and then the last assertion follows. This completes the proof. □

Appendix D. *Analysis of Algorithms 4 and 5*

LEMMA D.1. *Let $B^{(a)}$ and $B^{(b)}$ denote the input and output bases of Algorithm 4 on the projected block $B^{(a)}_{[ik+1,ik+k]}$ respectively, and $\alpha = 4(1+\varepsilon)/(3-\varepsilon)$. Then $L(B^{(a)}) = L(B^{(b)})$, $B^{(b)}_{[ik+1,ik+k]}$ is r-Rankin reduced and*

$$\|B^{(b)}\| \leqslant \alpha^{n^2+k} k^{k+3} \|B_0\|^{2n+1}$$

where $B_0 \in \mathbb{Z}^{m\times n}$ denotes the input of Algorithm 3.

https://doi.org/10.1112/S1461157014000333 Published online by Cambridge University Press

Proof. We use the notation of Algorithm 4. If $m_r = \mathrm{vol}(B^{(a)}_{[ik+1,ik+r]})^2$ holds, it is trivial. We assume that Algorithm 4 executes Steps 4–6, then Algorithm 4 runs Algorithm B.2 on H and I_k to output U. Since U is unimodular, $L(B^{(a)}) = L(B^{(b)})$. Since $U_{[1,r]} = H$ and $B^{(b)}_{[ik+1,ik+k]} = B^{(a)}_{[ik+1,ik+k]}U$, then $B^{(b)}_{[ik+1,ik+k]}$ is r-Rankin reduced.

We write $K^{(a)} = (\mathbf{b}^{(a)}_{ik+1}, \ldots, \mathbf{b}^{(a)}_{ik+k})$. Note that none of the basis vectors outside the block are modified, it suffices to bound $\|K^{(a)}U\|$. Applying Lemmas C.1(i) and C.2, we obtain

$$\|K^{(a)}U\| \leqslant \|K^{(a)}U\|_F \leqslant \|K^{(a)}\|_F\|U\|_F \leqslant \alpha^{k-1}k^{k+3}\|B^{(a)}\|^{2n+1}.$$

Thus, $\|B^{(b)}\| \leqslant \alpha^{k-1}k^{k+3}\|B^{(a)}\|^{2n+1}$. Since $B^{(a)}$ is LLL-reduced with factor ε (by Step 3 in Algorithm 3), we have $\|B^{(a)}\| \leqslant \sqrt{\alpha}^{n-1}\|B_0\|$ and the last assertion follows. This completes the proof. □

Lemma D.2. *Let $B \in \mathbb{Z}^{m\times n}$ be the input basis of Algorithm 5. If μ' and D'^2 are the $k \times k$ matrices defined in Line 1 and Line 2, respectively, of Algorithm 5, then there exist positive integers h_1 and h_2 such that*

$$h_1 \leqslant \|B\|^{2nk}, \quad h_1\mu' \subset \mathbb{Z}^{k\times k}, \quad \|h_1\mu'\|_F \leqslant k^k\|B\|^{2n(k-1)},$$
$$h_2 \leqslant \|B\|^{2nk}, \quad h_2D'^2 \subset \mathbb{Z}^{k\times k}, \quad \|h_2D'^2\|_F \leqslant k\|B\|^{2n(k+1)}.$$

Proof. For $B = QD\mu$, by [**10**, p. 523], we have

$$\|\mathbf{b}^*_j\|^2 = \frac{d_j}{d_{j-1}}, \quad \text{for } 1 \leqslant j \leqslant n, \tag{D.1}$$
$$d_\ell\mu_{j,\ell} \in \mathbb{Z}, \quad \text{for } 1 \leqslant \ell < j \leqslant n, \tag{D.2}$$

where $d_j = \mathrm{vol}(B_{[1,j]})^2 \in \mathbb{Z}$ and $d_j \leqslant \|B\|^{2j}$.

Let $\mu_{[i]} = (\mu_{\iota,\jmath})_{ik+r+1\leqslant\iota,\jmath\leqslant ik+k+r}$, then $\mu_{[ik+r+1,ik+k+r]} = (\mu_{[i]})^t$. Thus, $\mu' := (\mu_{[ik+r+1,ik+k+r]})^{-s} = R_k(\mu_{[i]})^{-1}R_k = R_k\widetilde{\mu_{[i]}}R_k$. By the definition of adjoint matrix and (D.2), then the integer h_1 satisfies $h_1\mu' \subset \mathbb{Z}^{k\times k}$: $h_1 = \prod_{j=ik+r+1}^{ik+k+r-1} d_j \leqslant \|B\|^{2n(k-1)}$. Note that B is size-reduced, then $\|\mu'\|_F = \|\widetilde{\mu_{[i]}}\|_F \leqslant k\|\mu_{[i]}\|_F^{k-1} \leqslant k^k$. This yields $\|h_1\mu'\|_F \leqslant k^k\|B\|^{2n(k-1)}$.

Since $D'^2 = \mathrm{Diag}(1/\|\mathbf{b}^*_{ik+k+r}\|^2, \ldots, 1/\|\mathbf{b}^*_{ik+r+1}\|^2)$, by (D.1), then the integer h_2 satisfies $h_2D'^2 \subset \mathbb{Z}^{k\times k}$: $h_2 = \prod_{j=ik+r+1}^{ik+k+r} d_j \leqslant \|B\|^{2nk}$. Note that $\|D'^2\|_F \leqslant \sqrt{k}\|B\|^{2(n-1)}$, we have $\|h_2D'^2\|_F \leqslant k\|B\|^{2n(k+1)}$. This completes the proof. □

Lemma D.3. *Let $B^{(a)}$ and $B^{(b)}$ denote the input and output bases, respectively, of Algorithm 5 on the projected block $B^{(a)}_{[ik+r+1,ik+k+r]}$, and $\alpha = 4(1+\varepsilon)/(3-\varepsilon)$ and $B_0 \in \mathbb{Z}^{m\times n}$ denote the input of Algorithm 3. Then $L(B^{(a)}) = L(B^{(b)})$, $B^{(b)}_{[ik+r+1,ik+k+r]}$ is $(1+\varepsilon, r)$-dual-Rankin reduced and*

$$\|B^{(b)}\| \leqslant \alpha^{\frac{3}{2}n^2k}k^{k^2+3k}\|B_0\|^{\frac{3}{2}nk}.$$

Proof. We use the notation of Algorithm 5. We assume that Algorithm 5 executes Steps 7–10. Since U is unimodular, U^{-s} is unimodular and therefore $L(B^{(a)}) = L(B^{(b)})$. Note that $U_{[1,r]} = H$ and $(B^{(b)}_{[ik+r+1,ik+k+r]})^{-s} = (B^{(a)}_{[ik+r+1,ik+k+r]})^{-s}U$, then $B^{(b)}_{[ik+r+1,ik+k+r]}$ is r-dual-Rankin reduced.

Let $K^{(a)} = (\mathbf{b}^{(a)}_{ik+r+1}, \ldots, \mathbf{b}^{(a)}_{ik+k+r})$. Since none of the basis vectors outside the block are modified, it suffices to bound $\|K^{(a)}U^{-s}\|$. Using Lemma C.1(i) and C.3, we have

$$\|K^{(a)}U^{-s}\| \leqslant \|K^{(a)}U^{-s}\|_F \leqslant \|K^{(a)}\|_F\|U^{-s}\|_F \leqslant \alpha^{(k-1)^2}k^{k^2+3k-2}\|B^{(a)}\|^{3nk-3n+1}.$$

https://doi.org/10.1112/S1461157014000333 Published online by Cambridge University Press

This yields $\|B^{(b)}\| \leqslant \alpha^{(k-1)^2} k^{k^2+3k} \|B^{(a)}\|^{3nk-2n}$. Since $\|B^{(a)}\| \leqslant \sqrt{\alpha}^{n-1} \|B_0\|$ (by Step 8 in Algorithm 3), the last assertion follows. This completes the proof. □

References

1. W. Banaszczyk, 'New bounds in some transference theorems in the geometry of numbers', *Math. Ann.* 296 (1993) no. 1, 625–635.
2. D. Dadush and D. Micciancio, 'Algorithms for the densest sub-lattice problem', *Proc. SODA '13* (SIAM, 2013) 1103–1122.
3. N. Gama, N. Howgrave-Graham, H. Koy and P. Nguyen, 'Rankin's constant and blockwise lattice reduction', *Proc. CRYPTO '06*, Lecture Notes in Computer Science 4117 (Springer, 2006) 112–130.
4. N. Gama, N. Howgrave-Graham and P. Q. Nguyen, 'Symplectic lattice reduction and NTRU', *Proc. EUROCRYPT '06*, Lecture Notes in Computer Science 4004 (Springer, 2006) 233–253.
5. N. Gama and P. Q. Nguyen, 'Finding short lattice vectors within Mordell's inequality', *Proc. STOC '08* (ACM, 2008) 207–216.
6. N. Gama, P. Q. Nguyen and O. Regev, 'Lattice enumeration using extreme pruning', *Proc. EUROCRYPT '10*, Lecture Notes in Computer Science 6110 (Springer, 2010) 257–278.
7. C. Hermite, 'Second letter to M. Jacobi (in French)', *J. reine angew. Math.* 40 (1850) 279–290.
8. A. Korkine and G. Zolotareff, 'Sur les formes quadratiques', *Math. Ann.* 6 (1873) 366–389.
9. J. C. Lagarias, H. W. Lenstra Jr. and C. P. Schnorr, 'Korkine Zolotarev bases and successive minima of a lattice and its reciprocal', *Combinatorica* 10 (1990) 333–348.
10. A. K. Lenstra, H. W. Lenstra Jr. and L. Lovász, 'Factoring polynomials with rational coefficients', *Math. Ann.* 261 (1982) 366–389.
11. S. S. Magliveras, T. van Trung and W. Wei, 'Primitive sets in a lattice', *Australas. J. Combin.* 40 (2008) 173–186.
12. J. Martinet, *Perfect lattices in Euclidean spaces* (Springer, 2002).
13. D. Micciancio, 'Efficient reduction among lattice problems', *Proc. SODA '08* (ACM/SIAM, 2008) 84–93.
14. D. Micciancio and P. Voulgaris, 'A deterministic single exponential time algorithm for most lattice problems based on Voronoi cell computations', *Proc. STOC '10* (ACM, 2010) 351–358.
15. L. J. Mordell, 'Observation on the minimum of a positive quadratic form in eight variables', *J. Lond. Math. Soc.* 19 (1944) 3–6.
16. P. Nguyen and D. Stehlé, 'An LLL algorithm with quadratic complexity', *SIAM J. Comput.* 39 (2009) no. 3, 874–903.
17. R. A. Rankin, 'On positive definite quadratic forms', *J. Lond. Math. Soc.* 28 (1953) 309–314.
18. O. Regev, Lecture 8: Dual lattices, 2004, http://www.cims.nyu.edu/regev/teaching/lattices-fall-2004/index.html.
19. C. P. Schnorr, 'A hierarchy of polynomial time lattice basis reduction algorithms', *Theoret. Comput. Sci.* 53 (1987) 201–224.
20. C. P. Schnorr and M. Euchner, 'Lattice basis reduction: improved practical algorithms and solving subset sum problems', *Math. Program.* 66 (1994) 181–199.
21. C. P. Schnorr and H. H. Hörner, 'Attacking the Chor–Rivest cryptosystem by improved lattice reduction', *Proc. EUROCRYPT '95*, Lecture Notes in Computer Science 921 (Springer, 1995) 1–12.

Jianwei Li
Institute for Advanced Study
Tsinghua University
Beijing 100084
China
lijianwei10@mails.tsinghua.edu.cn

Phong Q. Nguyen
INRIA
France
and
Institute for Advanced Study
Tsinghua University
Beijing 100084
China
http://www.di.ens.fr/∼pnguyen/

https://doi.org/10.1112/S1461157014000333 Published online by Cambridge University Press