Computing an LLL-reduced basis of the orthogonal lattice

Jingwei Chen[*], Damien Stehlé[†], Gilles Villard[‡]

Abstract

As a typical application, the Lenstra-Lenstra-Lovász lattice basis reduction algorithm (LLL) is used to compute a reduced basis of the orthogonal lattice for a given integer matrix, via reducing a special kind of lattice bases. With such bases in input, we propose a new technique for bounding from above the number of iterations required by the LLL algorithm. The main technical ingredient is a variant of the classical LLL potential, which could prove useful to understand the behavior of LLL for other families of input bases.

1 Introduction

Let $k < n$ be two positive integers. Given a full column rank $n \times k$ integer matrix $\mathbf{A} = (a_{i,j})$, we study the behaviour of the Lenstra-Lenstra-Lovász algorithm [6] for computing a reduced basis for the *orthogonal lattice* of $\mathbf{A}$

$$\mathcal{L}^{\perp}(\mathbf{A}) = \left\{\boldsymbol{m} \in \mathbb{Z}^n : \mathbf{A}^T\boldsymbol{m} = \mathbf{0}\right\} = \ker(\mathbf{A}^T) \cap \mathbb{Z}^n. \tag{1}$$

The algorithm proceeds by unimodular column transformations from the input matrix:

$$\mathrm{Ext}_K(\mathbf{A}) := \begin{pmatrix} K \cdot \mathbf{A}^T \\ \mathbf{I}_n \end{pmatrix} = \begin{pmatrix} K \cdot a_{1,1} & K \cdot a_{2,1} & \cdots & K \cdot a_{n,1} \\ \vdots & \vdots & \ddots & \vdots \\ K \cdot a_{1,k} & K \cdot a_{2,k} & \cdots & K \cdot a_{n,k} \\ 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & 1 \end{pmatrix} \in \mathbb{Z}^{(n+k)\times n} \tag{2}$$

where K is a sufficiently large positive integer. The related definitions and the LLL algorithm are given in Section 2. The reader may refer to [9] for a comprehensive review of LLL.

Usual techniques gives that LLL reduction requires $\mathcal{O}(n^2 \log(K \cdot \|\mathbf{A}\|))$ *swaps* (see Step 7 of Algorithm 1) for a basis as in (2), where $\|\mathbf{A}\|$ bounds from above the Euclidean norms of the rows and columns of $\mathbf{A}$. We recall that most known LLL reduction algorithms iteratively perform two types of vector operations: translations and swaps. The motivation for studying bounds on the number of swaps comes from the fact that this number governs known cost analyses of the reduction.

[*]Chongqing Key Laboratory of Automated Reasoning & Cognition, CIGIT, Chinese Academy of Sciences, Chongqing, China.

[†]ENS de Lyon, Laboratoire LIP (U. Lyon, CNRS, Inria, ENS de Lyon, UCBL), France.

[‡]CNRS, Laboratoire LIP (U. Lyon, CNRS, Inria, ENS de Lyon, UCBL), France.

Folklore applications of the reduction of bases as in (2) are for example the computation of integer relations between real numbers [3, 1], or the computation of minimal polynomials [5] (see also [9]). A main difficulty however, both theoretically and practically, remains to master the *scaling parameter* K that can be very large. Typically, an appropriate value for K may be derived from *a priori* bounds such as heights of algebraic numbers [5] and may overestimate the smallest suitable value for actual inputs. Since the usual bound on the number of swaps is linear in $\log K$, the overestimation could be a serious drawback. We show that this may not be always the case.

We consider the reduction of a basis as in (2) for obtaining a basis of the orthogonal lattice (1). We establish a bound on the number of swaps that does not depend on K as soon as K is above a threshold value (as specified in (7)). This threshold depends only on the dimension and invariants of the orthogonal lattice.

Our contribution. The analyses of LLL and many LLL variants bound the number of iterations using the geometric decrease of a potential that is defined using the Gram-Schmidt norms of the basis vectors; see (6). We are going to see that this classical potential does not capture a typical unbalancedness of the Gram-Schmidt norms that characterizes bases in (2). Taking into account the latter structure will lead us to a better bound for the number of iterations (see Table 1). Intuitively, as the basis being manipulated becomes reduced, two groups of vectors are formed: some with small Gram-Schmidt norms, and some others with large Gram-Schmidt norms. As soon they are formed, the two groups do not interfere much.

In Section 3 we introduce a new LLL potential function that generalizes the classical one for capturing the previously mentioned unbalancedness. Its geometric decrease during the execution also leads to a bound on the number of iterations (see Theorem 3.3). In Section 4, we specialize the potential to the case of bases as in (2) for computing the orthogonal lattice $\mathcal{L}^{\perp}(\mathbf{A})$. As discussed above, we will see that at some point the number of iterations can be shown to be independent of the scaling parameter K, or, in other words, independent of a further increase of the input size. We note that this new potential is defined for all lattice bases, but it may not always lead to better bounds on the number of LLL iterations.

The extended gcd algorithm in [4] uses a basis as in (2) with $k = 1$. It is shown in [4, Sec. 3, p. 127] that if K is sufficiently large, then the sequence of operations performed by LLL is independent of K. A somewhat similar remark had been made in [11]. Our new potential function allows a better understanding of the phenomenon.

Future work. Future research directions are to apply this potential to bit complexity studies of the LLL basis reduction [12, 10, 7], especially for specific input bases. Indeed, an interesting problem is to design an algorithm for computing a reduced basis for $\mathcal{L}^{\perp}(\mathbf{A})$ that features a bit complexity bound independent of the scaling parameter.

Notations. Throughout the paper, vectors are in column and denoted in bold. For $\boldsymbol{x} \in \mathbb{R}^m$, $\|\boldsymbol{x}\|$ is the Euclidean norm of $\boldsymbol{x}$. Matrices are denoted by upper case letters in bold, such as $\mathbf{A}$, $\mathbf{B}$, etc. For a matrix $\mathbf{A}$, $\mathbf{A}^T$ is the transpose of $\mathbf{A}$, and $\|\mathbf{A}\|$ bounds the Euclidean norms of the columns and rows of $\mathbf{A}$. The base of logarithm is 2.

2 Preliminaries

We give some basic definitions and results that are needed for the rest of the paper. A comprehensive presentation of the LLL algorithm and its applications may be found in [9].

Gram-Schmidt orthogonalization. Let $\boldsymbol{b}_1, \cdots, \boldsymbol{b}_n \in \mathbb{R}^m$ be linearly independent vectors.

Their *Gram-Schmidt orthogonalization* $\boldsymbol{b}_1^*, \cdots, \boldsymbol{b}_n^*$ is defined as follows:

$$\boldsymbol{b}_1^* = \boldsymbol{b}_1 \quad \text{and} \quad \forall i > 1 : \boldsymbol{b}_i^* = \boldsymbol{b}_i - \sum_{j=1}^{i-1} \mu_{i,j} \boldsymbol{b}_j^*,$$

where the $\mu_{i,j} = \frac{\langle \boldsymbol{b}_i, \boldsymbol{b}_j^* \rangle}{\langle \boldsymbol{b}_j^*, \boldsymbol{b}_j^* \rangle}$ for all $i > j$ are called the *Gram-Schmidt coefficients*. We call the $\|\boldsymbol{b}_i^*\|$'s the *Gram-Schmidt norms* of the $\boldsymbol{b}_i$'s.

LATTICES. A *lattice* $\Lambda \subseteq \mathbb{R}^m$ is a discrete additive subgroup of $\mathbb{R}^m$. If $(\boldsymbol{b}_i)_{i \leq n}$ is a set of generators for Λ, then

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

If the $\boldsymbol{b}_i$'s are linearly independent, then they are said to form a *basis* of Λ. When $n \geq 2$, there exist infinitely many bases for a lattice. Every basis is related by an integral unimodular transformation (a linear transformation with determinant ± 1) to any other. Further, the number of vectors of different bases of a lattice Λ is always the same, and we call this number the *dimension* of the lattice, denoted by $\dim(\Lambda)$. If $\mathbf{B} = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n) \in \mathbb{R}^{m \times n}$ is a basis for a lattice $\Lambda = \mathcal{L}(\mathbf{B})$, the *determinant* of the lattice is defined as $\det(\Lambda) = \sqrt{\det(\mathbf{B}^T \mathbf{B})}$. It is invariant across all bases of Λ.

SUCCESSIVE MINIMA. For a given lattice Λ, we let $\lambda_1(\Lambda)$ denote the minimum Euclidean norm of vectors in $\Lambda \setminus \{\mathbf{0}\}$. From Minkowski's first theorem, we have $\lambda_1(\Lambda) \leq \sqrt{n} \cdot \det(\Lambda)^{1/n}$, where $n = \dim(\Lambda)$. More generally, for all $1 \leq i \leq n$, we define the i-th *minimum* as

$$\lambda_i(\Lambda) = \min_{\substack{\boldsymbol{v}_1, \cdots, \boldsymbol{v}_i \in \Lambda \\ \text{linearly independent}}} \max_{j \leq i} \|\boldsymbol{v}_j\|.$$

Minkowski's second theorem states that $\prod_{i \leq n} \lambda_i(\Lambda) \leq \sqrt{n}^n \cdot \det(\Lambda)$.

SUBLATTICES. Let $\Lambda \subseteq \mathbb{R}^n$ be a lattice. We say that Λ' is a *sublattice* of Λ if $\Lambda' \subseteq \Lambda$ is a lattice as well. If Λ' is a sublattice of Λ then $\lambda_i(\Lambda) \leq \lambda_i(\Lambda')$ for $i \leq \dim(\Lambda')$.

ORTHOGONAL LATTICES. Given a full column rank matrix $\mathbf{A} \in \mathbb{Z}^{n \times k}$, the set $\mathcal{L}^\perp(\mathbf{A})$ defined in (1) forms a lattice, called the *orthogonal lattice* of $\mathbf{A}$. We have $\dim(\mathcal{L}^\perp(\mathbf{A})) = n - k$. From [8, Proposition 2.9] and Hadamard's inequality, we have that

$$\det(\mathcal{L}^\perp(\mathbf{A})) \leq \det(\mathbf{A} \cdot \mathbb{Z}^k) \leq \|\mathbf{A}\|^k. \tag{3}$$

LLL-REDUCED BASES. The goal of lattice basis reduction is to find a basis with vectors as short and orthogonal to each other as possible. Among numerous lattice reduction notions, the LLL-reduction [6] is one of the most commonly used. Let $\frac{1}{4} < \delta < 1$. Let $\mathbf{B} = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n) \in \mathbb{R}^{m \times n}$ be a basis of a lattice Λ. We say that $\mathbf{B}$ is *size-reduced* if all Gram-Schmidt coefficients satisfy $|\mu_{ij}| \leq \frac{1}{2}$. We say that $\mathbf{B}$ satisfies the *Lovász conditions* if for all i we have $\delta \|\boldsymbol{b}_i^*\|^2 \leq \|\boldsymbol{b}_{i+1}^*\|^2 + \mu_{i+1,i}^2 \|\boldsymbol{b}_i^*\|^2$. If a basis $\mathbf{B}$ is size-reduced and satisfies the Lovász conditions, then we say that $\mathbf{B}$ is *LLL-reduced* (with respect to the parameter δ). If a basis $\mathbf{B} = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n)$ of Λ is LLL-reduced, then we have:

$$\begin{aligned} &\forall i < n, \ \|\boldsymbol{b}_i^*\|^2 \leq \alpha \|\boldsymbol{b}_{i+1}^*\|^2, \\ &\forall i \leq n, \ \|\boldsymbol{b}_i\|^2 \leq \alpha^{i-1} \|\boldsymbol{b}_i^*\|^2, \end{aligned} \tag{4}$$

$$\forall i \le j \le n,\ \|\boldsymbol{b}_i\| \le \alpha^{\frac{n-1}{2}} \lambda_j(\Lambda), \tag{5}$$

where $\alpha = \frac{4}{4\delta-1}$. In particular, we have $\|\boldsymbol{b}_1\| \le \alpha^{\frac{n-1}{2}} \lambda_1(\Lambda)$. In this paper, we use the original LLL parameter $\delta = \frac{3}{4}$ and hence $\alpha = 2$.

THE LLL ALGORITHM. We now sketch the LLL algorithm. Although there exist many LLL variants in the literature, most of them follow the following structure. Step 7 is called an *LLL swap*.

Algorithm 1 (LLL)

Input: A basis $(\boldsymbol{b}_i)_{i\le n}$ of a lattice $\Lambda \subseteq \mathbb{Z}^n$.
Output: An LLL-reduced basis of Λ.

1: $i := 2$;
2: **while** $i \le n$ **do**
3: Size-reduce $\boldsymbol{b}_i$ by $\boldsymbol{b}_1, \cdots, \boldsymbol{b}_{i-1}$;
4: **if** Lovász condition holds for i **then**
5: Set $i := i + 1$;
6: **else**
7: (LLL swap) Swap $\boldsymbol{b}_i$ and $\boldsymbol{b}_{i-1}$; set $i := \max\{i-1, 2\}$;
8: **end if**
9: **end while**
10: Return $(\boldsymbol{b}_i)_{i\le n}$.

To clarify the structure of the algorithm, we omit some details in the above description, e.g., the update of Gram-Schmidt coefficients. From the sketch, we see that we can bound the running-time of LLL by the number of while loop iterations times the cost of each iteration. In fact, most cost bounds for LLL variants proceed via this simple argument. It was showed in [6] that the number of LLL swaps is $\mathcal{O}(n^2 \log \|\mathbf{B}\|)$. The following lemma plays a very important role in the analysis of LLL; see [6] for a proof.

Lemma 2.1. *Let* $\mathbf{B}$ *and* $\mathbf{B}'$ *be bases after and before an LLL swap between* $\boldsymbol{b}_i$ *and* $\boldsymbol{b}_{i+1}$. *Then*

$$\begin{aligned}
\max\{\|\boldsymbol{b}_i'^*\|, \|\boldsymbol{b}_{i+1}'^*\|\} &\le \max\{\|\boldsymbol{b}_i^*\|, \|\boldsymbol{b}_{i+1}^*\|\},\\
\min\{\|\boldsymbol{b}_i'^*\|, \|\boldsymbol{b}_{i+1}'^*\|\} &\ge \min\{\|\boldsymbol{b}_i^*\|, \|\boldsymbol{b}_{i+1}^*\|\},\\
\|\boldsymbol{b}_i^*\| \cdot \|\boldsymbol{b}_{i+1}^*\| &= \|\boldsymbol{b}_i'^*\| \cdot \|\boldsymbol{b}_{i+1}'^*\|,\\
\frac{\|\boldsymbol{b}_{i+1}'^*\|}{\|\boldsymbol{b}_{i+1}^*\|} = \frac{\|\boldsymbol{b}_i^*\|}{\|\boldsymbol{b}_i'^*\|} &\ge \frac{2}{\sqrt{3}},\\
\forall j \notin \{i, i+1\} \quad &: \quad \boldsymbol{b}_j'^* = \boldsymbol{b}_j^*.
\end{aligned}$$

3 A new potential

In this section, we introduce a variant of the classical LLL potential

$$\Pi(\mathbf{B}) = \sum_{i=1}^{n-1} (n-i) \log \|\boldsymbol{b}_i^*\| \tag{6}$$

of a lattice basis $\mathbf{B}$, which is well-suited for analyzing the number of LLL swaps for the case that both the input and output bases have k large Gram-Schmidt norms and $n-k$ small Gram-Schmidt

norms, for some $k < n$. This is for example the case for the input basis as (2); see Section 4.2. The new potential is aimed at accurately measuring the progress made during the LLL execution, for such unbalanced bases.

Definition 3.1. Let $k \leq n \leq m$ be positive integers and $\mathbf{B} \in \mathbb{R}^{m \times n}$ be full column rank. We let $s_1 < \ldots < s_{n-k}$ be the indices of the $n-k$ smallest Gram-Schmidt norms of $\mathbf{B}$ (using the lexicographical in case there are several $(n-k)$-th smallest Gram-Schmidt norms), and set $S = \{s_i\}_{i \leq n-k}$. We let $\ell_1 < \ldots < \ell_k$ be the indices of the other k Gram-Schmidt norms, and set $L = \{\ell_j\}_{j \leq k}$. The *$k$-th LLL potential* of $\mathbf{B}$ is defined as:

$$\Pi_k(\mathbf{B}) = \sum_{j=1}^{k-1} (k-j) \log \|\boldsymbol{b}^*_{\ell_j}\| - \sum_{i=1}^{n-k} i \log \|\boldsymbol{b}^*_{s_i}\| + \sum_{i=1}^{n-k} s_i.$$

Note that for $k = n$, we recover the classical potential Π. The rationale behind Π_k is that in some cases we know that the output basis is made of vectors of very unbalanced Gram-Schmidt norms. As this basis is reduced, this means the first vectors have a small Gram-Schmidt norm, while the last vectors have large Gram-Schmidt norms. During the execution of LLL, such short and large vectors do not interfere much. This is an unusual phenomenon: most often, long vectors are made shorter and short vectors are made longer, so that they are all balanced at the end. But this can happen if the long vectors are rather orthogonal to the short ones. When this is the case, LLL actually runs faster than usual, because it merely "sorts" the short vectors and the long vectors, without making them interact to create shorter vectors. Of course, it can do more intense computations among the short vectors and among the long vectors. Unbalancedness of Gram-Schmidt norms is not captured by the classical potential, but it is with Π_k. In particular, the new potential Π_k allows to not "pay" for the output unbalancedness in the analysis of the number of LLL swaps.

Similarly to the classical potential, the k-th LLL potential monotonically decreases with the number of LLL swaps. More precisely, we have the following

Proposition 3.2. *Let $\mathbf{B}$ and $\mathbf{B}'$ be the current n-dimensional lattice bases before and after an LLL swap. Then for any $k \leq n$, we have $\Pi_k(\mathbf{B}) - \Pi_k(\mathbf{B}') \geq \log(2/\sqrt{3})$.*

Proof. Recall that S and L are the index sets for the $n-k$ Gram-Schmidt norms and the other k Gram-Schmidt norms for the lattice basis $\mathbf{B}$. We define S' and L' for $\mathbf{B}'$ similarly.

Suppose that this LLL swap occurs between $\boldsymbol{b}_\kappa$ and $\boldsymbol{b}_{\kappa+1}$. Then we must be in one of the following four cases.

Case 1: $\kappa \in S$ and $\kappa + 1 \in S$.

Let $i_0 \leq n-k$ such that $\kappa = s_{i_0}$ and $\kappa + 1 = s_{i_0+1}$. From Lemma 2.1, we have $S' = S$ and $L' = L$, and hence $\kappa = s'_{i_0}$ and $\kappa+1 = s'_{i_0+1}$. For the other indices, we have $s'_i = s_i$ (for $i \leq n-k$) and $\ell'_j = \ell_j$ (for $j \leq k$). Then

$$\begin{aligned}
\Pi_k(\mathbf{B}) - \Pi_k(\mathbf{B}') &= \sum_{j=1}^{k} (k-j) \log \frac{\|\boldsymbol{b}^*_{\ell_j}\|}{\|\boldsymbol{b}'^*_{\ell'_j}\|} + \sum_{i=1}^{n-k} i \log \frac{\|\boldsymbol{b}'^*_{s'_i}\|}{\|\boldsymbol{b}^*_{s_i}\|} + \sum_{i=1}^{n-k} (s_i - s'_i) \\
&= i_0 \log \frac{\|\boldsymbol{b}'^*_{s'_{i_0}}\|}{\|\boldsymbol{b}^*_{s_{i_0}}\|} + (i_0+1) \log \frac{\|\boldsymbol{b}'^*_{s'_{i_0+1}}\|}{\|\boldsymbol{b}^*_{s_{i_0+1}}\|} \\
&= \log \frac{\|\boldsymbol{b}'^*_{\kappa+1}\|}{\|\boldsymbol{b}^*_{\kappa+1}\|} \geq \log\left(\frac{2}{\sqrt{3}}\right),
\end{aligned}$$

where the last inequality follows from Lemma 2.1.

Case 2: $\kappa \in L$ and $\kappa+1 \in L$.

The treatment of Case 1 can be adapted readily.

Case 3: $\kappa \in L$, $\kappa+1 \in S$, $S' = S$ and $L' = L$.

Let $j_0 \le k$ such that $\kappa = \ell_{j_0}$, and $i_0 \le n-k$ such that $\kappa+1 = s_{i_0}$. Then we have $\kappa = \ell'_{j_0}$ and $\kappa+1 = s'_{i_0}$. For the other indices, we have $s'_i = s_i^{(t)}$ (for $i \le n-k$) and $\ell'_j = \ell_j^{(t)}$ (for $j \le k$). Thus

$$\begin{aligned}
\Pi_k(\mathbf{B}) - \Pi_k(\mathbf{B}') &= \sum_{j=1}^{k}(k-j)\log\frac{\|\boldsymbol{b}^*_{\ell_j}\|}{\|\boldsymbol{b}'^*_{\ell'_j}\|} + \sum_{i=1}^{n-k} i\log\frac{\|\boldsymbol{b}'^*_{s'_i}\|}{\|\boldsymbol{b}^*_{s_i}\|} + \sum_{i=1}^{n-k}(s_i - s'_i) \\
&= (k-j_0)\log\frac{\|\boldsymbol{b}^*_{\ell_{j_0}}\|}{\|\boldsymbol{b}'^*_{\ell'_{j_0}}\|} + i_0\log\frac{\|\boldsymbol{b}'^*_{s'_{i_0}}\|}{\|\boldsymbol{b}^*_{s_{i_0}}\|} \\
&= (k-j_0+i_0)\log\frac{\|\boldsymbol{b}'^*_{\kappa+1}\|}{\|\boldsymbol{b}^*_{\kappa+1}\|} \ge \log\left(\frac{2}{\sqrt{3}}\right),
\end{aligned}$$

where the last inequality follows from Lemma 2.1 and the fact that $k - j_0 + i_0 \ge 1$.

Case 4: $\kappa \in L$, $\kappa+1 \in S$, $S' = S \cup \{\kappa\} \setminus \{\kappa+1\}$ and $L' = L \cup \{\kappa+1\} \setminus \{\kappa\}$.

Let $j_0 \le k$ such that $\kappa = \ell_{j_0}$, and $i_0 \le n-k$ such that $\kappa+1 = s_{i_0}$. Then $\kappa = s'_{i_0}$ and $\kappa+1 = \ell'_{j_0}$. For other indices, we have $s'_i = s_i$ (for $i \le n-k$) and $\ell'_j = \ell_j$ (for $j \le k$). Then

$$\begin{aligned}
\Pi_k(\mathbf{B}) - \Pi_k(\mathbf{B}') &= \sum_{j=1}^{k}(k-j)\log\frac{\|\boldsymbol{b}^*_{\ell_j}\|}{\|\boldsymbol{b}'^*_{\ell'_j}\|} + \sum_{i=1}^{n-k} i\log\frac{\|\boldsymbol{b}'^*_{s'_i}\|}{\|\boldsymbol{b}^*_{s_i}\|} + \sum_{i=1}^{n-k}(s_i - s'_i) \\
&= (k-j_0)\log\frac{\|\boldsymbol{b}^*_{\ell_{j_0}}\|}{\|\boldsymbol{b}'^*_{\ell'_{j_0}}\|} + i_0\log\frac{\|\boldsymbol{b}'^*_{s'_{i_0}}\|}{\|\boldsymbol{b}^*_{s_{i_0}}\|} + 1 \\
&= (k-j_0)\log\frac{\|\boldsymbol{b}^*_{\kappa}\|}{\|\boldsymbol{b}'^*_{\kappa+1}\|} + i_0\log\frac{\|\boldsymbol{b}'^*_{\kappa}\|}{\|\boldsymbol{b}^*_{\kappa+1}\|} + 1 \\
&\ge 1,
\end{aligned}$$

where the last inequality follows from Lemma 2.1. The observation that $1 \ge \log(2/\sqrt{3})$ allows to complete the proof. □

With the above property of the k-th LLL potential, we can bound the number of LLL swaps that LLL performs.

Theorem 3.3. *Let $\mathbf{B} \in \mathbb{R}^{m\times n}$ be a full column rank matrix. Let $\mathbf{B}'$ be the basis returned by the LLL algorithm when given $\mathbf{B}$ as input. Then the number of swaps that LLL performs is no greater than*

$$\min_{1\le k\le n} \frac{\Pi_k(\mathbf{B}) - \Pi_k(\mathbf{B}')}{\log\left(\frac{2}{\sqrt{3}}\right)}.$$

4 Application to orthogonal lattices

As an application of the k-th LLL potential Π_k, we consider the problem of computing an LLL-reduced basis of an orthogonal lattice. Let $\mathbf{A} \in \mathbb{Z}^{n\times k}$ with $n \ge k$. We aim at computing an LLL-reduced basis of the orthogonal lattice $\mathcal{L}^{\perp}(\mathbf{A})$, by LLL-reducing $\mathrm{Ext}_K(\mathbf{A})$ (as defined in (2)), for a sufficiently large integer K.

In Subsection 4.1, we provide a sufficient condition on the scaling parameter K so that a LLL-reduced basis of $\mathcal{L}^\perp(\mathbf{A})$ can be extracted from a LLL-reduced basis of $\mathcal{L}(\mathrm{Ext}_K(\mathbf{A}))$. For such a sufficiently large K, we study the Gram-Schmidt orthogonalizations of the input and output bases of the LLL call to $\mathrm{Ext}_K(\mathbf{A})$ in Subsection 4.2, and we provide a bound on the number of required LLL swaps which is independent of K in Subsection 4.3.

4.1 Correctness

For $n \geq k$, we define $\sigma_{n,k}$ as the map that embeds $\mathbb{R}^n$ into $\mathbb{R}^{n+k}$ by adding 0's in the first k coordinates.

$$\begin{array}{rccc} & \mathbb{R}^n & \to & \mathbb{R}^{n+k} \\ \sigma_{n,k}: & (x_1, \cdots, x_n)^T & \mapsto & (\underbrace{0, \cdots, 0}_{k}, \underbrace{x_1, \cdots, x_n}_{n})^T. \end{array}$$

We also define $\delta_{n,k}$ as the map that erases the first k coordinates of a vector in $\mathbb{R}^{n+k}$.

$$\begin{array}{rccc} & \mathbb{R}^{n+k} & \to & \mathbb{R}^n \\ \delta_{n,k}: & (x_1, \cdots, x_k, x_{k+1}, \cdots, x_{k+n})^T & \mapsto & (x_{k+1}, \cdots, x_{k+n})^T. \end{array}$$

We extend these functions to matrices in the canonical way. The following shows that if K is sufficiently large, then calling the LLL algorithm on $\mathrm{Ext}_K(\mathbf{A})$ provides an LLL-reduced basis of $\mathcal{L}^\perp(\mathbf{A})$.

Proposition 4.1. *Let $\mathbf{A} \in \mathbb{Z}^{n \times k}$ be full column rank and $\mathbf{B} = \mathrm{Ext}_K(\mathbf{A})$. If $\mathbf{B}'$ is an LLL-reduced basis of $\mathcal{L}(\mathbf{B})$ and*

$$K > 2^{\frac{n-1}{2}} \cdot \lambda_{n-k}(\mathcal{L}^\perp(\mathbf{A})), \tag{7}$$

then $\delta_{n,k}(\boldsymbol{b}'_1), \cdots, \delta_{n,k}(\boldsymbol{b}'_{n-k})$ is an LLL-reduced basis of $\mathcal{L}^\perp(\mathbf{A})$.

Proof. As $\mathbf{A} \in \mathbb{Z}^{n \times k}$ is full column rank, we have $\dim(\mathcal{L}^\perp(\mathbf{A})) = n - k$. For any basis $\mathbf{C} \in \mathbb{Z}^{n \times (n-k)}$ of $\mathcal{L}^\perp(\mathbf{A})$, we have $\sigma_{n,k}(\mathbf{C}) = \mathbf{B} \cdot \mathbf{C}$, and hence the lattice $\sigma_{n,k}(\mathcal{L}^\perp(\mathbf{A}))$ is a sublattice of $\mathcal{L}(\mathbf{B})$. This implies that, for all $i \leq n - k$,

$$\lambda_i(\mathcal{L}(\mathbf{B})) \leq \lambda_i(\sigma_{n,k}(\mathcal{L}^\perp(\mathbf{A}))) = \lambda_i(\mathcal{L}^\perp(\mathbf{A})).$$

It follows from (5) that, for all $i \leq n - k$,

$$\|\boldsymbol{b}'_i\|^2 \leq 2^{n-1} \cdot \lambda^2_{n-k}(\mathcal{L}(\mathbf{B})) \leq 2^{n-1} \cdot \lambda^2_{n-k}(\mathcal{L}^\perp(\mathbf{A})). \tag{8}$$

We now assume (by contradiction) that $\delta_{n,k}(\boldsymbol{b}'_i) \notin \mathcal{L}^\perp(\mathbf{A})$ for some $i \leq n - k$. Note that

$$\boldsymbol{b}'_i = \mathbf{B} \cdot \delta_{n,k}(\boldsymbol{b}'_i) = (K \cdot \delta_{n,k}(\boldsymbol{b}_{i'})^T \cdot \mathbf{A} \,|\, \delta_{n,k}(\boldsymbol{b}'_i)^T)^T.$$

As the subvector $K \cdot \delta_{n,k}(\boldsymbol{b}'_i)^T \cdot \mathbf{A}$ is non-zero, and using the assumption on K, we obtain that

$$\|\boldsymbol{b}'_i\|^2 = \|K \cdot \delta_{n,k}(\boldsymbol{b}'_i)^T \cdot \mathbf{A}\|^2 + \|\delta_{n,k}(\boldsymbol{b}'_i)\|^2 \geq K^2 > 2^{n-1} \cdot \lambda^2_{n-k}(\mathcal{L}^\perp(\mathbf{A})),$$

which contradicts (8).

From the above, we obtain that $\delta_{n,k}(\boldsymbol{b}'_1), \cdots, \delta_{n,k}(\boldsymbol{b}'_{n-k})$ are linearly independent vectors in $\mathcal{L}^\perp(\mathbf{A})$. They actually form a basis of $\mathcal{L}^\perp(\mathbf{A})$. To see this, consider an arbitrary vector $\boldsymbol{c} \in \mathcal{L}^\perp(\mathbf{A})$. The vector $\mathbf{B} \cdot \boldsymbol{c}$ belongs to the real span of $\boldsymbol{b}'_1, \cdots, \boldsymbol{b}'_{n-k}$ and to $\mathcal{L}(\mathbf{B})$. As $\mathbf{B}'$ is a basis of $\mathcal{L}(\mathbf{B})$, vector $\mathbf{B} \cdot \boldsymbol{c}$ is an integer combination of $\boldsymbol{b}'_1, \cdots, \boldsymbol{b}'_{n-k}$ and vector $\boldsymbol{c}$ is an integer combination of $\delta_{n,k}(\boldsymbol{b}'_1), \cdots, \delta_{n,k}(\boldsymbol{b}'_{n-k})$.

As $\mathbf{B}'$ is LLL-reduced and the first k coordinates of each of $\boldsymbol{b}'_1, \cdots, \boldsymbol{b}'_{n-k}$ are 0, we obtain that $\delta_{n,k}(\boldsymbol{b}'_1), \cdots, \delta_{n,k}(\boldsymbol{b}'_{n-k})$ form an LLL-reduced basis of $\mathcal{L}^\perp(\mathbf{A})$. □

To make this condition on K effective, we can bound $\lambda_{n-k}(\mathcal{L}^\perp(\mathbf{A}))$ from above. For instance, from Minkowski's second theorem, we have

$$\lambda_{n-k}(\mathcal{L}^\perp(\mathbf{A})) \leq (n-k)^{\frac{n-k}{2}} \cdot \det(\mathcal{L}^\perp(\mathbf{A})) \leq (n-k)^{\frac{n-k}{2}} \cdot \|\mathbf{A}\|^k.$$

Hence

$$K > 2^{\frac{n-1}{2}} \cdot (n-k)^{\frac{n-k}{2}} \cdot \|\mathbf{A}\|^k \tag{9}$$

suffices to guarantee that (7) holds.

The bound in (9) can be very loose. Indeed, in many cases, we expect the minima of $\mathcal{L}^\perp(\mathbf{A})$ to be balanced, and if they are so, then the following bound would suffice

$$K > 2^{\Omega(n)} \cdot \|\mathbf{A}\|^{\frac{k}{n-k}}. \tag{10}$$

4.2 On the LLL input and output bases

To bound the number of LLL swaps, we first investigate the matrix $\mathbf{B} = \mathrm{Ext}_K(\mathbf{A})$ given as input to the LLL algorithm, and the output matrix $\mathbf{B}'$.

Intuitively, from the shape of $\mathbf{B}$ and the fact that $\mathbf{A}$ is full rank, there must be k Gram-Schmidt norms of $\mathbf{B}$ that are "impacted" by the scaling parameter K, and hence have large magnitude, while other $n-k$ Gram-Schmidt norms of $\mathbf{B}$ should be of small magnitude.

On the other hand, after termination of the LLL call, the matrix $\mathbf{B}'$ must be of the form:

$$\begin{pmatrix} \mathbf{0} & * \\ \mathbf{C} & * \end{pmatrix},$$

where the columns of $\mathbf{C} \in \mathbb{Z}^{n \times (n-k)}$ form an LLL-reduced basis of the lattice $\mathcal{L}^\perp(\mathbf{A})$ (see Proposition 4.1). Since only the first k coordinates are related to the scaling parameter K, the submatrix $\mathbf{C}$ is "independent" of K. Thus, each of $\|\boldsymbol{b}'^*_1\|$, $\cdots$, $\|\boldsymbol{b}'^*_{n-k}\|$ should be relatively small (for a sufficiently large K), while each of $\|\boldsymbol{b}'^*_{n-k+1}\|$, $\cdots$, $\|\boldsymbol{b}'^*_n\|$ is "impacted" by K, and hence with large magnitude. The following result formalizes this discussion.

Proposition 4.2. *Let $\mathbf{A} \in \mathbb{Z}^{n \times k}$ be of full column rank and $\mathbf{B}'$ the output basis of LLL with $\mathbf{B} = \mathrm{Ext}_K(\mathbf{A})$ as input. If the scaling parameter $K \in \mathbb{Z}$ satisfies (7), then for the output matrix $\mathbf{B}'$ we have*

$$\forall i \leq n-k, \quad \forall j > n-k, \quad \|\boldsymbol{b}'^*_i\| < \|\boldsymbol{b}'^*_j\|.$$

Proof. From Proposition 4.1, we know that $\mathbf{B}'$ is of the form

$$\begin{pmatrix} \mathbf{0} & * \\ \mathbf{C} & * \end{pmatrix},$$

and that the columns of $\mathbf{C} \in \mathbb{Z}^{n \times k}$ form an LLL-reduced basis of $\mathcal{L}^\perp(\mathbf{A})$. We thus have, for $i \leq n-k$

$$\|\boldsymbol{b}'^*_i\|^2 \leq \|\boldsymbol{b}'_i\|^2 = \|\boldsymbol{c}_i\|^2 \leq 2^{n-k-1} \lambda^2_{n-k}(\mathcal{L}^\perp(\mathbf{A})).$$

Further, for $n-k < j \leq n$, we have

$$\|\boldsymbol{b}'^*_j\|^2 \geq 2^{-k} \|\boldsymbol{b}'^*_{n-k+1}\|^2 \geq 2^{-k} K^2.$$

The choice of K allows to complete the proof. □

We observe again that combining the condition of Proposition 4.2 together with a general purpose bound on $\lambda_{n-k}(\mathcal{L}^\perp(\mathbf{A}))$ allows to obtain a sufficient bound on K that can be efficiently derived from $\mathbf{A}$.

Although $\|\boldsymbol{b}^*_{s_i}\|$ is relatively small with respect to K, it can be bounded from below. In fact, we have a more general lower bound:

$$\forall i \leq n,\ \|\boldsymbol{b}^*_i\| \geq 1. \tag{11}$$

This is because that there is a coefficient in $\boldsymbol{b}_i$ which is equal to 1 and 0 for all other $\mathbf{b}_j$'s. This lower bound will be helpful in the proof of Theorem 4.3.

4.3 Bounding the number of LLL swaps

Suppose that K is a sufficient large positive integer satisfying (7). Proposition 4.1 guarantees that we can use LLL with $\mathbf{B} = \mathrm{Ext}_K(\mathbf{A})$ as input to compute an LLL-reduced basis for $\mathcal{L}^\perp(\mathbf{A})$. We now study the number of LLL swaps performed in this call to the LLL algorithm.

Theorem 4.3. *Let $\mathbf{A} \in \mathbb{Z}^{n\times k}$ with a non-zero k-th principal minor, and K an integer satisfying (7). Then, given $\mathbf{B} = \mathrm{Ext}_K(\mathbf{A})$ as its input, LLL allows to obtain an LLL-reduced basis of $\mathcal{L}^\perp(\mathbf{A})$ after at most $\mathcal{O}(k^3 + k(n-k)(1+\log\|\mathbf{A}\|))$ LLL swaps, where $\|\mathbf{A}\|$ is the maximum of the Euclidean norm of all rows and columns of the matrix $\mathbf{A}$.*

Proof. From Proposition 4.1, the LLL algorithm allows to obtain a LLL-reduced basis for $\mathcal{L}^\perp(\mathbf{A})$. We know from Theorem 3.3 that in order to obtain an upper bound on the number of LLL swaps, it suffices to find an upper bound to $\Pi_k(\mathbf{B})$ and a lower bound on $\Pi_k(\mathbf{B}')$, where $\mathbf{B}'$ is the basis returned by LLL when given $\mathbf{B}$ as input. From (11) we have

$$\begin{aligned}
\Pi_k(\mathbf{B}) &= \sum_{j=1}^{k}(k-j)\log\|\boldsymbol{b}^*_{\ell_j}\| - \sum_{i=1}^{n-k} i\log\|\boldsymbol{b}^*_{s_i}\| + \sum_{i=1}^{n-k} s_i \\
&\leq \sum_{j=1}^{k}(k-j)\log\|\boldsymbol{b}^*_{\ell_j}\| + \sum_{i=1}^{n-k} s_i \\
&\leq \sum_{j=1}^{k}(k-j)\log\|\boldsymbol{b}_{\ell_j}\| + \sum_{i=1}^{n-k}(k+i) \\
&\leq (1+\log K + \log\|\mathbf{A}\|)\frac{k(k-1)}{2} + \frac{(n-k)(n+k+1)}{2}.
\end{aligned}$$

Thanks to Proposition 4.2, we have

$$\begin{aligned}
\Pi_k(\mathbf{B}') &= \sum_{j=1}^{k}(k-j)\log\|\boldsymbol{b}'^*_{\ell'_j}\| - \sum_{i=1}^{n-k} i\log\|\boldsymbol{b}'^*_{s'_i}\| + \sum_{i=1}^{n-k} s'_i \\
&= \sum_{j=1}^{k}(k-j)\log\|\boldsymbol{b}'^*_{n-k+j}\| - \sum_{i=1}^{n-k} i\log\|\boldsymbol{b}'^*_{i}\| + \sum_{i=1}^{n-k} i.
\end{aligned}$$

Since the first k coefficients of $\boldsymbol{b}'^*_i$ are 0 (for $i \leq n-k$) and $\mathbf{A}$ is full-rank, we must have $\|\boldsymbol{b}'^*_{n-k+1}\| \geq K$. Further, since $\mathbf{B}'$ is LLL-reduced, combining with (4) we have, for $j \leq k$

$$\|\boldsymbol{b}'^*_{n-k+j}\| \geq 2^{\frac{1-j}{2}}\|\boldsymbol{b}'^*_{n-k+1}\| \geq 2^{\frac{1-j}{2}}K \geq 2^{\frac{1-k}{2}}K.$$

We hence obtain

$$\Pi_k(\mathbf{B}') \geq \left(\log K + \frac{1-k}{2}\right)\sum_{j=1}^{k}(k-j) - \sum_{i=1}^{n-k} i \log \|\boldsymbol{b}_i'^*\| + \frac{(n-k)(n-k+1)}{2}$$
$$\geq \frac{k(k-1)}{2}\left(\log K + \frac{1-k}{2}\right) - (n-k)\sum_{i=1}^{n-k}\log\|\boldsymbol{b}_i'^*\| + \frac{(n-k)(n-k+1)}{2},$$

where we used the fact that all $\|\mathbf{b}_i'^*\|$'s are ≥ 1. This is true for the $\|\mathbf{b}_i^*\|$'s and LLL cannot make the minimum Gram-Schmidt norm decrease. Using (3), we obtain:

$$\Pi_k(\mathbf{B}') \geq \frac{k(k-1)}{2}\left(\log K + \frac{1-k}{2}\right) - (n-k)k\log\|\mathbf{A}\| + \frac{(n-k)(n-k+1)}{2}.$$

Finally, using Theorem 3.3, we obtain that the number of LLL swaps is no greater than

$$\frac{\Pi_k(\mathbf{B}) - \Pi_k(\mathbf{B}')}{\log\left(\frac{2}{\sqrt{3}}\right)} \leq \frac{k(n-\frac{k}{2})\log\|\mathbf{A}\| + k^3 + (n-k)k}{\log\left(\frac{2}{\sqrt{3}}\right)},$$

which is of $\mathcal{O}(k^3 + k(n-k)(1+\log\|\mathbf{A}\|))$. □

From Theorem 4.3, the number of LLL swaps required by the LLL algorithm for computing an LLL-reduced basis for $\mathcal{L}^{\perp}(\mathbf{A})$ does not grow with K when K is sufficiently large, as opposed to what the classical LLL potential provides.

Thanks to [2, Lemma 2], we have $\det(\mathbf{B}^T\mathbf{B}) \leq (1+K^2\|\mathbf{A}\|^2)^k$, and hence

$$\Pi(\mathbf{B}) \leq \log\prod_{i\leq n}\left(1+K^2\|\mathbf{A}\|^2\right)^{\frac{\min(k,i)}{2}} \leq \frac{k(2n-k+1)}{2}\log\left(2K\|\mathbf{A}\|\right),$$

where $\Pi(\mathbf{B})$ is the classical potential function given as in (6). As the classical potential of the output basis is ≥ 0 (by integrality), the bound on the number of LLL swaps obtained using the classical potential is $\mathcal{O}(k(n-k/2)(1+\log K+\log\|\mathbf{A}\|)$.

In Table 1, we use both the all-purpose (9) and the heuristic (10) to set K (the first choice always works, while the second works for many $\mathbf{A}$'s). From this table, we observe that our new analysis is worse only if $k = n-1$ and $\log\|\mathbf{A}\| = o(n)$. But if $k = n-1$, the problem then becomes finding the unique nonzero integer vector $\boldsymbol{m}$ such that $\mathbf{A}\boldsymbol{m} = \mathbf{0}$. To solve this problem, one may use some other techniques from linear algebra, instead of LLL-reducing the basis of form (2).

Table 1: Upper bounds on the number of LLL swaps (K sufficiently large) for different k

	$k=1$	$k=n/2$	$k=n-1$
Classical analysis using (9)	$\mathcal{O}(n^2\log n + n\log\|\mathbf{A}\|)$	$\mathcal{O}(n^3\log n + n^3\log\|\mathbf{A}\|)$	$\mathcal{O}(n^2\log\|\mathbf{A}\|)$
Heuristic analysis using (10)	$\mathcal{O}(n^2 + n\log\|\mathbf{A}\|)$	$\mathcal{O}(n^3 + n^2\log\|\mathbf{A}\|)$	$\mathcal{O}(n^2\log\|\mathbf{A}\|)$
New analysis	$\mathcal{O}(n\log\|\mathbf{A}\|)$	$\mathcal{O}(n^3 + n^2\log\|\mathbf{A}\|)$	$\mathcal{O}(n^3 + n\log\|\mathbf{A}\|)$

References

[1] J. Chen, D. Stehlé, and G. Villard. A new view on HJLS and PSLQ: Sums and projections of lattices. In M. Kauers, editor, *Proceedings of ISSAC '13 (June 26-29, 2013, Boston, MA, USA)*, pages 149–156. ACM, New York, 2013. 2

[2] G. Hanrot. LLL: A tool for effective Diophantine approximation. In P. Q. Nguyen and B. Vallée, editors, *The LLL Algorithm: Survey and Applications*, pages 215–263. Springer, Berlin, 2010. 10

[3] J. Håstad, B. Just, J. C. Lagarias, and C.-P. Schnorr. Polynomial time algorithms for finding integer relations among real numbers. *SIAM Journal of Computing*, 18(5):859–881, 1989. Erratum: SIAM J. Comput., 43(1), 254–254, 2014. 2

[4] G. Havas, B. S. Majewski, and K. R. Matthews. Extended GCD and Hermite normal form algorithms via lattice basis reduction. *Experimental Mathematics*, 7(2):125–136, 1998. 2

[5] R. Kannan, A. K. Lenstra, and L. Lovász. Polynomial factorization and nonrandomness of bits of algebraic and some transcendental numbers. In R. A. DeMillo, editor, *Proceedings of the 16th Annual ACM Symposium on Theory of Computing (April 30 - May 2, 1984, Washington, DC, USA)*, pages 191–200. ACM, New York, 1984. 2

[6] A. K. Lenstra, H. W. Lenstra, and L. Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261(4):515–534, 1982. 1, 3, 4

[7] A. Neumaier and D. Stehlé. Faster LLL-type reduction of lattice bases. In S. A. Abramov, E. V. Zima, and X.-S. Gao, editors, *Proceedings of ISSAC '16 (July 20–22, 2016, Waterloo, Ontario, Canada)*, pages 373–380. ACM, New York, 2016. 2

[8] P. Q. Nguyen. *La Géométrie des Nombres en Cryptologie*. PhD thesis, Université Paris 7, Paris, 1999. 3

[9] P. Q. Nguyen and B. Vallée, editors. *The LLL Algorithm: Survey and Applications*. Springer, Berlin, 2010. doi: 10.1007/978-3-642-02295-1. 1, 2

[10] A. Novocin, D. Stehlé, and G. Villard. An LLL-reduction algorithm with quasi-linear time complexity: extended abstract. In L. Fortnow and S. P. Vadhan, editors, *Proceedings of STOC '11 (June 6–8, 2011, San Jose, USA)*, pages 403–412. ACM, New York, 2011. 2

[11] M. E. Pohst. A modification of the LLL reduction algorithm. *Journal of Symbolic Computation*, 4(1):123–127, 1987. 2

[12] A. Storjohann. Faster algorithms for integer lattice basis reduction. Technical Report 249, ETH, Department of Computer Scicence, Zürich, Switzerland, July 1996. 2