Technische Universität Darmstadt
Department of Computer Science
Cryptography and Computeralgebra

Bachelor Thesis

Lattice Basis Reduction in Infinity Norm

Vanya Sashova Ivanova

Technische Universität Darmstadt
Department of Mathematics

Supervised by Prof. Dr. Johannes Buchmann
Markus Rückert

Acknowledgements

First, and foremost, I would like to thank Prof. Dr. Johannes Buchmann for giving me the opportunity to write this thesis. I am deeply grateful to my direct supervisor, Markus Rückert, for his detailed and constructive remarks, and for all his help and support throughout my work.

Warranty

I hereby warrant that the content of this thesis is the direct result of my own work and that any use made in it of published or unpublished material is fully and correctly referenced.

Date: ..... Signature: .....

Abstract

In the high-tech world of today, the demand for security is constantly rising. That is why identifying hard computational problems for cryptographical use has become a very important task. It is crucial to find computational problems, which complexity could provide a basis for the security of the cryptosystems. However, there are only very few hard computational problems that are useful for cryptography. One of these, which holds great importance, is finding the shortest basis of a lattice, also called lattice basis reduction. The purpose of this paper is to provide an overview of the lattice basis reduction algorithms and to give an insight into the lattice reduction theory. The most important concepts, Gauss, LLL and BKZ reduction, were initially created to work with the Euclidean norm. Here however, the accent falls on the generalisation of the original algorithms to an arbitrary norm, and more precicely to the infinity norm. All three concepts in their three versions are explained in detail. An analysis of the complexity of the algorithms with respect to l_∞-norm is provided.

Contents

1 Introduction

Lattices are discrete Abelian subgroups of $\mathbb{R}^n$. The goal of lattice reduction is to find interesting lattice bases, such as bases consisting of reasonably short and almost orthogonal vectors. Lattice basis reduction algorithms are used in quite a few modern number-theoretical applications, as for example the discovery of the spigot algorithm for π. They hold a very important meaning especially for present-day cryptography. The security proofs of many effective cryptosystems, for example public-key encryptions or collision-resistant hash functions, are based on lattice reduction worst-case complexity, relatively efficient implementations, and great simplicity. In recent years, methods based on lattice reduction are being used repeatedly for the cryptanalytic attack on various systems.

This paper introduces the three most important algorithms for lattice basis reduction. All of the original concepts are based on the Euclidean norm and later generalized to an arbitrary norm. The emphasis here falls on the extension of these algorithms with respect to l_∞-norm. The infinity norm is essential for further research of lattice reduction since solving the shortest lattice basis problem with respect to l_∞ would allow, for example, breaking the Knapsack cryposystems or solving the well-known subset sum problem.

The theory of lattice reduction could be traced back to Lagrange, Gauss and Dirichlet. In the mathematical society today there is a dispute as to who was the first mathematician to study lattices and their basis reduction theory. Some argue that Lagrange started it by developing the reduction theory of binary quadratic forms. Others believe that lattice reduction per se actually started with Gauss who created an algorithm - the Gauss Algorithm, which solves the problem using what resembles a lifting of the Euclidean Great Common Divisor algorithm to 2-dimensional lattices. This Euclidean norm algorithm works in polynomial time with respect to its input. It's worst case complexity was first studied by Lagrange and the analysis was later continued and more precisely explained by Vallée [1] [12]. This oldest algorithm is developed only for low dimensions since they are a lot more intuitive for the human mind to understand.

A successful, efficient, but week algorithm that works for higher dimensions was introduced by A.K.Lenstra, H.W. Lenstra and L.Lovász [6]. This Euclidean norm algorithm is called LLL algorithm after the authors. It works with any basis in polynomial time with respect to the input when the number of integer variables is fixed. The result is however only an approximation of the shortest lattice vector. Schnorr [10] [9] [11] developed several LLL-type algorithms that work better than the original and defined the hierarchy of the polynomial time algorithms. He also introduced a new type of reduction by combining the Lovász and Hermite's definitions to obtain a flexible order of concepts - the so called block reduction. He developed a

brand new Euclidean norm algorithm that works very well in practice - the Block-Korkine-Zolotarev algorithm or BKZ-algorithm.

Kaib and Schnorr [4] [2] developed recently the Generalized Gauss Algorithm for the reduction of two dimensional lattice bases with respect to the arbitrary norm and obtained a universally sharp upper bound on the number of its iterations for all norms. Lovász and Scarf [8] created the LS algorithm that is in a way an extension of the LLL-algorithm to an arbitrary norm. Kaib and Ritter [3] generalized the strong and efficient concept of block reduction.

This paper begins with an introduction into the basic mathematical background so that the reader could follow the idea behind the reduction theory easily. It is then divided into three main parts, one for each of the three most important algorithms - Gauss, LLL and BKZ. Each of the three parts presents first the original concept with respect to the Euclidean norm, then their extension to an arbitrary norm and particularly to the infinity norm. Each of the sections is concluded with an analysis of the complexity of the algorithms with respect to l_∞-norm.

2 Mathematical Background

This section gives a basic overview of the mathematical foundations of the lattice basis reduction theory.

2.1 Basic Definitions

In this paper $\mathbb{R}^n$ denotes the n-dimensional real vector space with an inner product $\langle \cdot, \cdot \rangle : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}^n$. For all $\mathbf{u}, \mathbf{v}, \mathbf{w} \in \mathbb{R}^n$ and $\lambda \in \mathbb{R}$ the inner product has the following properties:

- $\langle \mathbf{u} + \mathbf{w}, \mathbf{v} \rangle = \langle \mathbf{u}, \mathbf{v} \rangle + \langle \mathbf{w}, \mathbf{v} \rangle$,
- $\langle \lambda \mathbf{u}, \mathbf{v} \rangle = \lambda \langle \mathbf{u}, \mathbf{v} \rangle$,
- $\langle \mathbf{u}, \mathbf{v} + \mathbf{w} \rangle = \langle \mathbf{u}, \mathbf{v} \rangle + \langle \mathbf{u}, \mathbf{w} \rangle$,
- $\langle \mathbf{u}, \lambda \mathbf{v} \rangle = \lambda \langle \mathbf{u}, \mathbf{v} \rangle$,
- $\langle \mathbf{u}, \mathbf{v} \rangle = \langle \mathbf{v}, \mathbf{u} \rangle$,
- $\langle \mathbf{u}, \mathbf{u} \rangle > 0$ for $\mathbf{u} \neq 0$.

A mapping $\| \cdot \| : \mathbb{R}^n \rightarrow \mathbb{R}$ is called a norm if for all $\mathbf{u}, \mathbf{v} \in \mathbb{R}^n$ and $\lambda \in \mathbb{R}$

- $\| \lambda \mathbf{v} \| = |\lambda| \cdot \| \mathbf{v} \|$,
- $\| \mathbf{u} + \mathbf{v} \| \leq \| \mathbf{u} \| + \| \mathbf{v} \|$,
- $\| \mathbf{u} \| \geq 0$ for $\mathbf{u} \neq 0$.

The l_1-norm is

$$\| (u_1, u_2, ..., u_n)^T \|_1 := \sum_{i=1}^{n} |u_i|.$$

The l_2-norm is computed with the help of the inner product

$$\| (u_1, u_2, ..., u_n)^T \|_2 := \sqrt{\langle u, u \rangle} = \sqrt{\left(\sum_{i=1}^{n} u_i^2\right)}.$$

The l_p-norm, $p \in [1, \infty]$ could be generalized to

$$\| (u_1, u_2, , u_n)^T \|_p := \sqrt{\left(\sum_{i=1}^{n} |u_i|^p\right)}.$$

The l_∞-norm or the maximum-norm is then

$$\| (u_1, u_2, ..., u_n)^T \|_\infty := \max_{i=1,2,...,n} |u_i|.$$

It holds for the l_1, l_2 and l_∞-norm:

- $\| u \|_2 \leq \| u \|_1 \leq \sqrt{n} \cdot \| u \|_2$,

- $\| u \|_\infty \leq \| u \|_2 \leq n \cdot \| u \|_\infty$.

Definition 2.1. Let $\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_m \in \mathbb{R}^n$ be an ordered basis of a lattice L. The orthogonal projection of this basis is then

$$\pi_i \ : \ \mathbb{R}^n \ \rightarrow \ span(\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_{i-1})^\perp,$$

$$\mathbf{b} \ - \ \pi_i(\mathbf{b}) \ \in \ span(\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_{i-1}).$$

Let $\hat{\mathbf{b}}_i \ := \ \pi_i(\mathbf{b}_i)$. The vectors $\hat{\mathbf{b}}_1, \hat{\mathbf{b}}_2, ..., \hat{\mathbf{b}}_m$ could be computed with the help of the Gram-Schmidt algorithm:

$$\hat{\mathbf{b}}_1 \ := \ \mathbf{b}_1,$$

$$\hat{\mathbf{b}}_i \ := \ \pi_i(\mathbf{b}_i) \ = \ \mathbf{b}_i \ - \ \sum_{j=1}^{i-1} \mu_{i,j} \hat{\mathbf{b}}_j, \text{ for } i \ = \ 2, 3, ..., m.$$

$\mu_{i,j}$ denote the Gram-Schmidt coefficients:

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

For $j \ > \ i$ it holds that $\mu_{i,j} \ = \ 0$ and $\mu_{i,i} \ = \ 1$. The orthogonal projection of $\mathbf{b}_i$ in $span(\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_k)$ is $\sum_{j=1}^{k} \mu_{i,j} \hat{\mathbf{b}}_j$ and in $span(\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_k)^\perp$ is $\sum_{j=k+1}^{i} \mu_{i,j} \hat{\mathbf{b}}_j$. The basis vectors could therefore be presented in the following manner

$$[\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_m] \ = \ [\hat{\mathbf{b}}_1, \hat{\mathbf{b}}_2, ..., \hat{\mathbf{b}}_m] \cdot [\mu_{i,j}]_{1 \leq i,j \leq m}^T.$$

For the matrix B consisting of the basis vectors $[\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_m]$ the quadratic form $QF_B(x_1, x_2, ..., x_m)$, where $x_1, x_2, ..., x_m$ are real variables, is equal to $\sum_{1 \leq i,j \leq m} \langle \mathbf{b}_i, \mathbf{b}_j \rangle \, x_i x_j$. This quadratic form is positive definite, therefore $QF_B(x_1, x_2, ..., x_m) \ \geq \ 0$ and $QF_B(x_1, x_2, ..., x_m) \ = \ 0$ if and only if $x_1 \ = \ x_2 \ = ... = \ x_m \ = \ 0$. For every positive definite, symmetrical quadratic form $QF \ = \ \sum_{1 \leq i,j \leq n} q_{ij} x_i x_j$ there is a lattice basis $\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_m$ such that $\langle \mathbf{b}_i, \mathbf{b}_j \rangle \ = \ q_{ij}$ for $1 \ \leq \ i, j \ \leq \ m$.

2.2 Lattices and Successive Minima

In the real vector space $\mathbb{R}^n$ a lattice L is a discrete Abelian subgroup. By discrete here we mean that any bounded subset of $\mathbb{R}^n$ contains at most finitely many lattice elements.

Definition 2.2. Let $\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_m \in \mathbb{R}^n$ be linearly independent vectors. Let L be a subgroup of $\mathbb{R}^n$

$$L(\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_m) := \sum_{i=1}^{m} \mathbf{b}_i t_i, \text{ where } t_1, t_2, , t_m \in \mathbb{Z}$$

is called a lattice with basis $(\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_m)$.

Definition 2.3. The rank or the dimension of a lattice L is the number of vectors that form the basis of this lattice.

Definition 2.4. The determinant of a lattice $L \subseteq \mathbb{R}^n$ with a basis $\mathbf{B} = (\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_m)$ is defined as:

$$\det L = \sqrt{(\det \mathbf{B}^T \mathbf{B})}$$

Theorem 2.5. *The determinant of a lattice is independent on the choice of the basis* $\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_m \in \mathbb{R}^n$.

Definition 2.6. The vectors $\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_k \in L$, where $L \subseteq \mathbb{R}^n$ is a lattice, build a primitive system if:

1. $\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_k$ are linear independent,
2. $span(\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_k) \cap L = L(\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_k)$.

Theorem 2.7. *In a lattice* $L \subseteq \mathbb{R}^n$ *the vectors* $\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_k \in L$ *are a basis of this lattice only when they build a primitive system with respect to* L.

Definition 2.8. Let $\| \cdot \|_p$ be an arbitrary norm. For a lattice $L \subseteq \mathbb{R}^n$ of rank m the successive minima $\lambda_1, \lambda_2, ..., \lambda_m$ with respect to the norm are defined as follows:

$$\lambda_i = \lambda_i(L) := \inf(r > 0 | \mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_i \in L \text{ are linearly independent with } \| \mathbf{c}_j \|_p \leq r \text{ for } j = 1, 2, ..., i)$$

The successive minima are geometrical invariants of the lattice, which means that when isometric transformations are done in the lattice, the successive minima stay unchanged. It holds that $\lambda_1 = \leq \lambda_2 \leq .... \leq \lambda_m$. For every lattice basis $\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_m$

$$\lambda_i \leq \max_{j=1,2,...,i} \| \mathbf{b}_j \|_p \text{ for } i = 1, 2, ..., m.$$

The successive minima are used as a criterion to check when a lattice basis is reduced - a basis is reduced when $\frac{\|\mathbf{b}_i\|_p}{\lambda_i}$, for $i = 1, 2, ..., m$, are "small". For the lattice basis to be reduced the vectors should be almost orthogonal. In general there is no basis $\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_m$ for which $\| \mathbf{b}_i \|_p = \lambda_i$ for $i = 1, 2, ..., 5$. The successive minima depend on the norm that is being used. For example the first successive minimum for the Euclidean norm and the standard inner product is

$$\| L \|_2 := \lambda_{1,2}(L) = \min \{ \| \mathbf{b} \|_2 : \mathbf{b} \in L \, 0 \}.$$

And the first successive minimum in l_∞-norm is

$$\| L \|_\infty := \lambda_{1,\infty}(L) = \min \{ \| \mathbf{b} \|_\infty : \mathbf{b} \in L \, 0 \}.$$

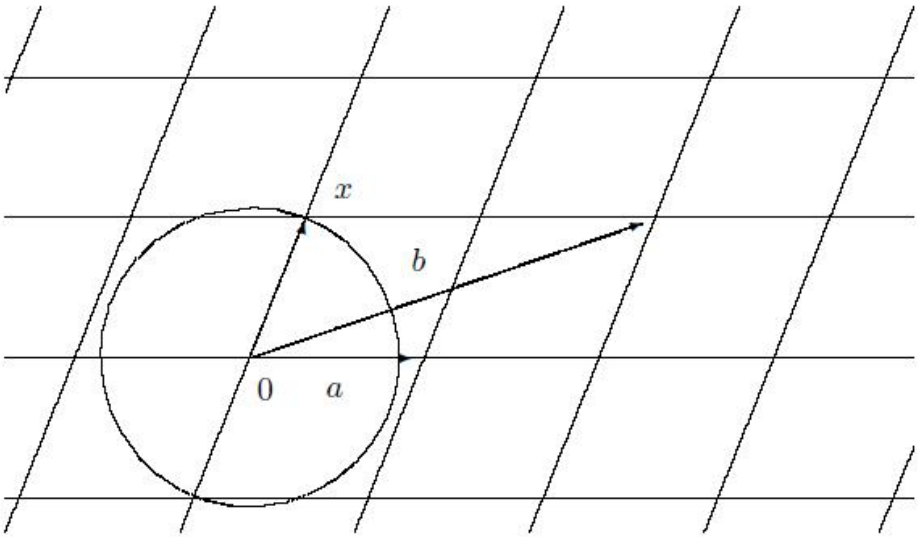


Figure 1: First Successive Minimum

An example of a first successive minimum $\lambda_1(L(\mathbf{a}, \mathbf{b})) = x$ is the figure above.

Since for every lattice $L \subseteq \mathbb{R}^n$ $\| \mathbf{x} \|_\infty \leq \| \mathbf{x} \|_2 \leq \sqrt{m} \| \mathbf{x} \|_\infty$ for all $\mathbf{x} \in \mathbb{R}^n$, it holds for the successive minima that

$$\lambda_{1,\infty}(L) \leq \lambda_{1,2}(L) \leq \sqrt{m}\lambda_{1,\infty}(L)$$

Theorem 2.9. *For each lattice $L \subseteq \mathbb{R}^n$ of rank m, $\| L \|_\infty \leq (\det L)^{\frac{1}{m}}$*

2.3 Distance Functions

The term "Distance Function" was integrated into the lattice reduction theory by Lovász and Scarf.

Definition 2.10. Let $\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_m \in \mathbb{R}^n$ be linearly independent vectors for all $m \leq n$ and let $\| \cdot \|_p$ be an arbitrary norm on $\mathbb{R}^n$. The functions $F_i : \mathbb{R}^n \to \mathbb{R}$ with

$$F_i(\mathbf{x}) := \min_{\xi_1,....,\xi_{i-1} \in \mathbb{R}} \| \mathbf{x} + \sum_{j=1}^{i-1} \xi_j \mathbf{b}_j \|_p \text{ for } 1 \leq i \leq m + 1$$

are called distance functions.

The distance functions determine the distance between a vector $\mathbf{b}_i$ and $span(\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_{i-1})$ with respect to an arbitrary norm. The functions $F_i(\mathbf{x})$ are actually the norm of $span(\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_{i-1})^\perp$. With respect to the Euclidean norm they are also Euclidean norms of the subspace. The distance functions are very useful because the length of the shortest lattice vector could be restricted by them.

Theorem 2.11. *For every basis* $\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_m$ *of a lattice* $L \subset \mathbb{R}^n$ *and every norm* $\| \cdot \|_p$

$$\min_{i=1,....,n} F_i(\mathbf{b}_i) \ \leq \ \lambda_{1,p}(L)$$

3 Gauss' Algorithm

The first person to develop an algorithm for lattice basis reduction was Gauss. This first algorithm generalizes the Euclidean algorithm. It was initially described within the general framework of binary integer quadratic forms by Gauss himself and was later specified in terms of lattice basis reduction by Vallée [12]. It is the first to be considered in this paper not only because it is the first to have a polynomial-time complexity but also because it works in the 2-dimensional space and thus gives an understanding of the higher dimensions.

Primarily, the algorithm was created to work with the Euclidean norm. Later Schnorr and Kaib [4] extended the concept to an arbitrary norm. In the dissertation of Kaib [2] a detailed explanation of the generalized algorithm is given as well as an extension, that is specifically useful for the infinity norm.

In this chapter we give a short explanation of what reduced lattice bases in the Gaussian sense mean and describe the original Gauss Algorithm, which works with the Euclidean norm. This algorithm is then enhanced to work with arbitrary norms in the Generalized Gauss Algorithm. An extension for the l_∞-norm and its time complexity is finally presented in the last sub-chapter.

3.1 Gauss - reduced Bases

Since we give a short description for both the original and the generalized Gauss algorithm here, in this section $\| \cdot \|_p$, for $p \in [1, \infty]$, will denote any norm.

Definition 3.1. An ordered basis $(\mathbf{a}, \mathbf{b}) \in \mathbb{R}^n$ is Gauss-reduced with respect to a norm $\| \cdot \|_p$ when

$$\| \mathbf{a} \|_p \leq \| \mathbf{b} \|_p \leq \| \mathbf{a} + \mathbf{b} \|_p.$$

When considering the standard inner product for the Gram-Schmidt coefficient μ and the Euclidean norm we have:

1. $\mu_{2,1} \leq \frac{1}{2} \Leftrightarrow \| \mathbf{b} \|_2 \leq \| \mathbf{a} - \mathbf{b} \|_2$,
2. $\mu_{2,1} \geq 0 \Leftrightarrow \| \mathbf{a} - \mathbf{b} \|_2 \leq \| \mathbf{a} + \mathbf{b} \|_2$.

Therefore the basis $(\mathbf{a}, \mathbf{b})$ is reduced if and only if

1. $\| \mathbf{a} \|_2 \leq \| \mathbf{b} \|_2$ and
2. $0 \leq \mu_{2,1} \leq \frac{1}{2}$.

Theorem 3.2. *For a reduced basis* $(\mathbf{a}, \mathbf{b}) \in \mathbb{R}^n$, $\| \mathbf{a} \|_p$ *and* $\| \mathbf{b} \|_p$ *are the two successive minima of the lattice* $L = \mathbb{Z}\mathbf{a} + \mathbb{Z}\mathbf{b}$.

Definition 3.3. An ordered basis $(\mathbf{a}, \mathbf{b}) \in \mathbb{R}^n$ is called well-ordered with respect to a norm $\| \cdot \|_p$ when

$$\| \mathbf{a} \|_p \leq \| \mathbf{a} - \mathbf{b} \|_p \leq \| \mathbf{b} \|_p.$$

The transition from an order basis to a well-ordered one could be done in few steps. We know that for the ordered basis $(\mathbf{a}, \mathbf{b})$, $\| \mathbf{a} \|_p \leq \| \mathbf{b} \|_p \leq$ $\leq \| \mathbf{a} + \mathbf{b} \|_p$. If $\| \mathbf{a} - \mathbf{b} \|_p > \| \mathbf{a} + \mathbf{b} \|_p$, let $\mathbf{b} := -\mathbf{b}$. If $\| \mathbf{a} \|_p > \| \mathbf{a} - \mathbf{b} \|_p$, $(\mathbf{b} - \mathbf{a}, -\mathbf{a})$ is the well-ordered basis for this lattice.

3.2 Gauss Algorithm with Euclidean Norm

The original version of the Gaussian algorithm works with the Euclidean norm. It takes as an input an arbitrary basis and produces as an output a basis that is reduced. In each iteration $\mathbf{b}$ is reduced according to $\mathbf{b} := \mathbf{b} - \lceil \mu_{2,1}\mathbf{a} \rfloor$. In other words, we make $\mathbf{b}$ as short as possible by subtracting a multiple of $\mathbf{a}$. At the end if $(\mathbf{a}, \mathbf{b})$ is not reduced, $\mathbf{a}$ and $\mathbf{b}$ are swapped.

Algorithm 1.1 Gauss Reduction Algorithm for Euclidean Norm

INPUT: A lattice basis $(\mathbf{a}, \mathbf{b}) \in \mathbb{R}^n$ such that $\| \mathbf{a} \|_2 \leq \| \mathbf{b} \|_2$

1. **WHILE** $|\mu_{2,1}| > \frac{1}{2}$ **DO**
   - $\mathbf{b} := \mathbf{b} - \mu\mathbf{a}$, where the integer μ is chosen to minimize the norm $\| \mathbf{b} - \mu\mathbf{a} \|_p$.
   - **IF** $\| \mathbf{a} \|_2 > \| \mathbf{b} \|_2$ **THEN** swap $\mathbf{a}$ and $\mathbf{b}$
2. **END WHILE**
3. **DO** $\mathbf{b} := \mathbf{b} \cdot sign(\mu_{2,1})$

OUTPUT: A reduced basis $(\mathbf{a}, \mathbf{b}) \in \mathbb{R}^n$

3.3 Generalized Gauss Algorithm

The generalized Gaussian algorithm finds the two successive minima for an arbirtary norm. Given a reduced basis, we provide a minimum size input basis requiring a given number of iterations. The sign of the basis vector is chosen in such a way that the algorithm iterates on a well-ordered basis. As a result all integral reduction coefficients μ are positive.

Algorithm 1.2 Generalized Gauss Algorithm

INPUT: A well-ordered lattice basis $(\mathbf{a}, \mathbf{b}) \in \mathbb{R}^n$

1. **WHILE** $\| \mathbf{b} \|_p > \| \mathbf{a} - \mathbf{b} \|_p$**DO**
   - $\mathbf{b} := \mathbf{b} - \mu\mathbf{a}$, where the integer μ is chosen to minimize the norm $\| \mathbf{b} - \mu\mathbf{a} \|_p$.
   - **IF**$\| \mathbf{a} + \mathbf{b} \|_p > \| \mathbf{a} - \mathbf{b} \|_p$ **THEN** $\mathbf{b} := -\mathbf{b}$.

- Swap **a** and **b**.

2. **END WHILE**

OUTPUT: A reduced basis $(\mathbf{a}, \mathbf{b}) \in \mathbb{R}^n$

The exchange in Step 3 produces either a well-ordered or a reduced basis. The algorithm traverses a sequence of well-ordered bases until a reduced one is found. In order to have a well defined algorithm in Step 1 the smallest possible μ, that minimizes the norm $\| \mathbf{b} - \mu\mathbf{a} \|_p$ for $p \in [1, \infty]$, should be chosen. In general this algorithm terminates after finitely many steps because the norm of the basis vector decreases with every iteration, except the last one.

3.4 Infinity Norm Algorithm

As Kaib explains and proves in his dissertation [2] there is a fast way to calculate the coefficient μ for l_∞-norm. This section describes the method proposed by him.

Let us consider the function $f(\mu) = \| \mathbf{b} - \mu\mathbf{a} \|_\infty$. It is a piecewise linear and convex function with at most n corners. Its minimum, according to the optimization theory, could be found on one of these corners. The reduction coefficient μ is therefore one of the two neighboring whole numbers of the corner that minimizes the function f. In order to find that minimum, the elements of the function $f(\mu) = \max f_i(\mu) | 1 \leq i \leq n$ should be taken.

Let us consider the graph of the function $f(x) = \max_{i \leq n} |b_i - xa_i|$, which is the maximal polygon. The ascending slope of the function is $f_i^+(x) := |a_i| (x - \frac{b_i}{a_i})$ and the descending - $f_i^-(x) := |a_i| (\frac{b_i}{a_i} - x)$, where all elements with $a_i = 0$ are already eliminated. All elements are sorted according to their absolute value, $|a_1| \geq |a_2| \geq .... \geq |a_n| > 0$ without loss of generality. Let $f^{(k)}(x) := \max_{i \leq k} |(b_i - xa_i)|$ be the maximal polygon that has the k steepest elements. For each $k = 1, 2, ..., n$ we compute the value of $f^{(k)}$ as an ordered subset of lines, which build the maximal polygon. The faces of the polygon are determined from these lines and from them $x^{(k)}$, which minimizes $f^{(k)}$, is calculated.

From the elements $f_{k+1} = \max (f_{k+1}^+, f_{k+1}^-)$ only the ones with the biggest $f_{k+1}^{\pm}(x^{(k)})$ values are to be considered. If $f_{k+1}(x^{(k)}) \leq$
$\leq f^{(k)}(x^{(k)})$, set $f^{(k)} = f^{(k+1)}$. In case there are more elements with the same slope, only the ascending one with the smallest $\frac{b_{\min}}{a_{\min}}$ and the descending one with the biggest $\frac{b_{\max}}{a_{\max}}$ are considered.

We divide the line segments of $f^{(k)}$ in two groups - R and L, where R contains the ascending and L - the descending segments. The absolute value of the slope is minimal for both groups. The value $x = x^{(k)}$ denotes the minimal point of $f^{(k)}$, that is the crossing point of the topmost line from both groups. The biggest element from the groups is denoted by (g_L, xL)

and (g_R, xR) respectively. The crossing point of the two lines is then $g_1 \cap g_2 = \frac{b_1 - b_2}{a_1 - a_2}$.

We initialize R with $(g^+, +\infty)$, L with $(g^-, -\infty)$ and $x := g^+ \cap g^-$. We iterate over all of the line, starting with the one with the steepest slope. Since the two cases, with ascending and descending lines, respectively, are symmetric, here only an algorithm only for the ascending case is given.

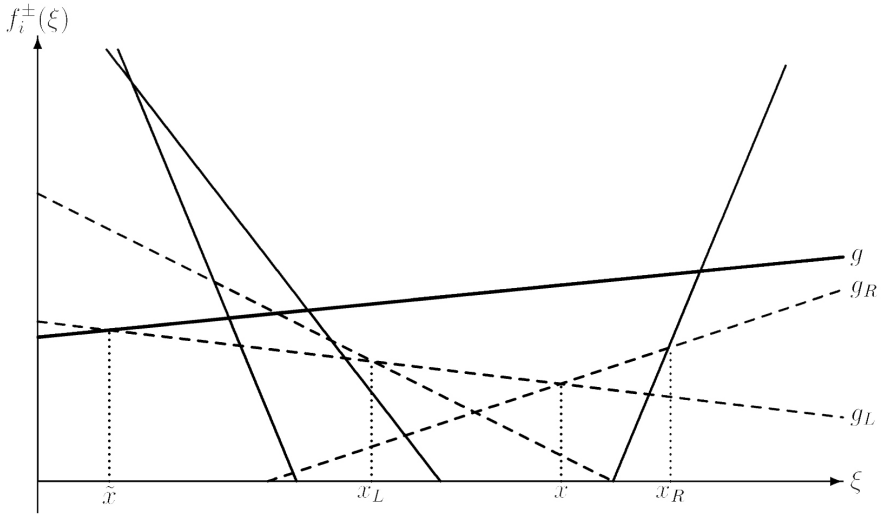
Algorithm 1.3 Infinity Norm

INPUT: A well-ordered basis $(\mathbf{a}, \mathbf{b}) \in \mathbb{R}^n$, $0 \leq x \leq \frac{1}{2}$

1. $\bar{x} := g \cap g_L$
2. If $\bar{x} \geq x$, go to the next step
3. As long as $\bar{x} \leq x_L$ do $[POP(L, \bar{x} := g \cap g_L]$ (deletion of an element)
4. $x := \bar{x}$
5. $\hat{x} := g \cap g_R$
6. As long as $\hat{x} \geq x_R$ do $[POP(R, \hat{x} := g \cap g_R]$
7. PUSH $(R, (g, \hat{x}))$ (insertion of an element)

OUTPUT: x that minimizes $f(x) = \| \mathbf{b} - x\mathbf{a} \|_\infty$

Consider the following figure:



This figure shows part of the graph of the function $f(\mu) = \| \mathbf{b} - \mu\mathbf{a} \|_\infty$. The continous lines on the graph are the ones that form the polygon of the function $f(x)$. The punctuated lines are to be deleted by the POP function.

3.5 Time Bounds

The time bound of the generalized Gauss algorithm has been proven by Schnorr and Kaib [4]. They use the normal arithmetic operations (addition, multiplication, division, subtraction, comparison and next integer computation) as unit costs and count arithmetical steps and oracle calls. It requires a norm oracle that produces $\| \mathbf{a} \|_p$ for a given $\mathbf{a} \in \mathbb{R}^n$, where $p \in [1, \infty]$.

Theorem 3.4. *Given an oracle for an arbitrary norm $\| \cdot \|_p$, for $p \in [1, \infty]$, there is an algorithm which $\| \cdot \|_p$-reduces a given basis $(\mathbf{a}, \mathbf{b}) \in \mathbb{R}^n$ using $O(n \log(n + \frac{\lambda_2}{\lambda_1}) + \log B)$ many steps, where $B = \max \frac{\|\mathbf{a}\|_p, \|\mathbf{b}\|_p}{\lambda_2}$.*

For efficiency, the basis is first reduced in a norm corresponding to a suitable inner product, that is the Euclidean norm. The resulting basis is subsequently reduced in the norm that we are interested in. The inner product is chosen so that $\{\mathbf{x} \in \mathbb{R}^n | \ \| \mathbf{x} \|_\infty \leq 1\}$ is spherical. The steps for producing the inner product are not counted since we assume they are given. The initial inner product reduction costs $O(1)$ arithmetical steps per iteration.

That gives a total of $O(n + \log B)$ arithmetical steps for the inner product reduction. In each iteration the Gram-Schmidt matrix is transformed and the transformation matrix H and the coefficient μ are produced. Each iteration requires 6 multiplications, 6 subtractions, 1 division and 1 next integer computation.

The final $\| \cdot \|_p$-reduction is done in $O(n + \log(n + \frac{\lambda_2}{\lambda_1}))$ many steps. As shown in the paper of Schnorr and Kaib this final reduction requires at most $\log_{1+\sqrt{2}}(2\sqrt{2} \max_{x,y \in \mathbb{R}^n} \frac{\|x\|_p \sqrt{<y,y>}}{\|y\|_p \sqrt{<x,x>}} + 1) + o(1) = O(\log n)$ many iterations. Every iteration besides the final one requires $O(1)$ norm computation and $O(n)$ arithmetical steps.

The whole algorithm, which computes a reduced basis $(\mathbf{a}, \mathbf{b})$ for an arbitrary norm, works in $O(n \log(n + \frac{\lambda_2}{\lambda_1}) + \log B)$ many steps, where $B = \max \frac{\|\mathbf{a}\|_p, \|\mathbf{b}\|_p}{\lambda_2}$.

In the case of l_∞-norm the following theorem holds:

Theorem 3.5. *For l_∞-norm there is an algorithm that reduces any given well-ordered basis $(\mathbf{a}, \mathbf{b})$ in $O(n \log n + \log B)$ many iterations, where $B = \frac{\|\mathbf{b}\|_\infty}{\lambda_2}$ and λ_2 is the second successive minima of the lattice.*

As presented in this section, the inner product reduction for l_∞-norm works also in $O(n + \log B)$ arithmetical steps.

The algorithm, shown in the previous section, needs $O(n \log n)$ arithmetical operation for sorting the elements. The cycle for sorting the lines costs at most $O(n)$ arithmetical operations. There is only one PUSH step for each element and at most $n - 1$ POP steps.

Hence the l_∞-norm reduction of a lattice basis $(\mathbf{a}, \mathbf{b}) \in \mathbb{R}^n$ takes at most $O(n \log n + \log B)$ arithmetical steps, where $B = \frac{\|\mathbf{b}\|_\infty}{\lambda_2}$ and λ_2 is the second successive minima of the lattice.

4 LLL

The subject of lattice basis reduction theory was revived in 1981 with H.W.Lenstra's work on integer programming, which is based on a novel lattice reduction technique and works in polynomial time only for fixed dimensions. Lovász later developed a version of the algorithm which computes a reduced basis of a lattice in polynomial time for any dimension. The LLL algorithm, named after its creators, reached its final version in the paper of A.K. Lenstra, H.W. Lenstra and L. Lovász [7], where it is applied to factor rational polynomials with respect to the Euclidean norm. It computes a reduced basis after polynomial many steps. Further enhancements were introduced later by Schnorr.
In 1992 Lovász and Scarf [8] proposed a generalized lattice reduction algorithm, which was later named after them - LS algorithm. They extend the meaning of an LLL-reduced basis to any norm. It works in polynomial many arithmetical operations and calculations of distance functions.
We begin this chapter with a short description of the mathematical meaning of LLL-reduced bases. The original LLL algorithm is then presented. This type of reduction is continued with the LS algorithm that works with an arbitrary norm. Finally, a specific enumeration for the infinity norm is introduced.

4.1 LLL-reduced Basis

Here we explain the meaning of an LLL-reduced basis and show the qualities of such a basis. Let $(\hat{\mathbf{b}}_1, \hat{\mathbf{b}}_2, ..., \hat{\mathbf{b}}_m)$ be the ordered orthogonal system for the basis $(\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_m) \in \mathbb{R}^n$ and $\mu_{i,j}$ the Gram-Schmidt coefficient.

Definition 4.1. An ordered lattice basis $(\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_m) \in \mathbb{R}^n$ is called LLL-reduced for a parameter δ, such that $\frac{1}{4} < \delta \leq 1$, when:

1. $|\mu_{i,j}| \leq \frac{1}{2}$ for $1 \leq j < i \leq m$
2. $\delta \cdot \| \hat{\mathbf{b}}_{k-1} \|_2^2 \leq \| \hat{\mathbf{b}}_k \|_2^2 + \mu_{k,k-1}^2 \cdot \| \hat{\mathbf{b}}_{k-1} \|_2^2$, for $k = 2, 3, ..., m$

The quality of the reduced basis depends on the parameter δ : the bigger the value, the stronger the reduction of the basis. In the original paper of Lenstra, Lenstra and Lovász the LLL algorithm is defined only for $\delta = \frac{3}{4}$. With the help of the orthogonal projection

$$\pi_k : \mathbb{R}^n \rightarrow span(\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_{k-1})^{\perp}.$$

The second part of the definition could be written in the following way:

$$\delta \cdot \| \pi_{k-1}(\mathbf{b}_{k-1}) \|_2^2 \leq \| \pi_{k-1}(\mathbf{b}_k) \|_2^2, \text{ for } k = 2, 3, ..., m.$$

In the case where the basis consists of only two vectors, for $\delta = 1$ we get a Gauss reduced basis.

If $(\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_m)$ is LLL-reduced with some δ then $(\pi_k(\mathbf{b}_k)$, $\pi_k(\mathbf{b}_{k+1})$, ..., $\pi_k(\mathbf{b}_j))$ is also LLL-reduced with δ and $1 \leq k < j \leq n$.

Theorem 4.2. *Let* $(\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_m)$ *be LLL-reduced for the parameter* δ*. For* $\alpha = \frac{1}{\delta - \frac{1}{4}}$ *we have*

$$\| \hat{\mathbf{b}}_i \|_2^2 \leq \alpha^{j-i} \cdot \| \hat{\mathbf{b}}_j \|_2^2 \text{ for } 1 \leq i \leq j \leq m.$$

For the special case of $\delta = \frac{3}{4}$ and $\alpha = 2$ we have $\| \mathbf{b}_1 \|_2^2 \leq 2^{j-1} \cdot \| \hat{\mathbf{b}}_j \|_2^2$ so that the square of the length for big values of j is not arbitrary small.

Theorem 4.3. *Let* $(\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_m)$ *be a basis of the lattice* $L \subset \mathbb{R}^n$*. Then for* $i = 1, 2, ..., m$ *we have*

$$\lambda_j \geq \min_{i=j,j+1,...,m} \| \hat{\mathbf{b}}_i \|_2.$$

This theorem shows that the lengths $\| \hat{\mathbf{b}}_j \|_2$ are a rough approximation of the successive minima λ_j.

Theorem 4.4. *Let* $(\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_m)$ *be the basis of the lattice* $L \subseteq \mathbb{R}^n$ *with a parameter* δ*. It holds for* $\alpha = \frac{1}{\delta - \frac{1}{4}}$ *that :*

1. $\alpha^{1-j} \leq \frac{\|\hat{\mathbf{b}}_j\|_2^2}{\lambda_j^2}$, *for* $j = 1, 2, ..., m$,
2. $\frac{\|\mathbf{b}_j\|_2^2}{\lambda_j^2} \leq \alpha^{n-1}$, *for* $j = 1, 2, ..., m$,
3. $\| \mathbf{b}_k \|_2^2 \leq \alpha^{j-1} \cdot \| \hat{\mathbf{b}}_j \|_2^2$, *for* $k \leq j$.

Definition 4.5. Let $\| \cdot \|_p$ for $p \in [1, \infty]$ be any norm in $\mathbb{R}^n$ and $0 < \Delta \leq 1$. A basis $\mathbf{b}_1$, $\mathbf{b}_2$, ..., $\mathbf{b}_m$ of a lattice $L \subset \mathbb{R}^n$ is called Lovász-Scarf reduced with Δ or LS-reduced with Δ if

1. $F_j(\mathbf{b}_i) \leq F_j(\mathbf{b}_i \pm \mathbf{b}_j)$, for all $j < i$,
2. $\Delta F_i(\mathbf{b}_i) \leq F_i(\mathbf{b}_{i+1})$, for $1 \leq j < m$.

In the case of the Euclidean norm, the terms δ LLL-reduced lattice basis and $\Delta = \sqrt{\delta}$ LS-reduced lattice basis are equivalent. The bigger the value of Δ, the stronger the basis reduction.

Theorem 4.6. *For each LS-reduced basis* $(\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_m)$ *of a lattice* $L \subset \mathbb{R}^n$ *with* $\Delta \in (\frac{1}{2}, 1]$ *the following holds:*

$$(\Delta - \tfrac{1}{2})^{i-1} \leq \frac{F_i(\mathbf{b}_i)}{\lambda_{i,\|\cdot\|_p}}(L) \leq (\Delta - \tfrac{1}{2})^{i-m}, \text{ for } i = 1, ..., m.$$

4.2 LLL Algorithm

Algorithm 2.1 LLL Algorithm

Input : A basis $(\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_m) \in \mathbb{R}^n$ δ with $0 \leq \delta \leq 1$

1. $k := 2$ (k is the stage)

   [Compute the Gram-Schmidt coefficient $\mu_{i,j}$ for $1 \leq j < i < k$ and $c_i = \| \hat{\mathbf{b}}_i \|_2^2$ for $i = 1, ..., k-1$]

2. **WHILE** $k \leq n$

   - **IF** $k = 2$ **THEN** $c_1 := \| \mathbf{b}_1 \|_2^2$
   - **FOR** $j = 1, ..., k-1$

     $$\mu_{k,j} := \frac{(\langle \mathbf{b}_k, \mathbf{b}_k \rangle - \sum_{i=1}^{j-1} \mu_{j,i}\mu k,ic_i)}{c_j}$$

   - $c_k := \langle \mathbf{b}_k, \mathbf{b}_k \rangle - \sum_{j=1}^{k-1} \mu_{k,j} c_j$

3. Reduce the size of $\mathbf{b}_k$

   - **FOR** $j = k-1, ..., 1$
     - $\mu := \lceil \mu_{k,j} \rfloor$
     - **FOR** $i = 1, ..., j-1 : \mu_{k,j} := \mu_{k,i} - \mu\mu_{j,i}$
     - $\mu_{k,j} := \mu_{k,j} - \mu$
     - $b_k := \mathbf{b}_k - \mu \mathbf{b}_j$

4. - **IF** $\delta c_{k-1} > c_k + \mu_{k,k-1}^2 c_{k-1}$
   - **THEN** swap $\mathbf{b}_k$ and $\mathbf{b}_{k-1}$, $k := \max(k-1, 2)$
   - **ELSE** $k := k + 1$

5. **END** while

OUTPUT : an LLL-reduced basis $(\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_m) \in \mathbb{R}^n$ with δ

In step 1, when entering degree k, the basis $(\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_{k-1})$ is already LLL-reduced with a parameter δ and the Gram-Schmidt coefficients $\mu_{i,j}$, for $1 \leq j < i < k$, as well as the values $c_i = \| \hat{\mathbf{b}}_i \|_2^2$, for $i = 1, ..., k-1$, are already available.

Step 3 is actually an algorithm that reduces the size of the basis. For additional information on this algorithm see the lecture slides of Schnorr.

For the swap of $\mathbf{b}_k$ and $\mathbf{b}_{k-1}$ the values $\| \hat{\mathbf{b}}_k \|_2^2, \| \hat{\mathbf{b}}_{k-1} \|_2^2$ and $\mu_{i,j}, \mu_{j,i}$, for $j = k-1, k$ and $i = 1, 2, ..., m$, have to be calculated again. It is important to keep the basis in exact arithmetics because otherwise a change in the lattice would be made that cannot be fixed.

Let B denote $\max(\| \mathbf{b}_1 \|_2^2, \| \mathbf{b}_2 \|_2^2, ..., \| \mathbf{b}_m \|_2^2)$ for the input basis. Throughout the algorithm the bit length of the numerators and denominators of the rational numbers $\| \hat{\mathbf{b}}_i \|_2^2$, $\mu_{i,j}$ is bounded by $O(m \log B)$. The

bit length of the coefficients of the $b_i \in \mathbb{Z}^n$ is also bounded by $O(m \log B)$ throughout the algorithm.
The algorithm preforms at most $O(m^3 n \log B)$ arithmetic operations on integers that are $O(m \log B)$ bits long.

4.3 LS Algorithm

Lovász and Scarf [8] introduced in 1992 an algorithm that gives a reduced basis for an arbitrary norm in polynomial many steps with respect to the size of the input.

Algorithm 2.2 LS Algorithm

INPUT : A basis $(\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_m) \in \mathbb{R}^n$, Δ with $0 \leq \Delta \leq 1$

1. $k := 2$
2. **WHILE** $k \leq n$
   - Reduce the length of $\mathbf{b}_k$
     - **FOR** $j = k-1, ..., 1$
       - Compute a whole number μ_j which minimizes $F_j(\mathbf{b}_k - \mu \mathbf{b}_j)$
       - $b_k := \mathbf{b}_k - \mu \mathbf{b}_j$
     - **IF** $F_{k-1}(\mathbf{b}_k) < \Delta F_{k-1}(b_{k-1})$
     - **THEN** swap $\mathbf{b}_k$ and $\mathbf{b}_{k-1}$, $k := \max(k-1, 2)$
     - **ELSE** $k := k + 1$
3. **END** while

OUTPUT : an LS-reduced basis $\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_m$ with a parameter Δ

In each step of the algorithm we search for the first index $k-1$ for which one of the conditions

1. $F_{k-1}(\mathbf{b}_k + \mu \mathbf{b}_j) \geq F_{k-1}(\mathbf{b}_k)$ for integral μ,
2. $F_{k-1}(\mathbf{b}_k) \geq \Delta F_{k-1}(\mathbf{b}_{k-1})$

is not satisfied, where F is a distance function. If the first condition is not satisfied we replace $\mathbf{b}_k$ by $\mathbf{b}_k + \mu_j \mathbf{b}_j$, with μ being the integer that minimizes $F_{k-1}(\mathbf{b}_k + \mu \mathbf{b}_j)$. If after the replacement the second condition holds we move to level k. If the second condition is not satisfied, we interchange $\mathbf{b}_k$ and $\mathbf{b}_{k-1}$ and move to the corresponding level.

Notice that the maximum value of the components of the vector $F_1(\mathbf{b}_1)$, ..., $F_{k-1}(\mathbf{b}^{k-1})$, $F_k(\mathbf{b}_k)$, ..., $F_m(\mathbf{b}_m)$, consisting of distance functions, does not increase at any step of the basis reduction algorithm. If we replace $\mathbf{b}_k$ by $\mathbf{b}_k + \mu_j \mathbf{b}_j$, none of the terms actually change. If $\mathbf{b}_k$ and $\mathbf{b}_{k-1}$ are interchanged, $F_{k-1}(\mathbf{b}_{k-1})$ simply becomes $F_{k-1}(b_k) \leq \Delta F_{k-1}(b_k)$.

4.4 Computing the Distance Functions in Infinity Norm

The problem of computing the distance functions for the l_∞-norm is actually an optimization problem:

Given is a basis $\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_m$ and $\mathbf{x} \in \mathbb{R}^n$.

Compute $F_i(\mathbf{x}) = \min_{z_i \in \mathbb{R}} \parallel x + \sum_{j=1}^{i-1} z_j \mathbf{b}_j \parallel_\infty$ such that

$$z_i \geq 0 \tag{1}$$

$$z_i := \parallel x + \sum_{j=1}^{i-1} z_j \mathbf{b}_j \parallel_\infty \tag{2}$$

$$-z_i \leq x_k + \sum_{j=1}^{i-1} z_j \mathbf{b}_{j,k} \leq z_i, for\ all\ k = 1, ..., m \tag{3}$$

We get the following optimization problem:

$$\text{Minimize } Q(x) = z_i$$

s.t.

$$z_i \geq 0 \tag{4}$$

$$z_i \leq \parallel x \parallel_\infty \tag{5}$$

$$\sum_{j=1}^{i-1} z_j \mathbf{b}_{j,k} - z_i \leq -x_k \tag{6}$$

$$-\sum_{j=1}^{i-1} z_j \mathbf{b}_{j,k} - z_i \leq x_k, for\ all\ k = 1, ..., m \tag{7}$$

Such an optimization problem is solved in polynomial time for all rational input values with the help of the Ellipsoid Method or the Karmarkars Algorithm. In practice the Simplex Algorithm is even more efficient.

The best and quickest way to solve such a problem would be to use the ILOG CPLEX Software, which is a large-scale programming software that is capable of solving huge, real-world optimization problem. The problem is simply written in the respective CPLEX language and the software finds a solution if there is such.

Another option is to model the problem in a language that translates the mathematical model into a linear or mixed-integer mathematical program, like for example Zimpl [5], and solve the resulting problem using JAVA. This approach is not as quick and affective as the first one but is the only way to solve such an optimization problem without expensive software.

4.5 Time Bounds

The LS basis algorithm is known to converge in polynomial time, including the number of variables n, for $F(\mathbf{x}) = |x|$ and a general lattice given by an integer basis. This argument is based on two observations:

1. An interchange between $\mathbf{b}_{k-1}$ and $\mathbf{b}_k$ preserves the value of the distance function for all indices other than k and $k - 1$.

2. For $F(\mathbf{x}) = |x|$, the product $F_{k-1}(\mathbf{b}_{k-1})F_k(\mathbf{b}_k)$ is constant when the vectors $\mathbf{b}_{k-1}$ and $\mathbf{b}_k$ are exchanged. This permits us to deduce that $\prod(F_{k-1}(\mathbf{b}_{k-1}))^{m-1}$ decreases by a factor Δ at each interchange. $\prod(F_{k-1}(\mathbf{b}_{k-1}))^{m-1} \geq 1$ is therefore true for any basis, from which the polynomial convergence follows.

The product $F_{k-1}(\mathbf{b}_{k-1})F_k(\mathbf{b}_k)$ is not constant in the general case, and therefore the algorithm does not execute in polynomial time with respect to the number of variables m. However it could be shown that the algorithm works in polynomial time for a fixed m. There are two arguments to support this conclusion both of which are going to be explained here briefly. They are both fully covered in the paper of Lovász and Scarf [7].

The idea behind both of them is to obtain a lower bound for the values of $F_i(\mathbf{b}_i)$. For this purpose, let $C \subset B(R)$ be a ball of radius R. The distance function $F(x) \geq \frac{|x|}{R}$. Let $\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_m$ be any basis that satisfies $F_i(\mathbf{b}_i) \leq U$ where U is equal to $\max F_i(\mathbf{a}_i)$. We have that $\mathbf{c}_i = \mathbf{b}_i + \sum_{j=1}^{i-1} \mu_{i,j}\mathbf{b}_j$ is a proper basis associated with $\mathbf{b}_i$, satisfying $F_i(\mathbf{c}_i) = F_i(\mathbf{b}_i)$ and $F_1(\mathbf{c}_i) \leq F_i(\mathbf{b}_i) + \frac{1}{2}\sum_1^{i-1} F_i(\mathbf{b}_j) \leq mU$. Therefore we get that $|\mathbf{c}_i| \leq mUR$.

The distance function is limited from below by the following inequality:

$$F_i(\mathbf{b}_i) \geq \min \frac{|(\mathbf{b}_i + \alpha_1\mathbf{c}_1 + ... + \alpha_{i-1}\mathbf{c}_{i-1})|}{R}.$$

This actually represents the distance between the vector $\mathbf{b}_i$ and the space $< \mathbf{c}_1, ..., \mathbf{c}_{i-1} >$. When this distance is represented by the Gram matrix $G(\mathbf{x}_1, ..., \mathbf{x}_i) = \det[(\mathbf{x}_j, \mathbf{x}_k)]_{j,k=1}^{i}$, for the integral $\mathbf{b}_j$ and $\mathbf{c}_1, ..., \mathbf{c}_{i-1}$ we get that $G(\mathbf{c}_1, ..., \mathbf{c}_{i-1}, \mathbf{b}_j) \geq 1$. From there it can be concluded that $F_i(\mathbf{b}_i) \geq 1/[R(mRU)^{i-1}] \geq 1/[R(mRU)^{n-1}]$. Set here $F_i(\mathbf{b}_i)$ to be equal to V.

As we already mentioned, each component of $F_1(\mathbf{b}_1)$, ..., $F_i(\mathbf{b}_i)$, $F_{i+1}(\mathbf{b}_{i+1})$, ..., $F_m(\mathbf{b}_m)$ is bounded above by $\max F_i(\mathbf{a}_i) = U$ throughout the algorithm. Scarf and Lovász also noticed that the first term in the sequence to change at any iteration decreases by a factor of $(1 - \epsilon)$.

The first argument for the polynomial convergence of the generalized algorithm is therefore to observe that the maximum number of interchanges is

$$[\log(U/V)/\log(1/(1 - \epsilon))]^m$$

Using the introduced lower bound V, it can be seen that the interchanges of the basis reduction algorithm is bounded above by :

$$[m \log(mUR)/\log(1/(1\ -\epsilon))]^m$$

The second argument for polynomiality, which achieves a different bound, depends on the observation that for a general distance function the product $F_i(\mathbf{b}_i)\ F_{i+1}(\mathbf{b}_{i+1})$ increases by a factor $\leq\ 2$ in any step of the algorithm for an interchange of b_i and b_{i+1}. Here $F_i(\mathbf{b}_i) = 1$. Let $\mathbf{y} = \mathbf{b}_i$ and $\mathbf{x} = \frac{\mathbf{b}_{i+1}}{F_i(\mathbf{b}_{i+1})}$. $F_{i+1}(\mathbf{x}) = \frac{1}{d_{\mathbf{x}}}$ and $F^*_{i+1}(\mathbf{b}_i) = \frac{1}{d_{\mathbf{x}}}$ with F^*_{i+1} the distance function associated with a projection of the original basis into $< \mathbf{b}_i, \mathbf{b}_{i+2}, ..., \mathbf{b}_m >$. Therefore

$$\begin{aligned} &F_i(\mathbf{b}_{i+1})F^*_{i+1}(\mathbf{b}_i)/F_i(\mathbf{b}_i)F_{i+1}(\mathbf{b}_{i+1}) \ = \\ &\quad = \ [\tfrac{F_i(\mathbf{b}_{i+1})}{d_{\mathbf{y}}}]/[\tfrac{F_i(\mathbf{b}_{i+1})}{d_{\mathbf{x}}}] \ = \ d_{\mathbf{x}}/d_{\mathbf{y}} \ \leq \ 2 \end{aligned}$$

Let $D(\mathbf{b}_1, ..., \mathbf{b}_m) \ = \ \prod (F_i(\mathbf{b}_i))^{\gamma^{m-1}}$, with $\gamma \ = \ 2 \ + \ \frac{1}{\log(\frac{1}{(1\ -\ \epsilon)})}$. It is easy to see that $D(\mathbf{b}_1, ..., \mathbf{b}_m)$ decreases by a factor of at least $1\ -\ \epsilon$ at each interchange required by the basis reduction algorithm. Since $V \ \leq \ F_i(\mathbf{b}_i) \ \leq \ U$ at each step of the algorithm , the number of interchanges is bounded above by:

$$\begin{aligned} &[(\gamma^m \ - \ 1)/(\gamma \ - \ 1)] \log(U/V) \log(1/(1 \ - \ \epsilon)) \\ &\quad \leq \ (\gamma^n \ - \ 1)/(\gamma \ - \ 1)] m \log(mUV) \log(1/(1 \ - \ \epsilon)) \end{aligned}$$

.

This second estimate is much better than the first one in terms of its dependence to U and R. And since the number of possible values of the vector $F_1(\mathbf{b}_1), ..., F_i(\mathbf{b}_i), F_{i+1}(\mathbf{b}_{i+1}), ..., F_m(\mathbf{b}_m)$ is finite, the basis reduction algorithm executes in finite time even when $\epsilon \ = \ 0$.

5 Block-Korkine-Zolotarev Algorithm

The algorithms presented in the previous section - LLL and LS, produce a reduced basis in polynomial time with respect to the input data but the result is only an approximation of the actual shortest lattice vector with an exception of an exponential factor. In order to improve the practical usage of these algorithms, Schnorr developed a hierarchy of reduction terms with respect to the Euclidean norm, the so called block reduction. The new algorithm, that Schnorr created, covers the general case of lattice basis using the LLL-reduced and the HKZ-reduced (Hermite-Korkine-Zolotarev, for more information see the script of Schnorr) bases as extreme cases.

In this chapter we present the practical BKZ algorithm and the subroutine ENUM which work with respect to the Euclidean norm. This block reduction theory is then expanded to an arbitrary norm and an algorithm dealing specifically with the infinity norm is finally explained.

5.1 BKZ-reduced lattice basis

Let $L = L(\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_m) \in \mathbb{R}^n$ be a lattice with an ordered basis $\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_m$. Furthermore, $\pi_i : \mathbb{R}^m \Rightarrow span(\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_m)$ holds true. Let L_i denote the lattice $\pi_i(L)$, which is a lattice of rank $m - i + 1$.

Definition 5.1. An ordered basis $\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_m \in \mathbb{R}^n$ is called size-reduced if $|\mu_{i,j}| \leq \frac{1}{2}$ for $1 \leq j < i \leq m$ An individual basis vector $\mathbf{b}_i$ is size-reduced if $|\mu_{i,j}| \leq \frac{1}{2}$ for $1 \leq j < i$.

Definition 5.2. An ordered basis $\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_m$ of a lattice $L \subset \mathbb{R}^n$ is called a Korkine-Zolotarev basis if:

1. it is size-reduced,
2. $\| \hat{\mathbf{b}}_i \|_2 = \lambda_1(L_i)$, for $i = 1, ..., m$.

This definition is equivalent to the original definition given in Korkine and Zolotarev's book from 1873.

Theorem 5.3. *Let β be an integer such that $2 \leq \beta < n$. A lattice basis $\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_m$ is β-reduced if:*

1. *it is size-reduced,*
2. $\| \hat{\mathbf{b}}_i \|_2 \leq \lambda_i(L_i(\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_{\min(i+\beta-1,m)}))$, *for* $i = 1, ..., m-1$.

Let α_β denote the maximum of $\frac{\|\mathbf{b}_1\|_2}{\|\mathbf{b}_\beta\|_2}$ over all Korkine-Zolotarev reduced bases. For $\ln \beta$ - the natural logarithm of β, $\alpha_2 = \frac{3}{4}, \alpha_3 = \frac{3}{2}$ and $\alpha_\beta \leq \beta^{1+\ln \beta}$. The constraint $\alpha_\beta^{\frac{1}{(\beta-1)}}$ converges to 1 with the increase of β.

The following theorem shows the strength of a β-reduced basis in comparison to an LLL-reduced one.

Theorem 5.4. *Every β-reduced basis $\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_m$ of a lattice $L \subset \mathbb{R}^n$ satisfies $\| \mathbf{b}_1 \|_2^2 \leq \alpha_\beta^{\frac{(n-1)}{\beta-1}} \lambda_1(L)^2$ provided that $\beta - 1$ divides $n - 1$.*

Definition 5.5. A basis $\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_m$ is called β-reduced with a parameter δ for $\frac{1}{4} < \delta \leq 1$ if:

1. it is size-reduced,
2. $\delta \| \hat{\mathbf{b}}_i \|_2^2 \leq \lambda_1(L_i(\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_{\max(i+\beta-1,m)}))^2$, for $i = 1, ..., m$.

Notice that for $\beta = 2$ and $\frac{1}{3} \leq \delta \leq 1$, a (β, δ)-reduced basis is actually an LLL-reduced basis.

Definition 5.6. A lattice basis $\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_m$ is called β-block reduced with a parameter δ, for $\frac{1}{2} \leq \delta \leq 1$ and $i = 1, ..., m$, if for the distance function F the following two inequations hold true:

1. $\delta F_i(\mathbf{b}_i) \leq \min F_i(\mathbf{b})$, where $\mathbf{b} \in \mathbb{Z}\mathbf{b}_i + ... + \mathbb{Z}\mathbf{b}_{\min(i+\beta-1,m)} - 0$,
2. $F_j(\mathbf{b}_i) \leq F_j(\mathbf{b}_i \pm \mathbf{b}_j)$, for all $j < i$.

5.2 BKZ Algorithm with Euclidean Norm

The following algorithm performs a β-reduction with respect to a parameter δ. Its backbone is the subroutine $ENUM(j, k)$, which is also presented in this section.

Algorithm 3.1 BKZ Algorithm with Euclidean Norm

INPUT: A lattice basis $\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_m \in \mathbb{Z}^n$ with parameters δ, with $0 \leq \delta \leq 1$, and β, with $2 < \beta < m$

1. Size-reduce $\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_\beta$; $j := m - 1$; $z := 0$
2. **WHILE** $z < m - 1$
   - $j := j + 1$, **IF** $j = m$ **THEN** $j := 1$
   - $k := \min(j + \beta - 1, m)$
   - $ENUM(j, k)$ (ENUM finds the whole numbers $(u_j, ..., u_k)$ from the representation of the vectors $\mathbf{b}_j^{new} = \sum_{i=j}^{k} u_i \mathbf{b}_i$ with $\bar{c}_j := \| \pi_j(\mathbf{b}_j^{new}) \|_2^2 = \lambda_{1,\|\cdot\|_2}^2(L(\pi_j(\mathbf{b}_j), ..., \pi_j(\mathbf{b}_k))))$
   - $h := \min(k + 1, m)$
   - **IF** $\bar{c}_j < \delta c_j$
   - **THEN**
     - extend $\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_{j-1}, \mathbf{b}_j^{new}$ to a basis $\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_{j-1}, \mathbf{b}_j^{new}, ..., \mathbf{b}_k^{new}, \mathbf{b}_{k+1}, ..., \mathbf{b}_m$ of a lattice L
     - size-reduce $\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_{j-1}, \mathbf{b}_j^{new}, ..., \mathbf{b}_h^{new}$

- $z := 0$

- **ELSE**
  - size-reduce $\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_m$
  - $z := z + 1$

3. **END WHILE**

OUTPUT: BKZ-reduced basis $\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_m$

Throughout the algorithm the integer j is cyclically shifted through the integers $1, 2, ..., m-1$. The variable z counts the number of positions j which satisfy the inequality

$$\delta \parallel \hat{\mathbf{b}}_j \parallel_2^2 \leq \lambda_{1,\|\cdot\|_2}^2(L(\pi_j(\mathbf{b}_j), ..., \pi_j(\mathbf{b}_k))).$$

If this inequality does not hold for some j, then b_j^{new} would be inserted into the basis, a size-reduction would be done and z would be reset to 0. The term $j = m$ is skipped since the inequality always holds for it. As defined earlier, a basis $\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_m$ is β-reduced with a parameter δ if it is size-reduced and $z = m - 1$. Therefore, the algorithm always produces a β-block reduced basis with a parameter δ up to a certain error.

The size reduction in step one could be alternatively replaced either by an LLL-reduction or an L^3FP-reduction (LLL Floating Point-reduction).

When adding the vectors $\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_{j-1}, \mathbf{b}_j^{new} = \sum_{i=j}^{k} u_i \mathbf{b}_i$ we differentiate between several cases for $g := \max i : j \leq i \leq k, u_i \neq 0$:

1. $|u_g| = 1$: $\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_j^{new}, \mathbf{b}_j, ..., \mathbf{b}_{g-1}, \mathbf{b}_{g+1}, ..., \mathbf{b}_m$ is a basis of the lattice. This means that the vector $\mathbf{b}_g$ could be removed from the lattice and replaced by $\mathbf{b}_j^{new}$.

2. $|u_g| > 1$: In this case the basis $\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_m$ is transformed into $\mathbf{b}_1$, $\mathbf{b}_2$, ..., $\mathbf{b}_{j-1}$, $\mathbf{b}_j^{'}$, ..., $\mathbf{b}_g^{'}$, $\mathbf{b}_{g+1}, ..., \mathbf{b}_m$ such that $\mathbf{b}_j^{new} = \sum_{i=j}^{g} u_i^{'} \mathbf{b}_i^{'}$ with $|u_i^{'}| = 1$. This is always possible when the result of the subroutine $ENUM(j,k)$ holds, thus when $ggT(u_j, ..., u_k) = 1$ holds. Otherwise $\mathbf{b} := \frac{\mathbf{b}_j^{new}}{ggT(u_j, ..., u_k)}$ is a lattice vector for which $\parallel \pi_j(\mathbf{b}) \parallel_2^2 = \frac{\hat{c}_j}{ggT(u_j, ..., u_k)^2}$ and therefore $\hat{c}_j > \lambda_{1,\|\cdot\|_2}^2 (L(\pi_j(\mathbf{b}_j), ..., \pi_j(\mathbf{b}_k)))$. The vector $\mathbf{b}_j^{'}$ is then removed from the basis and replaced by $\mathbf{b}_j^{new}$.

As mentioned before, the core of this algorithm is actually the routine $ENUM(j,k)$, which computes the vectors for which the distance function F_j is minimal. We are looking here for integers $(u_j, ..., u_k) \neq (0, ..., 0)$ and $\bar{F}_j := F_j(\sum_{i=j}^{k} u_i \mathbf{b}_i) = \lambda_{1,F_j}(L(\mathbf{b}_j, ..., \mathbf{b}_k))$. This is done with the help of a recursive enumeration of all coefficient vectors $(\tilde{u}_t, ..., \tilde{u}_k) \in \mathbb{Z}^{k-t+1}$, for $t = k, ..., j$ and $F_t(\sum_{i=t}^{k} \tilde{u}_i \mathbf{b}_i) < F_j(\mathbf{b}_j)$. For the coefficient vector different from 0, for which F_j is minimal throughout the whole enumeration, the integers $(u_j, ..., u_k)$ are set to $(\tilde{u}_j, ..., \tilde{u}_k)$.

For each step $t > p$ of the algorithm, the integers $(\tilde{u}_{t+1}, ..., \tilde{u}_s)$ are fixed. All left integers u_{t-1}, for which $F_t(\sum_{i=t}^{k} \tilde{u}_i \mathbf{b}_i) < F_j(\mathbf{b}_j)$, are enumerated.

Since $F_t(-\mathbf{x}) = F_t(\mathbf{x})$ for all $x \in \mathbb{R}^n$ if $\tilde{u}_k = ... = \tilde{u}_t = 0$ we could limit the search to only nonnegative integers. Every time a coefficient is computed it is organized in a search tree with depth $k - j + 2$.

Currently the Depth-First-Search is considered most appropriate for the enumeration. With respect to the Euclidean norm and $z \in \mathbb{R}$, the distance function F_t can be efficiently computed with the help of the Gram-Schmidt orthogonal system. From

$$\tilde{c}_t := \| \pi_t(\sum_{i=t}^{k} \tilde{u}_i \mathbf{b}_i) \|_2^2 \text{ and } c_t := \| \hat{\mathbf{b}}_t \|_2^2 = \| \pi_t(\mathbf{b}_t) \|_2^2, \text{ for } 1 \leq t \leq k$$

we get

$$F_t^2(\sum_{i=t}^{k} \tilde{u}_i \mathbf{b}_i) = \tilde{c}_t = \tilde{c}_{t+1} + (\tilde{u}_t + \sum_{i=t+1}^{k} \tilde{u}_i \mu_{i,t})^2 c_t$$

$-y_t$ is the minimal real point for $y_t := \sum_{i=t+1}^{k} \tilde{u}_i \mu_{i,t}$. We get the whole minimum through the symmetry

$$c_t(-y_t + x, \tilde{u}_{t+1}, ..., \tilde{u}_k) = c_t(-y_t - x, \tilde{u}_{t+1}, ..., \tilde{u}_k)$$

at the point $\nu_t := \lceil -y_t \rfloor$. Depending on whether $\nu_t > -y_t$ holds true or not, we get a sequence of non-declining values $\tilde{c}_t$ either in order $(\nu_t, \nu_t - 1, \nu_t + 1, \nu_t - 2, ...)$ or in order $(\nu_t, \nu_t + 1, \nu_t - 1, \nu_t + 2, ...)$.

Algorithm 3.2 ENUM with Euclidean Norm

INPUT: $j, k, \mathbf{b}_1, ..., \mathbf{b}_m$

1.
   - $s := t := j$, $\bar{c}_j := c_j$, $\tilde{u}_j := u_j := 1$
   - $\nu_j := y_j := \Delta_j := 0$, $\delta_j := 1$
   - **FOR** $i = j + 1, ..., k + 1$
     - $\tilde{c}_i := u_i := \tilde{u}_i := \nu_i := \Delta_i := 0$
     - $\delta_i := 1$

2. **WHILE** $t \leq k$

   - $\tilde{c}_t := \tilde{c}_{t+1} + (y_t + \tilde{u}_t)^2 c_t$
   - **IF** $\tilde{c}_t < \bar{c}_j$
   - **THEN**
     - **IF** $t > j$
     - **THEN**
       - $t := t - 1$, $y_t := \sum_{i=t+1}^{s} \tilde{u}_i \mu_{i,t}$
       - $\tilde{u}_t := \nu_t := \lceil -y_t \rfloor$, $\Delta_t := 0$
       - **IF** $\tilde{u}_t > -y_t$
       - **THEN** $\delta_t := -1$
       - **ELSE** $\delta_t := 1$

- **ELSE** $\bar{c}_j := \tilde{c}_j$, $u_i := \tilde{c}_i$, for $i = j, ..., k$

• **ELSE**

- $t := t + 1$
- $s := \max(s, t)$
- **IF** $t < s$ **THEN** $\Delta_t := -\Delta_t$
- **IF** $\Delta_t \delta_t \geq 0$ **THEN** $\Delta := \Delta + \delta_t$
- $\tilde{u}_t := \nu_t + \Delta_t$

3. **END WHILE**

OUTPUT: The minimal point $(u_j, ..., u_k) \in \mathbb{Z}^{k-j+1} - 0^{k-j+1}$ and the minimum $\bar{c}_j(u_j, ..., u_k)$

With a proper prereduction of the lattice basis, this algorithm gives a shortest lattice vector consisting only of integers. This routine finds the shortest vector with respect to the Euclidean norm up to a dimension 50 by setting $j = 1$ and $k := m$. When it is used as a subroutine of a (β, δ)-block reduction with $\delta < 1$ only the vectors $\mathbf{b}_j^{new}$ with $\bar{c} < \delta c_j$ are of interest. The costs could be therefore reduced from the beginning by setting $\bar{c}_j := \delta c_j$.

5.3 BKZ Algorithm with an Arbitrary Norm

The BKZ Algorithm with an arbitrary norm is analogous to the one with the Euclidean norm. Instead of $\bar{c}_j$ here the distance functions are used.

Algorithm 3.3 BKZ Algorithm with an Arbitrary Norm

INPUT: A lattice basis $\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_m \in \mathbb{Z}^n$ with parameters δ with $0 \leq \delta \leq 1$ and β with $2 \leq \beta \leq m$

1. Size-reduce $\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_\beta$; $j := m - 1$; $z := 0$
2. **WHILE** $z < m - 1$

- $j := j + 1$, **IF** $j = m$ **THEN** $j := 1$
- $k := \min(j + \beta - 1, m)$
- $ENUM(j, k)$ (ENUM finds the whole numbers $(u_j, ..., u_k)$ from the representation of the vectors $\mathbf{b}_j^{new} = \sum_{i=j}^{k} u_i \mathbf{b}_i$ with $\bar{F}_j := F_j(\sum_{i=j}^{k} u_i \mathbf{b}_i))$
- $h := \min(k + 1, m)$
- **IF** $\bar{F}_j < \delta F_j(\mathbf{b}_j)$
- **THEN**
  - extend $\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_{j-1}, \mathbf{b}_j^{new}$ to a basis $\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_{j-1}, \mathbf{b}_j^{new}, ..., \mathbf{b}_k^{new}, \mathbf{b}_{k+1}, ..., \mathbf{b}_m$ of a lattice L
  - size-reduce $\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_{j-1}, \mathbf{b}_j^{new}, ..., \mathbf{b}_h^{new}$

- $z := 0$

- **ELSE**
  - size-reduce $\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_m$
  - $z := z + 1$

3. **END WHILE**

OUTPUT: BKZ-reduced basis $\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_m$

The same logic is applied here as in the algorithm that works with respect to the Euclidean norm. j is cyclically shifted through the integers $1, 2, ..., m-1$. The variable z counts the number of positions j which satisfy the inequality $\delta F_j(\mathbf{b}_j) < \bar{F}_j$. Here again, if the inequality does not hold, $\mathbf{b}_j^{new}$ is inserted into the basis, a size-reduction is done and z is set to 0. The term $j = m$ is skipped since for it the inequality always holds. The basis $(\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_{j-1}, \mathbf{b}_j^{new})$ is extended to $(\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_{j-1}, \mathbf{b}_j^{new}, ..., \mathbf{b}_h^{new})$ using the coefficients u_j in the representation $\mathbf{b}_j^{new} = \sum_{i-j}^{h} u_i \mathbf{b}_i$. The matrix $T \in GL_{h-j+1}(\mathbb{Z})$ with $[u_j, ..., u_h] \cdot T = [1, 0, ..., 0]$ is computed at this point and the vectors $[\mathbf{b}_j^{new}, ..., \mathbf{b}_h^{new}]$ are set to $[\mathbf{b}_j, ..., \mathbf{b}_h] \cdot T^{-1}$.

The backbone of the algorithm is again the routine $ENUM(j, k)$, which computes the minimal point $(u_j, ..., u_k)$ so that the minimum of $\bar{F}_j$ can be found. Here we are looking for a vector $\mathbf{b} \in L(\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_m) \subset \mathbb{R}^n$ with $\| \mathbf{b} \|_p = \lambda_{1, \|\cdot\|_p}(L)$. Let $\bar{\mathbf{b}} = \sum_{i-1}^{m} u_i \mathbf{b}_i$ be the vector that has the minimum l_p-norm from all enumerated lattice vectors. At the beginning of the algorithm the vector $\bar{\mathbf{b}}$ is set to $\mathbf{b}_1$ which means that $(u_1, ..., u_m) = (1, 0, ..., 0)$. We can stop the search for the shortest lattice vector in a partial tree with a root $(\tilde{u}_t, ..., \tilde{u}_m)$ as soon as $F_t(\omega_t) \geq \| \bar{\mathbf{b}} \|_p$.

For this stop criterion the minimal point $(\lambda_t, ..., \lambda_m)$ of the function $f(\mu_t, ..., \mu_m) := \| \bar{\mathbf{b}} \|_p \| \sum_{i=1}^{m} \mu_i \omega_i \|_q - \tilde{c}_t$, with $\mu_i \in \mathbb{R}$ and $\sum_{i=1}^{m} \mu_i \omega_i = \tilde{c}_t$, must be first computed. The enumeration could be stopped if $f(\lambda_t, ..., \lambda_m)$ is negative. The minimum of this function is calculated in polynomial time with the help of the Ellipsoid method. The cost of the stop criterion is comparable to the cost of the computation of the distance functions. Our purpose is thus to find an optimal range of vectors $(\lambda_t, ..., \lambda_m)$. For $(\lambda_t, ..., \lambda_m) = (1, 0, ..., 0)$ we get

$$\frac{\bar{c}_t}{\|\omega_t\|_q} \geq \| \bar{\mathbf{b}} \|_p \Longrightarrow F_t(\omega_t) \geq \| \bar{\mathbf{b}} \|_p$$

This gives us a stop criterion which can be tested in linearly many arithmetical operations.

For $(\lambda_t, ..., \lambda_m) = (1, 0, ..., 0)$ we have

$$|\tilde{u}_t + y_t| \geq \frac{\|\bar{\mathbf{b}}\|_p \|\hat{\mathbf{b}}_t\|_q}{c_t} \implies F_t(\omega_t) \geq \| \bar{\mathbf{b}} \|_p$$

In this way we can limit the number of possible values of $\tilde{u}_t$ with constant $(\tilde{u}_{t+1}, ..., \tilde{u}_m)$ a priori.

This enumeration in the direction $\lambda_t(\omega_t - \omega_{t+1})$ can be stopped when $\| \bar{\mathbf{b}} = \|_p \sum_{i=t}^{m} \lambda_i \tilde{c}_i$ for any $(\lambda_t, ..., \lambda_m)$ with $\lambda_t \neq 0$.

For simplification here we limit the calculation only to the case $\sum_{i=t}^{s} \lambda_i \tilde{c}_i$ with $\lambda_t > 0$. The enumeration is thus stopped only in the direction $\omega_t - \omega_{t+1}$. Without this limitation with every stop different cases must be considered.

Algorithm 3.4 ENUM with an Arbitrary Norm

INPUT: $m, c_i, \mathbf{b}_i, \hat{\mathbf{b}}_i, \mu_{i,t}$ for $i = 1, ..., m$ and $1 \leq t < i \leq m$

1.
   - $s := t := 1, \tilde{u}_1 := \eta_1 := \delta_1 := 1, \nu_1 := y_1 := \Delta_1 := 0$
   - $\omega_1 := (0, ..., 0), c := \| \mathbf{b}_1 \|_p$
   - **FOR** $i = 2, ..., m + 1$
     - $\tilde{c}_i := u_i := \tilde{u}_i := \nu_i := y_i := \Delta_i := 0, \nu_i := \delta_i := 1$
     - $\omega_i := (0, ..., 0)$

2. **WHILE** $t \leq m$

   - $\tilde{c}_t := \tilde{c}_{t+1} + (y_t + \tilde{u}_t)^2 c_t$
   - **IF** $\tilde{c}_t < (R(\| \cdot \|_p)c)^2$
   - **THEN** $\omega_t := \omega_{t+1} + (y_t + \tilde{u}_t)\hat{\mathbf{b}}_t$
     - **IF** $t > 1$
     - **THEN**
       - **IF SCHNITT**$_p$ $(s, \omega_t, ..., \omega_s, \tilde{c}_t, ..., \tilde{c}_s, c) = 1$
       - **THEN**
         - **IF** $\eta_t = 1$ **THEN GO TO** 3
         - $\eta_t := 1, \Delta_t := -\Delta_t$
         - **IF** $\Delta_t \delta_t \geq 0$ **THEN** $\Delta_t := \Delta_t + \delta_t$
         - $\tilde{u}_t := \nu_t + \Delta_t$
       - **ELSE**
         - $t := t - 1, \eta_t := \Delta_t := 0$
         - $y_t := \sum_{i=t+1}^{s} \tilde{u}_i \mu_{i,t}$
         - $\tilde{u}_t := \nu_t := \lceil -y_t \rfloor$
         - **IF** $\tilde{u}_t > -y_i$
         - **THEN** $\delta_t := -1$
         - **ELSE** $\delta_t := 1$
     - **ELSE**
       - **IF** $\| \omega_1 \|_p < c$
       - **THEN** $(u_1, ..., u_s) := (\tilde{u}_1, ..., \tilde{u}_s), c := \| \omega_1 \|_p$
   - $t := t + 1$
   - $s := \max(t, s)$

– **IF** $\nu_t = 0$
– **THEN**
  * $\Delta_t := -\Delta_t$
  * **IF** $\Delta_t \delta_t \geq 0$ **THEN** $\Delta_t := \Delta_t + \delta_t$
– **ELSE** $\Delta_t := \Delta_t + \delta_t$
– $\tilde{u}_t := \nu_t + \Delta_t$

3. **END WHILE**

OUTPUT: $(u_1, ..., u_m)$, $c = \| \sum_{i=1}^{m} u_i \mathbf{b}_i \|_p$

The algorithm SCHNITT$_p$ works in the following way:

Algorithm 3.5 SCHNITT$_p$
INPUT: $s, \omega_t, ..., \omega_s, \tilde{c}_t, ..., \tilde{c}_s, c$

1. Compute the minimum F of the function $f(\lambda_t, ..., \lambda_s) := c \| \sum_{i=t}^{s} \lambda_i \omega_i \|_q - \sum_{i=t}^{s} \lambda_i \tilde{c}_i$ with respect to a subgroup of the vectors $(\lambda_t, ..., \lambda_s)$ with $\lambda_t > 0$ and $\sum_{i=t}^{s} \lambda_i \tilde{c}_i > 0)$. The most efficient constraint in practice is $(\lambda_t, ..., \lambda_s) = (1, 0, ..., 0)$

OUTPUT: 1, if $F < 0$ and 0 otherwise

η_t shows in how many directions the enumeration process of step t was broken. If $\eta_t = 1$ and SCHNITT$_p(....) = 1$ holds, then both directions can be broken, which means that t can be increased by 1. By always choosing a positive $\tilde{u}_s$ the redundancy of the enumeration process could be avoided. Thus at step s only one of the directions needs to be processed and η_s can be initialized directly with 1. Here c is the l_p-norm of the shortest lattice vector.

The number of the knots, which are traversed throughout the $ENUM(j, k)$ algorithm , is proportional to the volume of the set, in which the orthogonal vectors ω_t lie. Without the algorithm SCHNITT$_p$ the set is simply the l_2-norm ball $S_n(0, R(\| \cdot \|_p) \| \bar{\mathbf{b}} \|_p)$. Through the constraint $(\lambda_t, ..., \lambda_m) = (1, 0, ..., 0)$ the restraint $\| \mathbf{x} \|_2^2 \leq \| \bar{\mathbf{b}} \|_p \| \mathbf{x} \|_q$, for $\mathbf{x} \in \mathbb{R}^n$, is reached. For l_∞-norm this restraint means a decrease of the volume with an exponential factor. The cost of this step is linear.

5.4 Enumeration for Infinity Norm

We consider here the case of lattices consisting only of integers. The l_∞-norm of the shortest lattice vector is therefore also an integer and since $R(\| \cdot \|_p) = \sqrt{n} \tilde{c}_t < (R(\| \cdot \|_\infty c))^2$, it can be replaced by $\tilde{c}_t \leq n(c-1)^2$.

In practice often only l_∞-norm equal to 1 is wanted. In such cases c could be initialized with 2 and the enumeration is completed as soon as such a norm is found. Thus a deterministic algorithm is found with which all Knapsack problems with dimension up to 66 can be efficiently solved.

We consider the efficiency of the simplest case with $(\lambda_t, ..., \lambda_m) = (1, 0, ..., 0)$ with comparison to the complete enumeration with respect to the Euclidean norm. The following holds:

1. $F_t(\omega_t) \leq d \implies$ each $\nu \in \pm d^n$ with $\| \omega_t - \frac{1}{2}\nu \|_2^2 \leq \frac{nd^2}{4}$
2. $\min_{\mu \in \pm d^n} \| \omega_t - \frac{1}{2}\nu \|_2^2 \leq \frac{nd^2}{4} \iff \| \omega_t \|_2^2 \leq d \| \omega_t \|_1$.

This means that instead of searching for the shortest lattice vector in a ball with a center 0 and a radius $d\sqrt{n}$ we are searching in the conjunction of 2^n balls with centers $(\pm\frac{d}{2}, ..., \pm\frac{d}{2})$ and a radius $\frac{d}{2}\sqrt{n}$. The volume V_n of the conjunction of the smallest balls equals to

$$(2d)^n \int_0^{\frac{1}{2}+\frac{1}{2}\sqrt{n}} .... \int_0^{\frac{1}{2}+\frac{1}{2}\sqrt{n}} X(x_1, ..., x_n) dx_n ....dx_1$$

In the equation above $X(\mathbf{x}_1, ..., \mathbf{x}_n) = \begin{cases} 1, & \text{if} \sum_{i=1}^n x_i^2 \leq \sum_{i=1}^n x_i \\ 0, & \text{otherwise} \end{cases}$

Through transformation we get

$$V_n = d^n \int_1^{\sqrt{n}} \int_{\max(-1, -\sqrt{n-x_1^2})}^{\sqrt{n-x_1^2}} .. \\ .. \int_{\max(-1, -\sqrt{n-x_1^2-....-x_{n-1}^2})}^{\sqrt{n-x_1^2-....-x_{n-1}^2}} 1 dx_n ....dx_1$$

Thus $V_2 = (2 + \pi)d^2$, $V_3 = (2 + 4\pi)d^3$ and the volume of the big ball is $\frac{2+\pi}{2\pi}$ for $n = 2$ and $\frac{2+4\pi}{4\sqrt{3\pi}}$ for $n = 3$.

5.5 Time Bounds

Currently there is no proof of a polynomial upper bound for the BKZ algorithm in either Euclidean or Infinity norm. It is however known that it works very well in practice. It has a polynomial running time for a block size less than 25. For $\beta \geq 25$, where β is the block size, it is exponential in the dimension. In this chapter therefore no real time proof is presented, but only a basic description on how well the infinity norm enumeration works.

For the infinity norm, the analysis of the integrals, presented in the previous section, is very costly and does not give an exact solution for $n > 3$. The fraction of the volume is hence done with the help of the Monte-Carlo approximation. The conjunction of the small ball is about $(\frac{2+\pi}{2\pi})^{n-1}$th of the volume of the big ball. The rest of the costs for searching every knot of the search tree, which are received from tests, is linear. The full cost of the enumeration is thus reduced with an exponential factor.

A direct translation from the volume ration of the Gaussian volume heuristic results in an approximation of A_∞ of all cycles from step 2, which gives a lower value than the one from the test runs.

$$A_\infty \approx (\tfrac{2+\pi}{2\pi})^{k-j} A(j, n) + 2 \sum_{t=j+1}^{k} \tfrac{2+\pi}{2\pi}^{k-j} A(t, n))$$

Thus the sum of all arithmetical operations is

$$O(nA_\infty) \approx O(n \cdot ((\tfrac{2+\pi}{2\pi})^{k-j})A(j,n) + 2\sum_{t=j+1}^{k} \tfrac{2+\pi}{2\pi}^{k-j} A(t,n))$$

References

[1] Hervé Daudé, Philippe Flajolet, and Brigitte Vallée. An analysis of the gaussian algorithm for lattice reduction. pages 144–158, 1994.

[2] Michael Kaib. The gaußlattice basis reduction algorithm succeeds with any norm. pages 275–286, 1991.

[3] Michael Kaib and Harald Ritter. Block reduction for arbitrary norms. 1994.

[4] Michael Kaib and Claus P. Schnorr. The generalized gauss reduction algorithm. *J. Algorithms*, 21(3):565–578, 1996.

[5] Thorsten Koch. Zimpl user guide. 2006.

[6] A.K. Lenstra, H.W.jun. Lenstra, and László Lovász. Factoring polynomials with rational coefficients. *Math. Ann.*, 261:515–534, 1982.

[7] László Lovász and Herbert E. Scarf. The generalized basis reduction algorithm. *Math. Oper. Res.*, 17(3):751–764, 1992.

[8] Herbert E. Scarf and Laszlo Lovasz. The generalized basis reduction algorithm. (946), June 1990.

[9] C. P. Schnorr. A more efficient algorithm for lattice basis reduction. 1985.

[10] C. P. Schnorr. A hierarchy of polynomial time lattice basis reduction algorithms. *Theor. Comput. Sci.*, 53(2-3):201–224, 1987.

[11] C. P. Schnorr and M. Euchner. Lattice basis reduction: improved practical algorithms and solving subset sum problems. *Math. Program.*, 66(2):181–199, 1994.

[12] Brigitte Vallée. Gauss' algorithm revisited. *J. Algorithms*, 12(4):556–572, 1991.