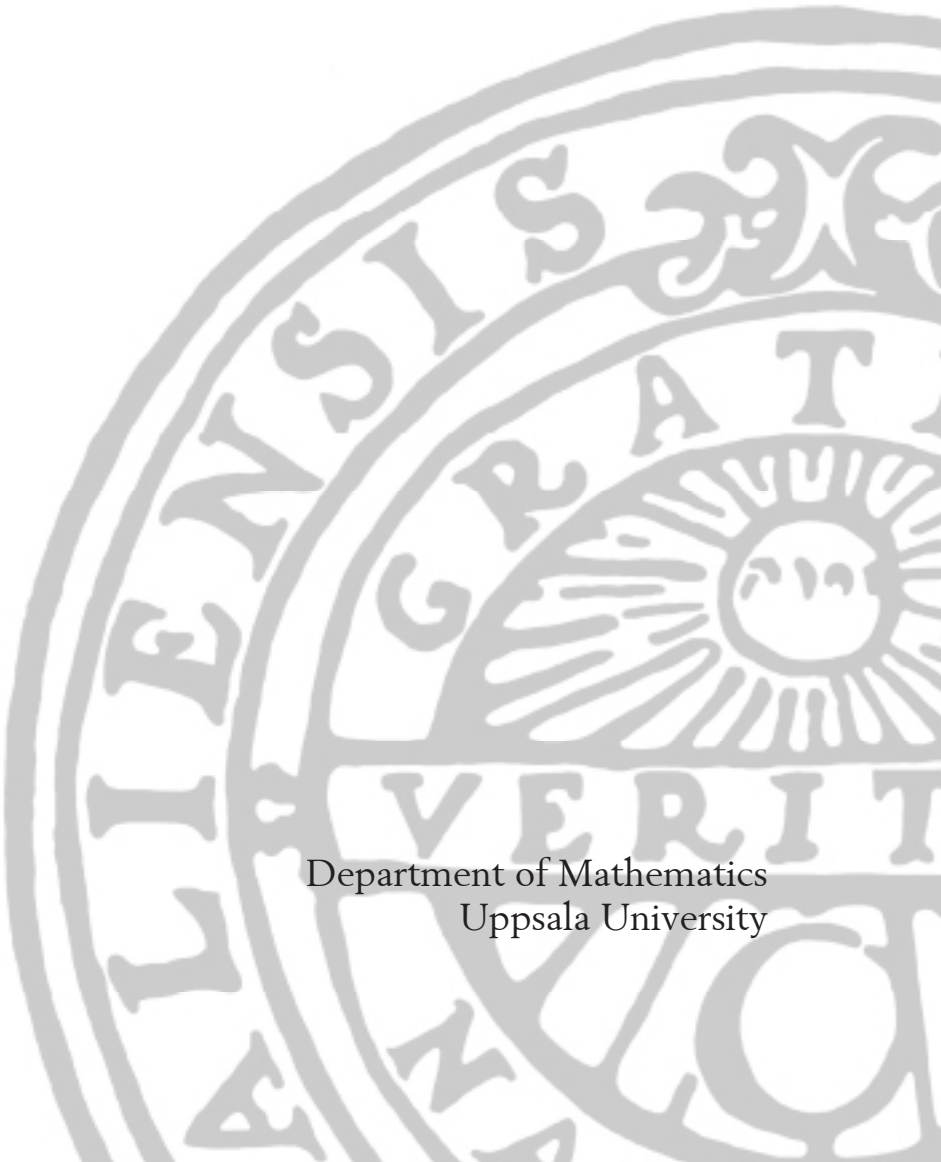
UPPSALA
UNIVERSITET

U.U.D.M. Project Report 2022:6

Lattice Basis Reduction Using LLL Algorithm with Application to Algorithmic Lattice Problems

Juraj Polách

Degree project E in mathematics
Supervisor: Silvelyn Zwanzig
Examiner: Julian Külshammer
February 2022



Department of Mathematics
Uppsala University

Lattice Basis Reduction Using LLL Algorithm with Application to Algorithmic Lattice Problems

Juraj Polách

February 25, 2022

Abstract

The LLL algorithm is recognized as one of the most important achievements of twentieth century with applications across many fields in mathematics. Yet, the subject is explained only on an advanced level in the current literature. We systematically present background and key concepts needed to understand the underlying problematic and afterwards give detailed proofs of all ideas needed to understand the algorithm. The paper focuses on application to lattice problems and proving results presented in the original paper of A. K. Lenstra et al. published in 1982.

Contents

1 Introduction

Shortest vector problem and closest vector problem, or so called lattice problems, are optimization problems in computer science related to additive groups called lattices.

Definition 1.1. A lattice L, is defined as the set of linear combinations of linearly independent vectors $v_1, \ldots, v_n \in \mathbb{R}^m$ $(m \geq n)$ with coefficients in $\mathbb{Z}$,

$$L = \{a_1 v_1 + a_2 v_2 + \cdots + a_n v_n : a_1, a_2, \ldots, a_n \in \mathbb{Z}\} \tag{1}$$

A very natural and intuitive example of a lattice is a simple grid. However, it turns out that extending the structure to n dimensions as by the definition above gives lattices rich combinatorial structure that makes it a powerful tool to address various problems within mathematics. These include integer programming, cryptography, factoring of polynomials over integers [8].

The most fundamental problems related to lattices: shortest vector problem and closest vector problem, are very difficult to solve. We start by outlining both of these:

Problem 1.1 (Shortest Vector Problem (SVP))**.** Find a (there can be multiple) shortest non-zero vector v in a given lattice L. The distance can be defined in any norm, but we will focus on the Euclidean norm i.e. $\|v\|$.

Problem 1.2 (Closest Vector Problem (CVP))**.** Given a vector $x \in \mathbb{R}^m$ such that $x \notin L$, find a (again there can be multiple) vector $v \in L$ that is closest to x. The distance can again be measured by any norm, but we focus on Euclidean norm i.e. we try to minimize $\|v - x\|$.

We visualize SVP, CVP and concept of lattices in 2 dimensions in Figure 1 [1] where the lattice points are indicated by a dot (notice that the lattice is not the entire collection $\mathbb{Z}^2$ represented by the grid). In Figure 1, solutions to SVP are vectors t and s. Solutions of CVP given vector v are both w and u.

It turns out that one of the most efficient ways to find at least an approximate solution to SVP and CVP is to manipulate basis vectors of the lattice in question in a specific way. This manipulation is referred to as a basis reduction. LLL algorithm developed by A. K. Lenstra et al. in 1982 [7] is the fundamental method still used to perform the basis reduction, even it was developed around 40 years ago. Despite its universal usefulness, the current literature is primarily for scholars that are already experts in the field. This paper presents the necessary background to the problematic as well as detailed proofs of all concepts needed to understand the LLL algorithm itself.

We show that using the LLL algorithm, one is able to find a vector that is no more than $(2/\sqrt{3})^n$ times longer than a shortest vector of a given lattice and present a method using which one can find a vector at most $2(2/\sqrt{3})^n$ times further away than a vector closest to a target vector outside the lattice. The n

[1] Figures in this paper are made with GeoGebra

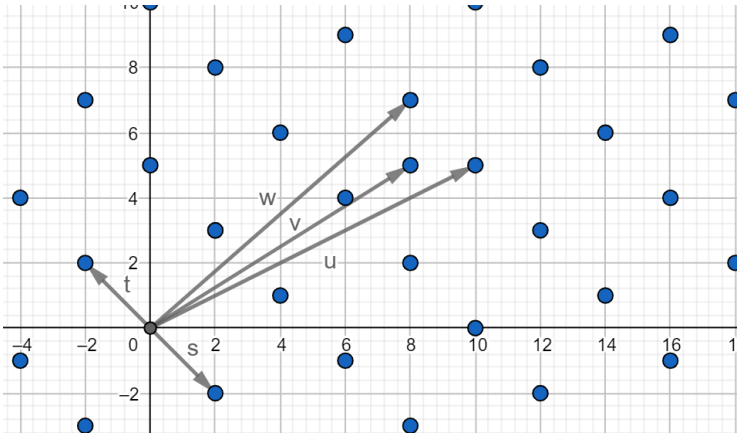


Figure 1: Visualisation of SVP and CVP

represents dimension of the given lattice. In contrast to the current literature, we present the LLL algorithm using 2 distinct subroutines while explaining how each relates to the SVP. Each subroutine impacts the basis vectors and their so called Gram-Schmidt orthogonalized counterparts in a different way. The main result of this paper is a detailed proof of how the subroutine of swapping basis vectors impacts the Gram-Schmidt orthogonalized vectors and the coefficients used to construct them. For the sake of completeness, we also provide a proof of why the LLL algorithm terminates and a proof of its polynomial run-time.

Section 2 formalizes the notion of how difficult it is to solve SVP and CVP using complexity theory. Section 3 introduces approximate versions of SVP and CVP and the concept of basis reduction used to solve these. The LLL algorithm used to find an LLL reduced basis is presented and analyzed in Section 4. We conclude by explaining main ways the LLL algorithm can and has been improved in Section 5.

2 Complexity Analysis of Lattice Problems

In order to understand the computational complexity of the SVP and CVP, we need to extend the definition of general optimization problem to a so called promise problem. This is a generalization of a decision problem where either "Yes" or "No" is an outcome for *every* input to a problem where there can be input instances for which neither "Yes" or "No" is the output. For these inputs we are allowed to output anything. Throughout this paper, we denote the shortest vector in a lattice L as $\lambda(L)$. In other words:

$$\lambda(L) = \arg\min\{\|x\| : x \in L, x \neq 0\}$$

Furthermore, when we write basis vectors of the lattice L as columns of a matrix, say B, we can also discuss a lattice generated by the matrix B. In such

case, we can write:

$$L(B) = \{Bx : x \in \mathbb{Z}^n\}$$

which is equivalent to the definition from equation (1).

Problem 2.1 (Promise SVP)**.** We refer to instances as pairs (B, d) with $B \in \mathbb{Z}^{m \times n}$ as basis of L and $d \in \mathbb{R}$ positive number. We then distinguish "Yes" or "No" cases as:

1. (B, d) is a "Yes" instance if $\lambda(L) \leq d$
2. (B, d) is a "No" instance if $\lambda(L) > d$

Problem 2.2 (Promise CVP)**.** We refer to instances as triples (B, x, d) with $B \in \mathbb{Z}^{m \times n}$ as basis of L and $d \in \mathbb{R}$ positive number and a vector $x \in \mathbb{R}^m$ such that $x \notin L$. We then distinguish "Yes" or "No" cases as:

1. (B, x, d) is a "Yes" instance if $\|x - z\| \leq d$ for some $z \in L$
2. (B, x, d) is a "No" instance if $\|x - z\| > d$ for all $z \in L$

During this paper, we do not address any algorithmic challenges with implementing algorithms to solve the problems above, but analyze their complexity from a purely mathematical point of view. As such, we are interested in whether these problems can be solved efficiently or not. From the point of view of complexity theory we distinguish between many complexity classes, but in this paper we shall focus primarily on the classes P and NP. The full formal definition of these classes is beyond this paper as well, but we shall outline key concepts in order to provide intuition for the reader.

We describe complexity of a given problem in terms of a model of computation. A natural reference point are Turing machines. For purposes of this paper, we define 4 types of Turing machine using definitions presented by D. S. Johnson [5]:

1. **Deterministic Turing Machines (DTMs)** These machines read input from one tape, have a finite size 'work tape' in order to perform operations and an output tape. They eventually halt and the output (solution) can be read from the output tape. In this case, we are primarily interested in *time* measured in number of steps before the machine halts and *space* measured in how much of the 'work tape' was visited at some point during the computation.
2. **Non-Deterministic Turing Machines (NDTMs)** These machines now have several choices where they should read the next command. This may result in different outcomes given one input. NDTMs can yield only 2 possible answers: "Yes" and "No". Thus, they can be only used for decision and promise problems.

3. **Random Turing Machines (RTMs)** These are NDTMs such that for each possible input there are either no accepting computations or at least half of all computations are accepting (output is "Yes" after the machine halts).

4. **Oracle Turing Machine (OTM)** This is a DTM, NDTM or RTM which is always associated with a given problem Y and the core idea is that the OTMs can solve this problem Y without any cost. It can also be viewed as invoking a subroutine for solving Y. Invoking this subroutine counts then as one step. We do note that the *space* is generally considered.

We can now describe complexity classes P, NP and R in terms of the Turing machines defined above. Class P can be seen as the class of problems that can be solved efficiently i.e. using a DTM in polynomial time in relation to the length of the input tape (more formally, string). Class NP is generally seen as the class of problems that are difficult to solve efficiently. In terms of Turing machines, we say that this is the set of decision problems solvable in polynomial-bounded time with a NDTM. It can also be useful to think of the class NP as a set of problems for which the solution can be checked by a DTM in polynomial time in relation to the input length. Finally, we introduce the class R (sometimes denoted also RP), which can be thought of as problems that can be solved efficiently, but with randomness involved. These algorithms are always correct on the "No" instances and 'usually' correct on "Yes" instances [8]. We remark that $\mathrm{P} \subseteq \mathrm{R} \subseteq \mathrm{NP}$.

In order to determine membership of a given problem to a complexity class, we need to simply show a method for solving the problem that meets requirements given by the class definition. As an example, by finding an algorithm that can run on a DTM in polynomial time we can conclude that the problem belongs to complexity class P. Many times, this approach turns out to be rather impractical and so we use a method called 'reduction'. For this purpose we use OTMs. As the name suggests, the key idea is to reduce a problem into another problem from a known class using a subroutine (Oracle). For example, if we have a DTM running in polynomial time solving problem X under the assumption that we have a DTM Oracle solving subroutine problem Y, we can conclude that X belongs to to class P if we are able to show that Y belongs to P (i.e. composition of polynomials is also a polynomial).

2.1 Complexity of Promise CVP

It was shown already in 1981 by P. van Emde Boas [13] that the promise CVP belongs to complexity class NP. The core idea of the paper was to show that promise CVP can be reduced to another problem already known to be of NP complexity: The subset sum problem outlined below.

Problem 2.3 (Subset Sum Problem)**.** Given $a_1, a_2, \ldots, a_n, s \in \mathbb{Z}$, find coefficients $y_i \in \{0, 1\}$ for $i = 1, 2, \ldots, n$ such that $\sum_i y_i a_i = s$. The corresponding promise problem is formulated by defining instances:

1. $(a_1, a_2, \ldots, a_n, s)$ is a "Yes" instance if there exist coefficients $y_i \in \{0, 1\}$ for $i = 1, 2, \ldots, n$ such that $\sum_i y_i a_i = s$

2. $(a_1, a_2, \ldots, a_n, s)$ is a "No" instance if there do not exist coefficients $y_i \in \{0, 1\}$ for $i = 1, 2, \ldots, n$ such that $\sum_i y_i a_i = s$

This problem is proven to be NP-hard. For reference see Garey and Johnson [3]. In order to outline the reduction itself, we refer to [9], where D. Micciancio and S. Goldwasser associate $a = [a_1, a_2, \ldots, a_n]$ to lattice basis vectors b_i and the target sum s to the target vector t as follows:

$$b_i = [a_i, \underbrace{0, \ldots, 0}_{i-1}, 2, \underbrace{0, \ldots, 0}_{i-1}]^T$$

$$t = [s, \underbrace{1, \ldots, 1}_{n}]^T$$

This way, the equivalent promise CVP problem instance in L_p-norm is given by the triplet $(B, t, \sqrt[p]{n})$ where the lattice defining matrix B is given as:

$$B = \begin{bmatrix} a \\ 2I_n \end{bmatrix}$$

where I_n is the identity matrix in n-th dimension and a is written as row vector. The reduction can be easily seen when looking at the difference $By - t$:

$$By - t = \begin{bmatrix} \sum_i y_i a_i - s \\ 2y_1 - 1 \\ \vdots \\ 2y_n - 1 \end{bmatrix}$$

If there exists $y \in \{0, 1\}^n$ such that $\sum_i y_i a_i = s$, the first entry is 0, remaining entries are ± 1 and the distance $\|Bx - t\|$ in L_p-norm is at most $\sqrt[p]{n}$. Conversely, assuming the promise CVP instance $(B, t, \sqrt[p]{n})$ is a "Yes" instance with vector $z \in \mathbb{Z}^n$ such that

$$\|Bz - t\|_p^p = |\sum_{i=1}^{n} z_i a_i - s|^p + \sum_{i=1}^{n} |2z_i - 1|^p \leq n \tag{2}$$

we see that the only way the inequality in equation (2) holds is if $\sum_{i=1}^{n} z_i a_i = s$ as $2z_i - 1 = \pm 1$ for all i.

2.2 Complexity of Promise SVP

In the previous section, we have demonstrated that promise CVP and in turn CVP itself is NP-hard. Despite obvious similarities between CVP and SVP, NP-hardness of the SVP problem remained an open problem for another 17 years after 1981 when P. van Emde Boas published the conjecture in his paper.

Breakthrough has been achieved for so called randomized reductions by M. Ajtai in 1998 [1]. The proof itself is well beyond the scope of this paper, thus in this section we shall only focus on what kind of reduction was used during the proof and its implications.

The first concept important to understand is 'Reverse Unfaithful Random' reduction, in short RUR-reduction. We will use the definition as per D. S. Johnson [5] where properties that must be satisfied are:

1. Outputs must be 'faithful' for "No" instances in a sense that if the answer is "No" in the promise SVP, corresponding outputs in the reduced problem are also "No" instances.

2. Outputs must be 'abundant' for "Yes" instances in a sense that if the answer is "Yes" in the promise SVP then at least $1/p(x)$ corresponding outputs in the reduced problem are "Yes" instances ($p(x)$ is a fixed polynomial depending on the length of input).

By repeating the probabilistic algorithm we can see that the probability that we would get a "No" instance in the reduced problem given that the corresponding solution to the promise SVP is a "Yes" instance is decreasing. Hardness under RUR-reductions gives evidence of intractability of SVP in a probabilistic sense. More concretely, it shows that the SVP is not in R unless R = NP [8]. The key result of [1] is that M. Ajtai shown that there is this probabilistic Turing machine which in polynomial time reduces any problem in NP to the SVP.

3 Lattices and Lattice Reduction

The previous two sections show that both SVP and CVP problems are very difficult to solve exactly. The natural next step is to look for an approximate solution. The approximation is given in relation to the dimension of Lattice:

Problem 3.1 (Approximate SVP)**.** For a given lattice L of dimension n find a non-zero vector v that is no more than $\psi(n)$ times longer than the shortest vector $\lambda(L)$. We can define the distance in any norm, but we will focus on the Euclidean norm i.e. $\|v\| \leq \psi(n)\|\lambda(L)\|$.

Problem 3.2 (Approximate CVP)**.** Given a vector $x \in \mathbb{R}^m$ such that $x \notin L$, find a vector $v \in L$ that is closest to x up to factor of $\psi(n)$. The distance can again be measured by any norm, but we focus on Euclidean norm i.e. if d is the distance to the closest vector in the lattice we try to find v such that $\|v - x\| \leq \psi(n)d$.

There are multiple ways one can approach the approximate problems above. As a basis for L is any set of independent vectors that generates L (and any two such sets have the same number of elements) [4], one of the most popular approaches is lattice reduction. The notion of reduced basis $v_1, v_2, \ldots, v_n \in \mathbb{R}^m$ for lattice L can be defined as follows:

Definition 3.1 (Reduced Basis)**.** We call a basis $v_1, v_2, \ldots, v_n \in \mathbb{R}^m$ for a lattice L reduced if it satisfies

$$|\mu_{ij}| \leq \frac{1}{2} \text{ for } 1 \leq j < i \leq n \tag{3}$$

and

$$||v_i|| \leq ||v_{i+1}|| \text{ for } 1 < i \leq n-1 \tag{4}$$

where μ_{ij} is the Gram-Schmidt coefficient defined in (6).

Definition 3.2 (Gram-Schmidt Orthogonalization)**.** Gram-Schmidt orthogonalization process gives mutually orthogonal vectors $v_i^* \in \mathbb{R}^m$ with $1 \leq i \leq n$ and numbers $\mu_{ij} \in \mathbb{R}$ with $1 \leq j < i \leq n$ defined inductively as

$$v_i^* = v_i - \sum_{j=1}^{i-1} \mu_{ij} v_j^* \tag{5}$$

$$\mu_{ij} = \frac{v_i \cdot v_j^*}{\|v_j^*\|^2} \tag{6}$$

where $v_1, v_2, \ldots, v_n \in \mathbb{R}^m$ is the basis of lattice L.

We can also write vectors $v_1^*, v_2^*, \ldots, v_n^*$ as columns of a matrix, say B_{GS}. Define also the matrix $U_{GS} \in \mathbb{R}^{n \times n}$ where the (i,j)th entry is μ_{ij} defined for $1 \leq j < i \leq n$ in equation (6). For $i = j$ we set the entries to 1 and for $i < j$ we set entries to 0. U_{GS} is upper triangular with diagonal entries equal to 1 and so $\det(U_{GS}) = 1$. Matrices B and B_{GS} can be also expressed as $B_{GS} = BU_{GS}$. Note, that these orthogonalized vectors are not necessarily a basis for the lattice L, but $\text{span}(B) = \text{span}(B_{GS})$. Thus, even though the orthogonalization and basis reduction are closely related, we need to keep clear distinction between them.

Expanding on the notion of reduced basis, the first condition set by equation (3) essentially says that it should be 'as orthogonal as possible'. This can be seen on Figure 2 where the vector v_2^* is orthogonal to v_1 and v_2' is a reduced vector that is not orthogonal to v_1, but is as orthogonal as possible with $\mu_{21} \leq \frac{1}{2}$. The second condition set by equation (4) essentially says that vectors should be ordered from shortest to longest. The solution to approximate SVP given a reduced basis is by construction v_1, the shortest vector in the reduced basis. How good the approximation is, is determined by how 'well' we are able to reduce the basis and on a particular reduction algorithm. We are able to also solve the CVP using a reduced basis and reduction algorithms, but here the goodness of approximation doesn't only depend on how 'well' we are able to reduce the basis, but also on what we do afterwards. During this paper, we will present two methods:

1. Using Babai's Closest Vertex Algorithm as outlined by Hoffstein et al. [4] to provide basic intuition.

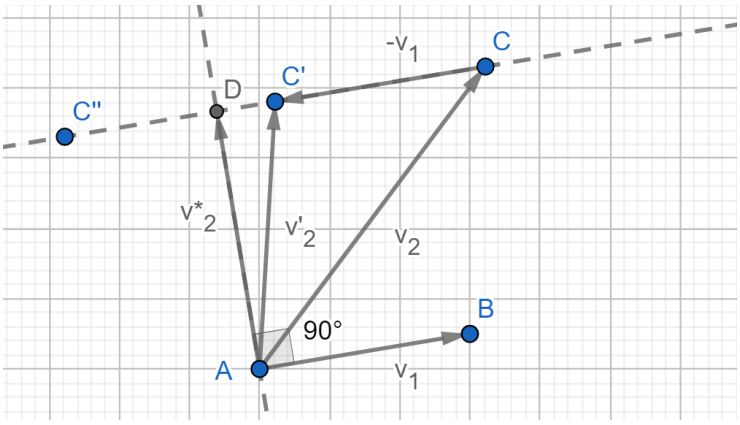


Figure 2: Difference between basis reduction and orthogonalization

2. Using the celebrated LLL algorithm proposed by A. K. Lenstra et al. in 1982 [7].

We will describe both of these methods assuming the lattice is of full rank. In other words, when $\text{span}(B) = \mathbb{R}^n$ which is the case when $m = n$ for basis vectors such that $v_1, v_2, \ldots, v_n \in \mathbb{R}^m$. Both methods can also be used in the more general case where $m > n$, but then we need to project w to the closest point in $\text{span}(B)$ and look for closest vector to that point instead.

Algorithm 1 Babai's Closest Vertex Algorithm

Let L be a lattice with basis vectors $v_1, v_2, \ldots, v_n \in \mathbb{R}^n$. We denote the matrix B as the matrix with columns as the basis vectors of the lattice. Furthermore, let $w \in \text{span}(B) = \mathbb{R}^n$ be an arbitrary vector. The algorithm executes in 3 steps:

1. Express w in terms of the basis vectors i.e. write $w = Bt$ for $t \in \mathbb{R}^n$.
2. Calculate $a \in \mathbb{R}^n$ as $a_i = \lceil t_i \rfloor$ for $i = 1, 2, \ldots, n$.
3. Return vector $v = Ba$

where the operator $\lceil x \rfloor$ returns the integer closest to $x \in \mathbb{R}$.

We begin with Babai's closest vertex algorithm given as Algorithm 1 and demonstrate its underlying idea in the Figure 3. In the figure, we are trying to express the vector v in terms of the two basis vectors: v_1 and v_2. As indicated by the parallel line p to v_1, the way to express the target vector v using the basis vectors is to take close to $2v_2$ and a bit less than $\frac{1}{2}v_1$ resulting in the vector $\overrightarrow{OF}$. It is clear that the algorithm only performs well in case we have a relatively 'good' basis in terms of both the orthogonality as well as length of

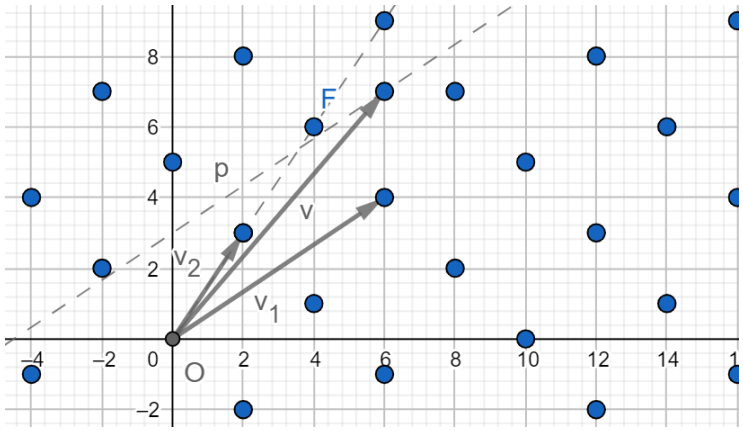


Figure 3: Demonstration of Babai's algorithm

basis vectors. We note that even the basis of the lattice is relatively 'good' used in Figure 3, the algorithm arrives to a sub-optimal solution.

Babai's closest vertex algorithm demonstrates usefulness of basis reduction. There are many different basis reduction algorithms and improvements to these are made continuously. Interestingly though, the majority of these are still based on the LLL algorithm developed by A. K. Lenstra et al. in 1982 [7] and Korkine-Zolotareff reduction developed in 1986 see [6]. Clearly, the LLL algorithm is central in the theory of lattices and solving approximate SVP and CVP. For a more comprehensive overview of lattice basis reduction algorithms, we refer to C. P. Schnorr [11] and D. Micciancio and S. Goldwasser [9].

Before proceeding to the analysis of the LLL Algorithm, we outline a couple more concepts relevant in our discussion about lattices. Here we primarily refer to J. Hoffstein et al. [4] and D. Micciancio and S. Goldwasser [9].

Definition 3.3 (Fundamental Domain and Determinant of Lattice)**.** Let L be a lattice with basis vectors $v_1, v_2, \ldots, v_n \in \mathbb{R}^m$. We denote the matrix B as the matrix with basis vectors of the lattice as its columns. Determinant of L, denoted $\det(L)$, is the n-dimensional volume of the open parallelepiped $F(B)$:

$$F(B) = \{Bt : 0 \leq t_i \leq 1\}$$

The open parallelepiped $F(B)$ is called the fundamental domain of lattice L.

As an example, for vectors v_1 and v_2 in Figure 4, the fundamental domain is the area $ABC'C$.

An important property of any lattice and its determinant is that the determinant is independent of the basis vectors used to generate the lattice. An algebraic result of this is the fact that two bases $B, B' \in \mathbb{R}^{m \times n}$ are equivalent (span the same lattice) if and only if there exists a matrix $U \in \mathbb{Z}^{n \times n}$ such

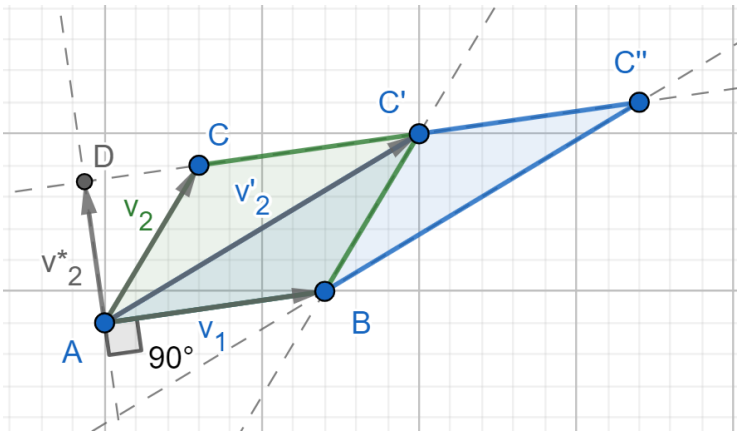


Figure 4: Fundamental domain

that $B' = BU$ and $\det(U) = \pm 1$. In other words, we can express each basis by starting from the other one and sequentially (not simultaneously) adding or subtracting respective basis vectors to and from each other. For example, we can consider Figure 4 where we could instead of the basis vectors v_1 and v_2 take basis vectors v_1 and $v_2' := v_2 + v_1$. The fundamental domain of these two vectors ($ABC''C'$) has the same area as the original two vectors as the height v_2^* of the parallelepiped does not change. The determinant of the lattice L is also equal to the square root of the determinant of the so-called Gram matrix $B^T B$, which is an $n \times n$ matrix with its (i, j)th entry given as inner product of v_i and v_j. The name relates to the orthogonality (or in other words projection) coefficient defined in (6). We can now utilize the matrices U_{GS} and B_{GS} defined during the Gram-Schmidt orthogonalization process and summarize the above using the following equation

$$\det(L) = \sqrt{\det(B^T B)} \tag{7}$$

$$= \sqrt{\det((B_{GS}U_{GS}^{-1})^T (B_{GS}U_{GS}^{-1}))} \tag{8}$$

$$= \sqrt{\det((U_{GS}^{-1})^T)\det(B_{GS}^T B_{GS})\det(U_{GS}^{-1})} \tag{9}$$

$$= \sqrt{1 \cdot \prod_{i=1}^{n} ||v_i^*||^2 \cdot 1} = \prod_{i=1}^{n} ||v_i^*|| \tag{10}$$

where the vectors v_i^* are Gram-Schmidt orthogonalized vectors defined in equation (5) and $\det(B_{GS}^T B_{GS}) = \prod_{i=1}^{n} ||v_i^*||^2$ follows from the fact that the v_i^* are mutually orthogonal. We can visualize the intuition behind the above again in Figure 4 as the area (2-dimensional volume) of the fundamental domain can also be calculated as $\|v_2^*\| \times \|v_1\| = \|v_2^*\| \times \|v_1^*\|$ as $\|v_1\| = \|v_1^*\|$ by definition.

We now relate back to basis vectors $v_1, v_2, \ldots, v_n$ with Hadamard's inequality.

Lemma 3.1 (Hadamard's Inequality)**.** *Let L be a lattice, take any basis $v_1, v_2, \ldots, v_n \in \mathbb{R}^m$ for L. Then*

$$\det(L) \leq \|v_1\|\|v_2\| \cdots \|v_n\|$$

The closer a basis $v_1, v_2, \ldots, v_n$ is to being orthogonal, the closer Hadamard's inequality comes to being an equality. Thus, one way to look at reducing a basis is to try to bring this inequality as close as possible to equality. With these terms defined, we are ready to formalize concept of an LLL reduced basis in the next section.

4 LLL Reduced Basis

We now have the intuition behind how a reduced basis helps to solve the SVP and CVP. When working with the LLL reduced basis and LLL Algorithm, we change the original idea of increasing length of basis vectors defined in (4), and instead start working with Gram-Schmidt orthogonalized vectors $v_1^*, v_2^*, \ldots, v_n^*$.

Definition 4.1 (LLL Reduced Basis)**.** As in A. K. Lenstra et al. [7], we define an LLL reduced basis $v_1, v_2, \ldots, v_n \in \mathbb{R}^m$ for a lattice L as a basis that satisfies

$$\text{(Size Condition)} \quad |\mu_{ij}| \leq \frac{1}{2} \text{ for } 1 \leq j < i \leq n$$

$$\text{(Lovász Condition)} \quad ||v_i^* + \mu_{ii-1}v_{i-1}^*||^2 \geq \frac{3}{4}||v_{i-1}^*||^2 \text{ for } 1 < i \leq n$$

where μ_{ij} is a Gram-Schmidt coefficient defined in equation (6) and $v_1^*, v_2^*, \ldots, v_n^* \in \mathbb{R}^m$ are Gram-Schmidt orthogonalized vectors defined in (5).

We at this point remark that the constant $\frac{3}{4}$ in (Lovász Condition) can be replaced by any real number $\delta \in (\frac{1}{4}, 1)$, but in this paper we shall focus on calculations with $\frac{3}{4}$. We will see later on in equation (31) that the idea behind the left hand side of (Lovász Condition) is to look at the length of the Gram-Schmidt orthogonalized vector of the basis vector v_i if it would switch position with v_{i-1}. The parameter δ thus represents a potential imperfection as lengths of Gram-Schmidt orthogonalized vectors v_i^* can decrease as i increases. The idea is that if they would, they would do so slowly. It remains an open problem to see if the LLL algorithm terminates for $\delta = 1$, which would get rid of this imperfection. We can also compare the notion of an LLL reduced basis and basis reduced by Definition 3.1. It turns out that even these are closely related, one does not necessarily imply the other. To demonstrate this, we use the following two examples. First, let v_1, v_2, v_3 be defined as

$$\begin{bmatrix} v_1 & v_2 & v_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 2 \\ 0 & 0 & \epsilon \end{bmatrix}$$

where $0 < \epsilon < 2\sqrt{\delta}$. Clearly v_1, v_2, v_3 constitute a basis that is reduced by Definition 3.1, but $||v_3^* + \mu_{32}v_2^*||^2 = \epsilon^2 \leq 4\delta = \delta||v_2^*||^2$. Next, let v_1', v_2', v_3' be defined as

$$\begin{bmatrix} v_1' & v_2' & v_3' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2\delta + \epsilon' \end{bmatrix}$$

where $0 < \epsilon' < 2(1 - \delta)$. Clearly v_1', v_2', v_3' constitute a basis that is LLL reduced, but $\|v_2'\| > \|v_3'\|$.

We will now outline a couple of useful properties of any LLL reduced basis as introduced by A. K. Lenstra et al. [7] in order to make the connection to approximate SVP.

Lemma 4.1. *Let a basis $v_1, v_2, \ldots, v_n \in \mathbb{R}^m$ for a lattice L be LLL reduced and let $v_1^*, v_2^*, \ldots, v_n^*$ be defined by the Gram-Schmidt orthogonalization process of this basis. We then have*

$$||v_j||^2 \leq 2^{i-1}||v_i^*||^2 \text{ for } 1 \leq j \leq i \leq n, \tag{11}$$

$$\det(L) \leq \prod_{i=1}^{n} ||v_i|| \leq 2^{n(n-1)/4}\det(L) \tag{12}$$

and

$$||v_1|| \leq 2^{(n-1)/4}\det(L)^{1/n} \tag{13}$$

Inequality (11), in Lemma 4.1 gives us a relation between the orthogonalized vectors v_i^* and the basis vectors v_i. In particular, we can see that the length of basis vectors is bounded by the length of the orthogonalized vectors, however the bound is exponentially weaker as we proceed through index i. This is easily seen in Figure 5 where even though $\|v_2^*\| \leq \|v_2\|$ (which is always the case) we see that $\sqrt{2}\|v_2^*\| \geq \|v_2\|$.

Inequality (12) tells us more about what is the worst case scenario when it comes to 'orthogonality' of an LLL reduced basis. Recall from Hadamard's Inequality (Lemma 3.1) that the closer the inequality is to equality, the more orthogonal we can expect basis vectors to be.

Most importantly though, we are able to define a boundary for the length of the first (shortest) vector, v_1, in an LLL reduced basis using inequality (13). Note, that the determinant of a lattice $\det(L)$ is independent of basis we choose. Thus, this boundary can be seen as a general property of any LLL reduced basis. We also note that this does not give us any relation just yet to the length of the shortest vector $\lambda(L)$ in the lattice L.

If we work with the more general case of using δ instead of the constant $\frac{3}{4}$, powers of 2 in inequalities of Lemma 4.1 need to be replaced with powers of $4/(4\delta - 1)$ [7]. We leave the proof of the general case and instead proceed with the proof in case $\delta = \frac{3}{4}$.

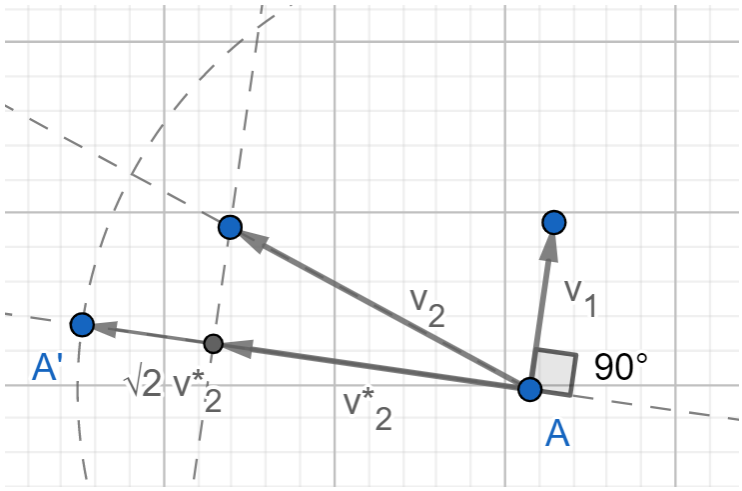


Figure 5: Boundary on size of vectors

Proof of Lemma 4.1 We first begin by proving the inequality (11), by using the definition of an LLL reduced basis. We can multiply out the square in left hand side of (Lovász Condition) as $v_i^* \cdot v_{i-1}^* = 0$ to arrive into a simplified version:

$$\begin{aligned} ||v_i^* + \mu_{ii-1}v_{i-1}^*||^2 &= ||v_i^*||^2 + 2\mu_{ii-1}v_i^* \cdot v_{i-1}^* + \mu_{ii-1}^2||v_{i-1}^*||^2 \\ &= ||v_i^*||^2 + \mu_{ii-1}^2||v_{i-1}^*||^2 \end{aligned}$$

implying

$$||v_i^*||^2 \geq (\frac{3}{4} - \mu_{ii-1}^2)||v_{i-1}^*||^2$$

where by (Size Condition) we have $\mu_{ii-1}^2 \leq 1/4$ and so $(\frac{3}{4} - \mu_{ii-1}^2) \geq \frac{1}{2}$. Now we can deduce by induction $||v_j^*||^2 \leq 2^{i-j}||v_i^*||^2$ for $1 \leq j \leq i \leq n$. Next, using the definition of Gram-Schmidt orthogonalized vector v_i^* in equation (5) and the fact that $v_i^* \cdot v_j^* = 0$ for $i \neq j$ we get:

$$\begin{aligned} ||v_i||^2 &= ||v_i^* + \sum_{j=1}^{i-1} \mu_{ij}v_j^*||^2 = ||v_i^*||^2 + \sum_{j=1}^{i-1} \mu_{ij}^2||v_j^*||^2 \\ &\leq ||v_i^*||^2 + \sum_{j=1}^{i-1} \frac{1}{4}2^{i-j}||v_i^*||^2 = ||v_i^*||^2(1 + \frac{1}{4}\sum_{j=1}^{i-1} 2^{i-j}) \\ &= ||v_i^*||^2(1 + \frac{1}{4}(2^i - 2)) = ||v_i^*||^2\frac{1}{2}(2^{i-1} + 1) \\ &\leq 2^{i-1}||v_i^*||^2 \end{aligned}$$

Inequality (11) now follows directly as we have shown earlier $||v_j^*||^2 \leq 2^{i-j}||v_i^*||^2$ for $1 \leq j \leq i \leq n$.

Proceeding further to (12), we can see that the first inequality is Hadamard's Inequality. The second inequality follows by taking the square root of (11) and using the result of equation (10) which says that the determinant of L can be calculated as $\det(L) = \prod_{i=1}^{n} ||v_i^*||$.

Inequality (13) follows if we take the square root of (11) and we multiply over $0 < i \leq n$ with $j = 1$. In other words,

$$\begin{aligned} ||v_1||^n &\leq \prod_{i=1}^{n} 2^{(i-1)/2}||v_i^*|| \\ &= 2^{n(n-1)/4} \prod_{i=1}^{n} ||v_i^*|| \\ &= 2^{n(n-1)/4}\det(L) \end{aligned}$$

Finally, taking the n-th root of both sides completes the proof. □

Lemma 4.1 laid the groundwork for us to now be able to finally give the 'goodness of approximation' result for approximate SVP. In particular, we are looking for a function $\psi(n)$ such that v_1 is at most $\psi(n)$ times longer than the shortest vector $\lambda(L)$ of the lattice L i.e. $\|v\| \leq \psi(n)\|\lambda(L)\|$.

Lemma 4.2. *Let basis $v_1, v_2, \ldots, v_n \in \mathbb{R}^m$ for lattice L be LLL reduced. Then*

$$||v_1||^2 \leq 2^{n-1}||x||^2 \tag{14}$$

for every $x \in L$ where $x \neq 0$.

Essentially, Lemma 4.2 is saying that the length of v_1, the first vector of an LLL reduced basis, is at most $2^{(n-1)/2}$ times the length of any vector $x \in L$, including a shortest vector. Again, we can replace the constant $\frac{3}{4}$ in (Lovász Condition) with δ to get the more general result $||v_1|| \leq (4/(4\delta-1))^{(n-1)/2}\lambda(L)$. It is also customary to work with $\delta = 1/4 + (3/4)^{n/(n-1)}$ to keep $\delta < 1$ ensuring polynomial running time of the LLL algorithm. This way, we obtain the more simplified version $||v_1|| \leq (2/\sqrt{3})^n\lambda(L)$. This results in the approximation function $\psi(n) = (2/\sqrt{3})^n$. Even though ψ is exponential in the rank of the lattice, this result is still an achievement as the approximation function is independent of the input size and the LLL algorithm allowed for the first time to solve SVP exactly in fixed dimension [9].

Proof of Lemma 4.2 Let $v_1^*, v_2^*, \ldots, v_n^*$ be defined by the Gram-Schmidt orthogonalization process of $v_1, v_2, \ldots, v_n$. Using the definition of a lattice we can write

$$x = \sum_{i=1}^{n} r_i v_i = \sum_{i=1}^{n} r_i^* v_i^*$$

for some $r_i \in \mathbb{Z}$, $r_i^* \in \mathbb{R}$ with $i = 1, 2, \ldots, n$. Let k be the smallest integer such that $r_i = 0$ for all $i > k$. Then we have

$$x = \sum_{i=1}^{k} r_i v_i = \sum_{i=1}^{k} r_i^* v_i^*$$

with $|r_k| \geq 1$. In other words, we express x using only the first k vectors of the basis. As this basis is LLL reduced, these are with high likelihood also k shortest vectors of the basis. Furthermore, as $v_j \cdot v_i^* = v_j^* \cdot v_i^* = 0$ for $j < i$ and $v_j^* \cdot v_i^* = 0$ for $j > i$, we can see that

$$\begin{aligned} x \cdot v_k^* &= \sum_{i=1}^{k} r_i v_i \cdot v_k^* = \sum_{i=1}^{k-1} r_i v_i \cdot v_k^* + r_k v_k \cdot v_k^* = r_k v_k \cdot v_k^* \\ &= \sum_{i=1}^{k} r_i^* v_i^* \cdot v_k^* = \sum_{i=1}^{k-1} r_i^* v_i^* \cdot v_k^* + r_k^* v_k^* \cdot v_k^* = r_k^* v_k^* \cdot v_k^* \end{aligned}$$

In brief, $x \cdot v_k^* = r_k v_k \cdot v_k^* = r_k v_k^* \cdot v_k^* = r_k^* v_k^* \cdot v_k^*$. As $r_k^* = r_k \geq 1$ we have

$$||x||^2 = ||\sum_{i=1}^{k} r_i^* v_i^*||^2 = \sum_{i=1}^{k} (r_i^*)^2 ||v_i^*||^2 \tag{15}$$

$$\geq (r_k^*)^2 ||v_k^*||^2 \geq ||v_k^*||^2 \tag{16}$$

and by inequality (11) we have

$$||v_1||^2 \leq 2^{k-1} ||v_k^*||^2 \leq 2^{n-1} ||v_k^*||^2 \leq 2^{n-1} ||x||^2 \tag{17}$$

□

As a remark, the result of the more general case when using δ instead of $\frac{3}{4}$ can be seen by replacing the powers of 2 in equation (17) with $4/(4\delta - 1)$.

We finalize this section by extending the concept of the shortest vector in a lattice to the concept of successive minima.

Definition 4.2 (Successive Minima)**.** The ith minimum $\lambda_i(L)$ of a lattice L is the radius of the smallest sphere centered around the origin containing i linearly independent vectors $x_1, x_2, \ldots, x_i \in L$ i.e. for Euclidean norm we have

$$\lambda_i(L) = \max\{||x_1||, ||x_2||, \ldots, ||x_i||\}$$

For example, $\lambda_1(L)$ is the length of a shortest vector in lattice L. The next lemma gives a relation between lengths of vectors in an LLL reduced basis and successive minima of the corresponding lattice.

Lemma 4.3. *Let a basis $v_1, v_2, \ldots, v_n \in \mathbb{R}^m$ for a lattice L be LLL reduced. Furthermore, let $x_1, x_2, \ldots, x_l \in L$ be linearly independent. Then*

$$||v_i||^2 \leq 2^{n-1} max\{||x_1||^2, ||x_2||^2, \ldots, ||x_l||^2\}$$

for $i = 1, 2, \ldots, l$.

In other words, Lemma 4.3 says that $||v_i|| \leq 2^{(n-1)/2}\lambda_l(L)$ for the first l vectors in an LLL reduced basis (we will see that for the more general case when using δ we get $||v_i|| \leq (4/(4\delta - 1))^{(n-1)/2}\lambda_l(L)$ for the first l vectors in an LLL reduced basis). When we refer to the first l basis vectors, we are in high likelihood referring to l shortest vectors in the basis.

Proof of Lemma 4.3 The trick lies in careful renumbering of x_i and using the linear independence between x_i. As in the previous lemma, we have $x_j = \sum_{i=1}^{n} r_{ij}v_i$ for some $r_{ij} \in \mathbb{Z}$ with $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, l$. For fixed j let $k(j)$ be smallest integer such that $r_{ij} = 0$ for all $i > k(j)$. Consider now inequalities (15) and (16). Rewriting yields:

$$\begin{aligned} ||x_j||^2 = ||\sum_{i=1}^{k(j)} r_{ij}^* v_i^*||^2 &= \sum_{i=1}^{k(j)} (r_{ij}^*)^2 ||v_i^*||^2 \\ &\geq (r_{k(j)j}^*)^2 ||v_{k(j)}^*||^2 \geq ||v_{k(j)}^*||^2. \end{aligned}$$

We thus have $||v_{k(j)}^*||^2 \leq ||x_j||^2$ for $j = 1, 2, \ldots, l$.

Renumber x_j such that $k(1) \leq k(2) \leq \cdots \leq k(l)$. The claim is that $j \leq k(j)$ for all $j = 1, 2, \ldots, l$. We show this by contradiction. Assume that there exists j such that $j > k(j)$, $1 \leq j \leq l$. We would then have $x_1, x_2, \ldots, x_j$ that can all be expressed by the linear combinations of $x_1, x_2, \ldots, x_{k(j)-1}$ this is less than j and as such in contradiction with $x_1, x_2, \ldots, x_l \in L$ being linearly independent. From this and (11) we have

$$||v_j||^2 \leq 2^{k(j)-1}||v_{k(j)}^*||^2 \leq 2^{k(j)-1}||x_j||^2 \leq 2^{n-1}||x_j||^2$$

for $j = 1, 2, \ldots, l$ completing the proof. □

4.1 LLL Algorithm

We follow the original outline of the LLL algorithm presented by A. K. Lenstra et al. in [7] as closely as possible in order to be able to give more detailed proofs of claims presented in their paper. We first outline the algorithm on a high level to introduce its operations as well as intuition behind how each step gets us closer to a solution of approximate SVP. In order to analyze the algorithm, we give properties of basis vectors during and after termination of the algorithm as lemmas. These lemmas are eventually all proven later on in the paper. At the end of this section, we will extend the algorithm to find a solution of approximate CVP.

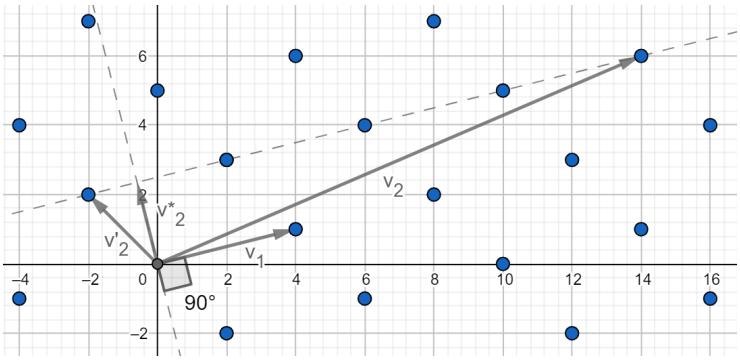


Figure 6: Visualization of the LLL algorithm in relation to the SVP

Algorithm 2 LLL Algorithm

We are given an arbitrary basis $v_1, v_2, \ldots, v_n \in \mathbb{R}^m$ for a lattice L and the LLL algorithm outputs a new basis that is LLL reduced. During the algorithm we keep track of a current subscript k, $k = 2, \ldots, n, n+1$. The algorithm is initialized by computing Gram-Schmidt orthogonalized vectors $v_1^*, v_2^*, \ldots, v_n^*$, their lengths, respective μ_{ij} for $1 \leq j < i \leq n$ and setting $k = 2$. After initialization, we proceed with step 1: size reduction.

1. **Size reduction**
   (a) If $k = n + 1$ terminate the algorithm and output the current set of vectors $v_1, v_2, \ldots, v_n$.
   (b) For $l = k-1, k-2, \ldots 1$ set $v_k := v_k - \lceil \mu_{kl} \rfloor v_l$ while updating respective μ_{ij}'s accordingly.
   (c) Check if $||v_k^* + \mu_{k,k-1} v_{k-1}^*||^2 \geq \frac{3}{4}||v_{k-1}^*||^2$ (Lovász Condition) holds
   If yes, set $k := k + 1$ and perform size reduction step from start.
   If no, perform swap step.
2. **Swap**
   (a) Set $\begin{pmatrix} v_k \\ v_{k-1} \end{pmatrix} := \begin{pmatrix} v_{k-1} \\ v_k \end{pmatrix}$ and update respective μ_{ij}'s and v_i^*'s
   (b) If $k > 2$ set $k := k-1$ and then perform size reduction step, otherwise just perform the size reduction step directly.

Observe, that throughout the algorithm the vectors $v_1, v_2, \ldots, v_n$ continuously change, but always in a way that in their entirety they remain a basis of L. We also note that the above outline of the LLL algorithm can be further optimized to achieve better computational performance. Computational

optimization of the algorithm is beyond the scope of this paper.

We visualize how each step in the LLL algorithm gets closer to a solution of SVP using Figure 6. If we consider the size reduction step on basis vectors v_1 and v_2 with $k = 2$, we can see that the vector v_2 would be size reduced to $v_2' = v_2 - 4v_1$. At this point, we can proceed to the check (in point 1.(c)) that will say that v_2^* is too long for v_2' to be the second vector in the basis and the new basis will be given by

$$\begin{pmatrix} v_1 \\ v_2 \end{pmatrix} := \begin{pmatrix} v_2' \\ v_1 \end{pmatrix}$$

during the swap step of the algorithm. After performing the size reduction step one more time in order to finalize the algorithm, we can see that the new v_1 $(= v_2')$ is indeed a shortest vector in the lattice L.

Before continuing the analysis, we want to check whether the output vectors (in point 1.(a)) indeed form an LLL reduced basis and whether the algorithm actually terminates. We need to show that the LLL algorithm terminates since the value of the sub-script k may, but also may not be increased during the size reduction step and it is decreased during the swap step. This is formalized in Lemma 4.4 and Lemma 4.5.

Lemma 4.4. *Output vectors $v_1, v_2, \ldots, v_n$ of the LLL algorithm form an LLL reduced basis of the lattice L i.e. satisfy:*

$$\textit{(Size Condition)} \quad |\mu_{ij}| \leq \frac{1}{2} \textit{ for } 1 \leq j < i \leq n$$

$$\textit{(Lovász Condition)} \quad ||v_i^* + \mu_{ii-1} v_{i-1}^*||^2 \geq \frac{3}{4} ||v_{i-1}^*||^2 \textit{ for } 1 < i \leq n$$

where μ_{ij} is the Gram-Schmidt coefficient defined in equation (6) *and $v_1^*, v_2^*, \ldots, v_n^* \in \mathbb{R}^m$ are Gram-Schmidt orthogonalized vectors defined in* (5)*.*

Lemma 4.5. *The LLL algorithm terminates.*

We give a proof of Lemma 4.4 directly, but will need to wait with a proof of Lemma 4.5 until we have defined how each step of the algorithm impacts the values of orthogonalized vectors v_i^* and the respective Gram-Schmidt coefficients μ_{ij}.

Proof of Lemma 4.4 Once the algorithm terminates, we have by construction of the swap step (Lovász Condition) satisfied for all $i = 1, 2, \ldots, n$. Furthermore, the size reduction step ensures that (Size Condition) $|\mu_{ij}| \leq \frac{1}{2}$ is satisfied for $1 \leq j < i \leq n$. To show this, we compute the updated values of μ_{kj} after size reducing ('orthogonalizing') v_k. Let $l = k - 1$ and $r_l = \lceil \mu_{kl} \rfloor$. From the definition of the Gram-Schmidt orthogonalization process, equation (6), we have coefficients μ_{kj} given as

$$\mu_{kj} = \frac{v_k \cdot v_j^*}{\|v_j^*\|^2} \text{ for } 1 \leq j < k \leq n.$$

From the definition of Gram-Schmidt orthogonalized vectors v_j^*, equation (5), we see that v_j^* is unaffected by updating v_k for $j = 1, 2, \ldots, k-1$ and so μ_{ij} remains unchanged for $i = 1, 2, \ldots, k-1$ and $j = 1, 2, \ldots, k-2$ (and also we see that (Lovász Condition) still remains satisfied for $i = 1, 2, \ldots, n-1$ after performing the last size reduction step). Updating $v_k := v_k - r_l v_l$ implies

$$\mu_{kj} = \frac{(v_k - r_l v_l) \cdot v_j^*}{\|v_j^*\|^2} = \frac{v_k \cdot v_j^*}{\|v_j^*\|^2} - r_l \frac{v_l \cdot v_j^*}{\|v_j^*\|^2} = \mu_{kj} - r_l \mu_{lj} \tag{18}$$

for $j = 1, 2, \ldots, l-1$ and

$$\mu_{kl} = \frac{(v_k - r_l v_l) \cdot v_l^*}{\|v_l^*\|^2} = \frac{v_k \cdot v_l^*}{\|v_l^*\|^2} - r_l \frac{v_l \cdot v_l^*}{\|v_l^*\|^2} = \mu_{kl} - r_l. \tag{19}$$

Clearly (19) implies that now $\mu_{k,k-1} \leq \frac{1}{2}$. We proceed by setting $l = k-2$ and $r_l = \lceil \mu_{kl} \rfloor$ from which (19) implies $\mu_{k,k-2} \leq \frac{1}{2}$. We can inductively see that also $\mu_{kj} \leq \frac{1}{2}$ for $j = 1, 2, \ldots k-1$. As the algorithm terminates once we reach $k = n+1$ we see that $|\mu_{ij}| \leq \frac{1}{2}$ is satisfied for $1 \leq j < i \leq n$ completing the proof. □

Note, that if we would replace $n+1$ in point 1.(a) of the LLL algorithm with l ($1 \leq l \leq n$), the output vectors $v_1, v_2, \ldots, v_{l-1}$ will constitute an LLL reduced basis for a sub-lattice of lattice L. We now formalize how the updated values of the basis vectors $v_1, v_2, \ldots, v_n$, the orthogonalized vectors $v_1^*, v_2^*, \ldots, v_n^*$ and the coefficients μ_{ij} are calculated through Lemma 4.6 and Lemma 4.7. It is from Lemma 4.7 we can see the intuition behind the (Lovász Condition) and how the new $\|v_{k-1}^*\|^2$ ($= \|c_{k-1}^*\|^2$) becomes less than $\frac{3}{4}$ of its original value.

Lemma 4.6 (Size reduction iteration)**.** *The new values of μ_{kj} are given as $\mu_{kj} - r\mu_{rj}$ for $j = 1, 2, \ldots, l-1$ and as $\mu_{kj} - r$ for $j = l$ during each iteration in the step 1.(b). All other μ_{ij} and v_i^* are invariant during this part of the algorithm.*

Lemma 4.7 (Swap)**.** *For a given k, $1 < k \leq n$ define*

$$c_i^* = v_i^* \quad \text{for} \quad i = 1, 2, \ldots, k-2, k+1, k+2, \ldots, n \tag{20}$$

$$c_{k-1}^* = v_k^* + \mu_{k,k-1} v_{k-1}^* \tag{21}$$

$$\nu_{k,k-1} = \mu_{k,k-1} \frac{\|v_{k-1}^*\|^2}{\|c_{k-1}^*\|^2} = \frac{v_k \cdot v_{k-1}^*}{\|c_{k-1}^*\|^2} \tag{22}$$

$$c_k^* = v_{k-1}^* - \nu_{k,k-1} c_{k-1}^* \tag{23}$$

$$\begin{pmatrix} \nu_{k-1,j} \\ \nu_{kj} \end{pmatrix} = \begin{pmatrix} \mu_{kj} \\ \mu_{k-1,j} \end{pmatrix} \quad \text{for} \quad j = 1, 2, \ldots k-2 \tag{24}$$

$$\begin{pmatrix}\nu_{i,k-1}\\ \nu_{ik}\end{pmatrix} = \begin{pmatrix}\mu_{ik}\|v_k^*\|^2/\|c_{k-1}^*\|^2 + \mu_{k,k-1}\nu_{k,k-1}\\ \mu_{i,k-1} - \mu_{ik}\mu_{k,k-1}\end{pmatrix} \text{ for } i = k{+}1, k{+}2, \dots n \quad (25)$$

$$\nu_{ij} = \mu_{ij} \text{ for } 1 \le j < i \le n, \{i,j\} \cap \{k, k-1\}. \quad (26)$$

The new values of μ_{ij} and v_i^* after the swap step are given as

$$v_i^* := c_i^* \text{ for } i = 1, 2, \dots, n \quad (27)$$

$$\mu_{ij} := \nu_{ij} \text{ for } 1 \le j < i \le n. \quad (28)$$

Proof of Lemma 4.6 Let $r_l = \lceil \mu_{kl} \rfloor$. During the proof of Lemma 4.4 we have seen that the vectors v_j^* are unaffected by updating v_k for $j = 1, 2, \dots, k-1$ and $l = k-1, k-2, \dots 1$ and so μ_{ij} remains unchanged for $i = 1, 2, \dots, k-1$ and $j = 1, 2, \dots, k-2$.

From equations (18) and (19) we see that the updated value of μ_{kj} is given by $\mu_{kj} := \mu_{kj} - r_l\mu_{lj}$ for $j = 1, 2, \dots, l-1$ and $\mu_{kl} := \mu_{kl} - r_l$.

We proceed by showing that also v_k^* is unaffected. Plugging in $v_k - r_l v_l$ for v_k, $\mu_{kj} - r_l\mu_{lj}$ for μ_{kj} with $j = 1, 2, \dots, l-1$ and $\mu_{kj} - r_l$ for μ_{kl} in the definition of Gram-Schmidt orthogonalized vector, equation (5), yields

$$\begin{aligned}
v_{k_{new}}^* &= v_k - r_l v_l - (\sum_{j=1}^{l-1}(\mu_{kj} - r_l\mu_{lj})v_j^* + (\mu_{kj} - r_l)v_l^* + \sum_{j=l+1}^{k-1}\mu_{kj}v_j^*)\\
&= v_k - r_l v_l - (\sum_{j=1}^{l-1}(-r_l\mu_{lj})v_j^* + (-r_l)v_l^* + \sum_{j=1}^{k-1}\mu_{kj}v_j^*)\\
&= v_k - \sum_{j=1}^{k-1}\mu_{kj}v_j^* - r_l v_l + r_l(\sum_{j=1}^{l-1}\mu_{lj}v_j^* + v_l^*)\\
&= v_k - \sum_{j=1}^{k-1}\mu_{kj}v_j^* - r_l v_l + r_l v_l = v_k^*.
\end{aligned}$$

We see that v_{k+1} remains unchanged, which enables us to verify that $\mu_{k+1,j}$ remains unchanged for $j = 1, 2, \dots, k$ which in turn enables us to see that v_{k+2} is unchanged. Inductively following these steps until v_{k+1} and $\mu_{n,n-1}$ we conclude the proof. □

Proof of Lemma 4.7 Clearly, $c_i^* = v_i^*$ for $1 \le i \le k-2$ as well as for $k+1 \le i \le n$. For $1 \le i \le k-2$ this follows from the fact that none of c_k, c_{k-1}, c_k^*, c_{k-1}^* is used during the Gram-Schmidt orthogonalization process. For $k+1 \le i \le n$ this follows from the fact that it geometrically does not make a difference with respect to which vector (c_k or c_{k-1}) one orthogonalizes first. In other words, c_k and c_{k-1} construct the same plane as v_k and v_{k-1} (regardless

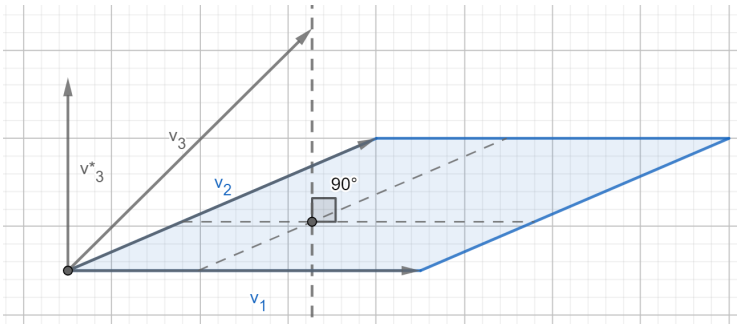


Figure 7: Orthogonalization with respect to plane generated by v_1 and v_2

what vector is used first during the construction). This is visualized on Figure 7.

For c^*_{k-1}, equation (21), we have

$$c^*_{k-1} = c_{k-1} - \sum_{j=1}^{k-2} \nu_{k-1,j} c^*_j = c_{k-1} - \sum_{j=1}^{k-2} \frac{c_{k-1} \cdot c^*_j}{\|c^*_j\|^2} c^*_j \tag{29}$$

$$= v_k - \sum_{j=1}^{k-2} \frac{v_k \cdot v^*_j}{\|v^*_j\|^2} v^*_j \tag{30}$$

$$= v_k - \sum_{j=1}^{k-1} \frac{v_k \cdot v^*_j}{\|v^*_j\|^2} v^*_j + \frac{v_k \cdot v^*_{k-1}}{\|v^*_{k-1}\|^2} v^*_{k-1} = v^*_k + \mu_{k,k-1} v^*_{k-1}. \tag{31}$$

For c^*_k, equation (23), we have

$$\begin{aligned} c^*_k &= c_k - \sum_{j=1}^{k-1} \nu_{kj} c^*_j = c_k - \sum_{j=1}^{k-2} \frac{c_k \cdot c^*_j}{\|c^*_j\|^2} c^*_j - \frac{c_k \cdot c^*_{k-1}}{\|c^*_{k-1}\|^2} c^*_{k-1} \\ &= v_{k-1} - \sum_{j=1}^{k-2} \frac{v_{k-1} \cdot v^*_j}{\|v^*_j\|^2} v^*_j - \frac{c_k \cdot c^*_{k-1}}{\|c^*_{k-1}\|^2} c^*_{k-1} \\ &= v^*_{k-1} - \nu_{k,k-1} c^*_{k-1} \end{aligned}$$

completing the proof for equations (20), (21), (23) and (27).

Moving on to μ's and ν's we first observe that $\nu_{ij} = \mu_{ij}$ for $1 \leq j < i \leq n$ such that $\{i, j\} \cap \{k, k-1\}$ as calculations of those don't involve either c^*_k or c^*_{k-1}. Due to that the updated value of v_k is simply v_{k-1} (and vice versa) we see that $\nu_{k-1,j} = \mu_{kj}$ and $\nu_{kj} = \mu_{k-1,j}$ for $j = 1, 2, \ldots, k-2$. This completes the proof of equations (24), (26) and their part of equation (28).

Matters become slightly more complicated for $\nu_{k,k-1}$ as

$$\nu_{k,k-1} = \frac{c_k \cdot c^*_{k-1}}{\|c^*_{k-1}\|^2} = \frac{v_{k-1} \cdot (v^*_k + \mu_{k,k-1} v^*_{k-1})}{\|c^*_{k-1}\|^2} \tag{32}$$

$$= \mu_{k,k-1} \frac{v_{k-1} \cdot v^*_{k-1}}{\|c^*_{k-1}\|^2} \tag{33}$$

$$= \mu_{k,k-1} \frac{\|v^*_{k-1}\|^2}{\|c^*_{k-1}\|^2} = \frac{v_k \cdot v^*_{k-1}}{\|c^*_{k-1}\|^2}. \tag{34}$$

where the step between (32) and (33) follows from the fact that $v_{k-1} \cdot v^*_k = 0$ and step between (33) and (34) from v^*_i being mutually orthogonal implying

$$v_k \cdot v^*_k = (v^*_k + \sum_{j=1}^{k-1} \mu_{kj} v^*_j) \cdot v^*_k = \|v^*_k\|^2.$$

This completes the proof of equation (22) and its part of equation (28).
We proceed by computing ν^*_{ik} and $\nu^*_{i,k-1}$ for $i > k$

$$\nu_{i,k-1} = \frac{c_i \cdot c^*_{k-1}}{\|c^*_{k-1}\|^2} = \frac{v_i \cdot (v^*_k + \mu_{k,k-1} v^*_{k-1})}{\|c^*_{k-1}\|^2} \tag{35}$$

$$= \frac{v_i \cdot v^*_k}{\|c^*_{k-1}\|^2} + \mu_{k,k-1} \frac{v_i \cdot v^*_{k-1}}{\|c^*_{k-1}\|^2} \tag{36}$$

$$= \frac{(v_i \cdot v^*_k)\|v^*_k\|^2}{\|v^*_k\|^2 \|c^*_{k-1}\|^2} + \mu_{k,k-1}\nu_{k,k-1} \tag{37}$$

$$= \mu_{ik} \frac{\|v^*_k\|^2}{\|c^*_{k-1}\|^2} + \mu_{k,k-1}\nu_{k,k-1} \tag{38}$$

where the step between (36) and (37) follows from (34). For ν_{ik} we start by observing $v_i = c_i$ for $i > k$. Using the definition of v^*_i from the Gram-Schmidt orthogonalization process, we can write

$$v^*_i + \sum_{j=1}^{i-1} \mu_{ij} v^*_j = v_i = c_i = c^*_i + \sum_{j=1}^{i-1} \nu_{ij} c^*_j. \tag{39}$$

We have earlier shown $v^*_i = c^*_i$ and $\nu_{ij} = \mu_{ij}$ for $1 \leq j < i \leq n$ such that $\{i, j\} \cap \{k, k-1\}$. Substituting this into equation (39) yields

$$\mu_{ik} v^*_k + \mu_{i,k-1} v^*_{k-1} = \nu_{ik} c^*_k + \nu_{i,k-1} c^*_{k-1}. \tag{40}$$

Next, we want to express v^*_k and v^*_{k-1} in terms of c^*_k and c^*_{k-1} due to c^*_k next to ν_{ik} on the right hand side of (40). We can express v^*_{k-1} by rearranging terms in equation (23) to obtain

$$v^*_{k-1} = c^*_k + \nu_{k,k-1} c^*_{k-1}. \tag{41}$$

Plugging in from (41) to (21) we get

$$
\begin{aligned}
v_k^* &= c_{k-1}^* - \mu_{k,k-1}(c_k^* + \nu_{k,k-1}c_{k-1}^*) && (42)\\
&= (1 - \mu_{k,k-1}\nu_{k,k-1})c_{k-1}^* - \mu_{k,k-1}c_k^* && (43)\\
&= (\frac{\|c_{k-1}^*\|^2}{\|c_{k-1}^*\|^2} - \mu_{k,k-1}\frac{v_k \cdot v_{k-1}^*}{\|c_{k-1}^*\|^2})c_{k-1}^* - \mu_{k,k-1}c_k^* && (44)\\
&= \frac{\|c_{k-1}^*\|^2 - (v_k \cdot v_{k-1}^*)^2/\|v_{k-1}^*\|^2}{\|c_{k-1}^*\|^2}c_{k-1}^* - \mu_{k,k-1}c_k^* && (45)\\
&= \frac{\|v_k^*\|^2}{\|c_{k-1}^*\|^2}c_{k-1}^* - \mu_{k,k-1}c_k^* && (46)
\end{aligned}
$$

where the step between (43) and (44) follows from expressing $\nu_{k,k-1}$ from equation (34), the step between (44) and (45) from simply expressing $\mu_{k,k-1}$ using its definition and the final step follows from expressing $\|c_{k-1}^*\|^2$ using (31) as

$$\|c_{k-1}^*\|^2 = \|v_k^*\|^2 + \mu_{k,k-1}^2\|v_{k-1}^*\|^2 = \|v_k^*\|^2 + \frac{(v_k \cdot v_{k-1}^*)^2}{\|v_{k-1}^*\|^2}. \quad (47)$$

Using the newly obtained expressions for v_k^*, v_{k-1}^* and $\nu_{i,k-1}$ from equations (41), (46) and (38) in equation (40) yields

$$
\begin{aligned}
&\mu_{ik}(\frac{\|v_k^*\|^2}{\|c_{k-1}^*\|^2}c_{k-1}^* - \mu_{k,k-1}c_k^*) + \mu_{i,k-1}(c_k^* + \nu_{k,k-1}c_{k-1}^*) =\\
&= \nu_{ik}c_k^* + (\mu_{ik}\frac{\|v_k^*\|^2}{\|c_{k-1}^*\|^2} + \mu_{k,k-1}\nu_{k,k-1})c_{k-1}^*.
\end{aligned}
$$

Subtracting $\mu_{ik}\frac{\|v_k^*\|^2}{\|c_{k-1}^*\|^2}c_{k-1}^*$ from both sides and rearranging terms yields

$$
\begin{aligned}
\nu_{ik}c_k^* &= \mu_{i,k-1}(c_k^* + \nu_{k,k-1}c_{k-1}^*) - \mu_{ik}\mu_{k,k-1}c_k^* - \mu_{k,k-1}\nu_{k,k-1}c_{k-1}^* && (48)\\
&= (\mu_{i,k-1} - \mu_{k,k-1})\nu_{k,k-1}c_{k-1}^* + (\mu_{i,k-1} - \mu_{ik}\mu_{k,k-1})c_k^*. && (49)
\end{aligned}
$$

Multiplying both sides by c_k^* sets the first term in (49) to zero as $c_k^*$and c_{k-1}^* are orthogonal and then dividing both sides by $\|c_k^*\|^2$ yields

$$\nu_{ik} = \mu_{i,k-1} - \mu_{ik}\mu_{k,k-1}$$

completing the proof of equations (25) and (28) and in turn concluding the proof of the entire lemma.

□

Proof of Lemma 4.5 Define values d_l for $l = 1, 2, \ldots, n$ and D.

$$d_l = \prod_{i=1}^{l} \|v_i^*\|^2$$

$$D = \prod_{l=1}^{n} d_l. \tag{50}$$

Further, define c_i^* for $i = 1, 2, \ldots, n$ and ν_{ij} for $1 \leq j < i \leq n$ as in Lemma 4.7. We show next that the swap step decreases the value of D by factor of less than $\frac{3}{4}$. In order to do so, we compute $\|c_k^*\|^2$.

$$\begin{aligned}
\|c_k^*\|^2 &= \|v_{k-1}^* - \nu_{k,k-1} c_{k-1}^*\|^2 && (51)\\
&= \|v_{k-1}^*\|^2 + \|\nu_{k,k-1} c_{k-1}^*\|^2 - 2\nu_{k,k-1} v_{k-1}^* c_{k-1}^* && (52)\\
&= \|v_{k-1}^*\|^2 + (\frac{v_k \cdot v_{k-1}^*}{\|c_{k-1}^*\|^2})^2 \|c_{k-1}^*\|^2 - 2\frac{v_k \cdot v_{k-1}^*}{\|c_{k-1}^*\|^2} v_{k-1}^* c_{k-1}^* && (53)\\
&= \|v_{k-1}^*\|^2 + \frac{(v_k \cdot v_{k-1}^*)^2}{\|c_{k-1}^*\|^2} - 2\frac{v_k \cdot v_{k-1}^*}{\|c_{k-1}^*\|^2} v_{k-1}^* (v_k^* + \mu_{k,k-1} v_{k-1}^*) && (54)\\
&= \|v_{k-1}^*\|^2 + \frac{(v_k \cdot v_{k-1}^*)^2}{\|c_{k-1}^*\|^2} - 2\frac{v_k \cdot v_{k-1}^*}{\|c_{k-1}^*\|^2} v_{k-1}^* (v_k - \sum_{j=1}^{k-2} \mu_{kj} v_j^*) && (55)\\
&= \|v_{k-1}^*\|^2 + \frac{(v_k \cdot v_{k-1}^*)^2}{\|c_{k-1}^*\|^2} - 2\frac{v_k \cdot v_{k-1}^*}{\|c_{k-1}^*\|^2} v_{k-1}^* v_k && (56)\\
&= \|v_{k-1}^*\|^2 - \frac{(v_k \cdot v_{k-1}^*)^2}{\|c_{k-1}^*\|^2} = \frac{\|v_{k-1}^*\|^2 \|c_{k-1}^*\|^2 - (v_k \cdot v_{k-1}^*)^2}{\|c_{k-1}^*\|^2} && (57)\\
&= \|v_k^*\|^2 \cdot \|v_{k-1}^*\|^2 / \|c_{k-1}^*\|^2 && (58)
\end{aligned}$$

where the step between (57) and (58) follows from expressing $\|c_{k-1}^*\|^2$ using (47). The Equation (58) implies

$$\|c_k^*\|^2 \|c_{k-1}^*\|^2 = \frac{\|v_k^*\|^2 \cdot \|v_{k-1}^*\|^2}{\|c_{k-1}^*\|^2} \|c_{k-1}^*\|^2 = \|v_k^*\|^2 \cdot \|v_{k-1}^*\|^2.$$

Thus, we can see that the only d_l that is impacted by the swap step is d_{k-1}. More concretely, we see that

$$\begin{aligned}
d_{k-1_{new}} &= \|c_{k-1}^*\|^2 \prod_{i=1}^{k-2} \|c_i^*\|^2 = \|c_{k-1}^*\|^2 \prod_{i=1}^{k-2} \|v_i^*\|^2 && (59)\\
&\leq \frac{3}{4} \|v_{k-1}^*\|^2 \prod_{i=1}^{k-2} \|v_i^*\|^2 = \frac{3}{4} d_{k-1_{old}} && (60)
\end{aligned}$$

where the step between (59) and (60) follows from (31) in Lemma 4.7 and (Lovász Condition) implying $\|c_{k-1}^*\|^2 \leq \frac{3}{4}\|v_{k-1}^*\|^2$. Consequently, $D_{new} \leq$

$\frac{3}{4}D_{old}$ and if we are now able to show that D has a lower bound larger than 0, we know that the swap step can be performed only finite amount of times. If we denote the amount of times the swap step is performed by K, we know that the size reduction step is performed $2K + n - 1$ times and then the algorithm terminates.

We now show that D has a lower bound larger than 0. From equation (10) we see that d_l is a square of the determinant of the sub-lattice L_l with basis vectors $v_1, v_2, \ldots, v_l$. Equation (13) in Lemma 4.1 gives us the upper bound for the first vector in an LLL reduced basis as

$$||v_1|| \leq 2^{(n-1)/4}\det(L)^{1/n}.$$

Vectors $v_1, v_2, \ldots, v_l$ do not need to constitute an LLL reduced basis for the sub-lattice L_l. Regardless, any vector in the sub-lattice L_l must be at least as long as the shortest vector $\lambda(L_l)$ of sub-lattice L_l. Thus, we have

$$\lambda(L_l) \leq 2^{(n-1)/4}\det(L_l)^{1/n}. \tag{61}$$

Expressing $\det(L_l)$ from equation (61) we get

$$\sqrt{d_l} = \det(L_l) \geq \frac{\lambda(L_l)^n}{2^{n(n-1)/4}} > 0.$$

We conclude that each d_l and thus also D has a lower bound larger than 0 depending only on the lattice L, K is finite and the algorithm terminates after $2K + n - 1$ times performing the size reduction step. □

4.2 Solving CVP Using the LLL Algorithm

We have now proven necessary properties of the LLL algorithm as well as outlined key ideas that we can apply to solve approximate CVP. In particular, we have seen that the LLL algorithm is able to return a solution to the Approximate SVP within a factor of $\psi(n) = (2/\sqrt{3})^n$ when using $\delta = 1/4+(3/4)^{n/(n-1)}$. Using the LLL Algorithm (also called the nearest plane algorithm) we are able to find a solution to approximate CVP within a factor of $\psi(n) = 2(2/\sqrt{3})^n$ when using $\delta = 1/4 + (3/4)^{n/(n-1)}$ [9]. The nearest plane algorithm is outlined as Algorithm 3.

Algorithm 3 The Nearest Plane Algorithm

Given an LLL reduced basis $v_1, v_2, \ldots, v_n \in \mathbb{R}^m$ for lattice L and a target vector x, an approximate closest vector $v \in L$ to x can be found using the following steps:

1. Project x orthogonally to span(B) and call the resulting point x_B
2. For $l = n, n-1, \ldots 1$ set $x_B := x_B - \lceil (x_B \cdot v_l)/\|v_l\|^2 \rfloor v_l$.
3. Output $x - x_B$

where lattice basis vectors constitute columns of matrix B.

In other words, we are running the size reduction step of the LLL algorithm as if the target vector x is a new vector in the basis of the lattice L and instead of the Gram-Schmidt orthogonalized vectors we are using lattice basis vectors. We visualize the steps during which the algorithm arrives to the vector that we in turn shall subtract from the target vector x to find solution to Approximate CVP in Figure 8. We start with the target vector x and size reduction step of it with respect to v_2 of the LLL reduced basis to arrive to vector $x' := x - 3v_2$. In the second step we are size reducing with respect to v_1. As $(x' \cdot v_1)/\|v_1\|^2 = 1/2$ we arrive to $x'' := x' - v_1$, but we can also stick with x'. The solution to Approximate CVP is then given by vector $x - x''$ (or $x - x'$ resulting in the second vector that is as close to x as the first one). Note however, that points O_1, R_1 and T_1 do not belong to the lattice.

4.3 Run-time Analysis of LLL Algorithm

The next natural question is whether we can in any meaningful way estimate the number of times size reduction or swap steps are made. We give the upper bound as the next lemma.

Lemma 4.8. *An upper bound on number of times the swap step is performed, denoted by K, is given by*

$$K = \mathcal{O}(n^2 \log B)$$

Proof of Lemma 4.8 Define the initial value of D from (50) as D_{start} and the value of D once the LLL Algorithm terminates D_{end}. Clearly,

$$\begin{aligned} D_{end} &\leq (\frac{3}{4})^K D_{start} \\ \log_{3/4} D_{end} &\geq K + \log_{3/4} D_{start} \\ \log(4/3)K &\leq \log D_{start} - \log D_{end} \\ K &\leq \frac{\log(D_{start}/D_{end})}{\log(4/3)} \end{aligned}$$

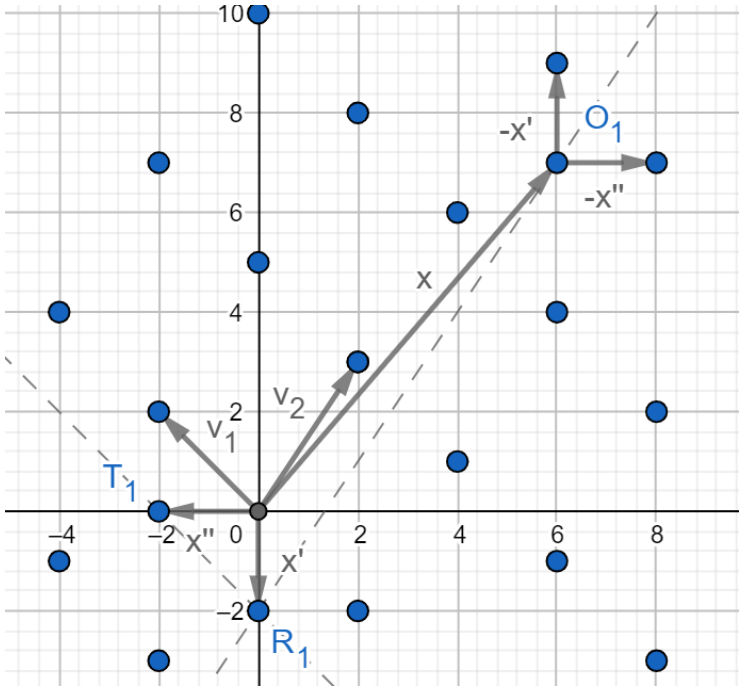


Figure 8: Visualization of the nearest plane algorithm

where flipping of inequalities follows from the fact that $\log_{3/4}(x) < 0$ for $x > 1$ and $\log(3/4) < 0$. If we now take D_{end} as a constant, we can see

$$K = \mathcal{O}(\log D_{start}).$$

One useful estimation of D_{start} is to use the length of the (not yet LLL reduced) basis vectors $v_1, v_2, \ldots, v_n$ for lattice $L \subset \mathbb{R}^n$ given in the beginning of the LLL Algorithm. Define $B = \max_{i=1,2,\ldots,n} \|v_i\|^2$. We can now observe

$$D_{start} = \prod_{l=1}^{n} d_l = \prod_{l=1}^{n} \prod_{i=1}^{l} \|v_i^*\|^2 \tag{62}$$

$$\leq \prod_{l=1}^{n} \prod_{i=1}^{l} \|v_i\|^2 \leq \prod_{l=1}^{n} \prod_{i=1}^{l} B \tag{63}$$

$$= \prod_{l=1}^{n} B^l = B^{n(n+1)/2} \tag{64}$$

where the step between (62) and (63) follows from $\|v_i\|^2 \geq \|v_i^*\|^2$ which we can see by expressing $\|v_i\|^2$ in terms of $\|v_i^*\|^2$ using (5) and observing that the term $\sum_{j=1}^{k-1} \mu_{kj}^2 \|v_j^*\|^2$ in (67) is a sum of squares

$$\|v_i\|^2 = \|v_k^* + \sum_{j=1}^{k-1} \mu_{kj} v_j^*\|^2 \tag{65}$$

$$= \|v_k^*\|^2 + \|\sum_{j=1}^{k-1} \mu_{kj} v_j^*\|^2 \tag{66}$$

$$= \|v_k^*\|^2 + \sum_{j=1}^{k-1} \mu_{kj}^2 \|v_j^*\|^2 \tag{67}$$

Each of the above steps follows from $v_i^* \cdot v_j^* = 0$ for $i \neq j$. We can now conclude that

$$K = \mathcal{O}(n^2 \log B).$$

□

From the relation of K to the initial value of D (D_{start}), we can directly see that one of the ways to decrease its upper bound is to decrease D_{start}. This relates to the structure of basis vectors at the initiation of LLL algorithm.

5 Conclusion

In order to make the LLL algorithm more approachable, we have taken the direction of basis reduction instead of factoring of polynomials presented in the original work of A. K. Lenstra et al. in [7]. In addition to presenting the subject in an intuitive way, contribution of this paper are also detailed proofs of how each step in the LLL algorithm impacts respective basis vectors outlined in Lemma 4.7. It is in fact the proof of Lemma 4.7 that gives the most intuition behind the algorithm, yet it was neglected in the current literature.

Further research and improvements to the LLL algorithm can be seen from two different perspectives:

1. Improving the theoretical worst case reduction result (measured for example by the approximation function $\psi(n)$ in the Approximate SVP and CVP problems)

2. Improving the computational efficiency of lattice reduction algorithms

Some of the best known algorithms improving the theoretical worst case reduction result for approximate SVP in polynomial time are block-wise algorithms that solve SVP exactly in low dimension (by splitting the basis into blocks), however the approximation factor of all of these algorithms remains exponential as in the LLL algorithm [10]. Another notable improvement to approximation factor is due to the probabilistic solution of Ajtai et. al. in 2001 [2] achieving $\psi(n) = 2^{O(n \ln \ln n / \ln n)}$, however the output of the algorithm is only guaranteed to be short with high probability.

When improving the computational efficiency, we are either concerned with how well the algorithm is implemented or decreasing the upper bound K for how many times the swap step is performed defined in Lemma 4.8. One of the improvements in this direction was reduction in segments presented by C. P. Schnorr in [12] where the efficiency in terms of bit operations was improved from $\mathcal{O}(n^{7+\epsilon})$ of the original LLL algorithm to $\mathcal{O}(n^3 \log n)$ for $\epsilon > 0$.

References

[1] M. Ajtai. The shortest vector problem in L_2 is NP-hard for randomized reductions,. *STOC '98: Proceedings of the thirtieth annual ACM symposium on Theory of computing, Pages 10–19*, 1998.

[2] M. Ajtai, R. Kumar, and D. Sivakumar. A sieve algorithm for the shortest lattice vector problem. *Proceedings on 33rd Annual ACM Symposium on Theory of Computing, STOC 2001, pages 266-275*, 2001.

[3] M. R. Garey and D. S. Johnson. *Computers and intractability: a Guide to the theory of NP-completeness*. W. H. Freeman, 1979.

[4] J. Hoffstein, J. Pipher, and J. H. Silverman. *An introduction to mathematical cryptography*. Springer, second edition, 2008.

[5] D. S. Johnson. A catalog of complexity classes. In *Handbook of Theoretical Computer Science, vol. A (Algorithms and Complexity)*, pages 67–161. Elsevier Science Publishers, Amsterdam, 1990.

[6] J. C. Lagarias, H. W. Lenstra, Jr., and C. P. Schnorr. Korkine-Zolotareff bases and successive minima of a lattice and its reciprocal lattice. *Tech Report MSRI 07718-86, Mathematical Sciences Research Institute, Berkeley.*, 1986.

[7] A. K. Lenstra, H. W. Lenstra, Jr., and L. Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 1982.

[8] D. Micciancio. The shortest vector in a lattice is hard to approximate to within some constant. *Society for Industrial and Applied Mathematics*, 2001.

[9] D. Micciancio and S. Goldwasser. *Complexity of lattice problems, a cryptographic perspective*. Kluwer Academic Publishers, 2002.

[10] P. Q. Nguyen and B. Vallée. *The LLL algorithm, survey and applications*. Springer, 2009.

[11] C. P. Schnorr. A hierarchy of polynomial time lattice basis reduction algorithms. *Theoret. Comput. Sci. 53 201-224*, 1987.

[12] C. P. Schnorr. Fast LLL-type lattice reduction. *Information and Computation*, 2006.

[13] P. van Emde Boas. Another NP-complete partition problem and the complexity of computing short vectors in a lattice. *Tech. Report 81-04, Mathematische Instituut, Universiry of Amsterdam*, 1981.