Basisreduktionsalgorithmen für Gitter kleiner Dimension

Dissertation
zur Erlangung des Grades
des Doktors der Naturwissenschaften
der Technischen Fakultät
der Universität des Saarlandes

vorgelegt von

Oliver van Sprang

Universität des Saarlandes
Fachbereich 14 - Informatik

Saarbrücken

1994

Inhaltsverzeichnis

Tabellenverzeichnis

Kapitel 1

Einführung

Diese Arbeit handelt von einem Teilbereich des faszinierenden Gebiets der *Geometrie der Zahlen*, der sogenannten *Reduktionstheorie.*

Das Interesse am Problem der Gitterreduktion ist durch seine Universalität begründet, das heißt die Lösung des Basisreduktionsproblems beinhaltet die Lösung vieler weiterer Probleme. Daraus ergibt sich ein großes Anwendungsgebiet von Reduktionsalgorithmen für Gitter von der algorithmischen Zahlentheorie bis hin zur Kryptographie. So können Reduktionsalgorithmen beispielsweise sowohl zur Bestimmung des erzeugenden Polynoms eines algebraischen Zahlkörpers aus seiner Multiplikationstabelle [2] als auch zur Widerlegung der Sicherheit von Knapsack und Public-Key Kryptosystemen [6, 21, 34] verwendet werden. Auch die lange unbewiesene Mertenssche Vermutung konnte nach achtjähriger Arbeit durch Anwendung von Gitterreduktion zum Auffinden von Gegenbeispielen, von Odlyzko und Te Riele [25] widerlegt werden.

Bevor bekannte und offene Probleme aus dem Gebiet geschildert und neue Beiträge und Ergebnisse präsentieren werden, werden im folgenden Abschnitt die Grundlagen dargestellt.

1.1 Grundlagen

Dieser Abschnitt hat zwei Ziele. Zum einen werden Definitionen für die Objekte der *Geometrie der Zahlen* gegeben und zum anderen werden Notationen eingeführt, die im Verlauf der Arbeit verwendet werden.

1.1.1 Grundlegende Notationen

Bezeichne $\mathbb{Q}, \mathbb{R}$ und $\mathbb{C}$ respektive den Körper der rationalen, reellen und komplexen Zahlen.

Für $z \in \mathbb{R}$ bezeichne $\lfloor z \rceil$ die nächste ganze Zahl an z. Die Eindeutigkeit für $z \in \mathbb{Z}/2$ erhält man durch folgende Definition:

$$\lfloor z \rceil = \begin{cases} \operatorname{sign}(z)(\lfloor |z| + 1/2 \rfloor) & \text{falls } z \in \mathbb{Z}/2 \\ \operatorname{sign}(z)(\lfloor |z| + 1/2 \rfloor - 1) & \text{sonst.} \end{cases}$$

Dabei ist für $z \in \mathbb{R}^{\geq 0}$ $\lfloor z \rfloor = \max\{x \in \mathbb{Z} : x \leq z\}$.

Es sei hier wie überall in dieser Arbeit $n \in \mathbb{N}$. Vektoren

$$\mathbf{v} = (v_1, \ldots, v_n)^t$$

seien immer *Spaltenvektoren*, also $n \times 1$ Matrizen. Vektoren und deren skalare Komponenten sind durch Fettdruck $\mathbf{v}$ und Normaldruck v_1 leicht zu unterscheiden. Matrizen werden mit großen lateinischen und deren Komponenten mit kleinen zweispaltig indizierten Buchstaben bezeichnet. Zum Beispiel: $M = (m_{ij})_{1 \leq i,j \leq n} \in \mathbb{Z}^{n \times n}$. Häufig wird auch die folgende Schreibweise verwendet: $M = (\mathbf{m}_1, \ldots, \mathbf{m}_n)$. Für zwei Vektoren $v, w \in \mathbb{R}^n$ bezeichne

$$\langle \mathbf{v}, \mathbf{w} \rangle = \mathbf{v}^t \cdot \mathbf{w} = \sum_{i=1}^{n} v_i w_i$$

das euklidische Skalarprodukt und

$$||\mathbf{v}|| = \sqrt{\langle \mathbf{v}, \mathbf{v} \rangle}$$

die von diesem Standardskalarprodukt induzierte Norm. Es gilt die *Dreiecksungleichung*:

$$|\,||\mathbf{v}|| - ||\mathbf{w}||\,| \leq ||\mathbf{v} + \mathbf{w}|| \leq ||\mathbf{v}|| + ||\mathbf{w}||$$

und die *Cauchy-Schwarzsche Ungleichung*

$$|\langle \mathbf{v}, \mathbf{w} \rangle| \leq ||\mathbf{v}|| \cdot ||\mathbf{w}||$$

für alle $\mathbf{v}, \mathbf{w} \in \mathbb{R}^n$.

1.1.2 Gitter und deren Repräsentation

Ein *Gitter* $L \subset \mathbb{R}^n$ ist eine diskrete additive Untergruppe des $\mathbb{R}^n$. Für ein Gitter L sind damit folgende Aussagen äquivalent:

i) 0 ist kein Häufungspunkt

ii) Für alle $r \in \mathbb{R}^{>0}$ ist die Menge $\{\mathbf{v} \in L : \| v \|^2 < r\}$ endlich.

Umgekehrt ist jede diskrete additive Untergruppe des $\mathbb{R}^n$ ein Gitter.

Jedes Gitter L kann geschrieben werden als

$$L = \sum_{j=1}^{k} \mathbf{b}_j \mathbb{Z},\ B = (\mathbf{b}_1, \ldots, \mathbf{b}_k) \text{ linear unabhängig}\ . \tag{1.1}$$

Die Matrix B heißt *Basis* des Gitters L. Alle Basen haben die gleiche Länge k, die auch *Rang* des Gitters genannt wird. Matrizen B, die die zuvor genannten Eigenschaften aufweisen, bezeichnen wir als *zulässige Matrizen.*

Gilt $k = n$ für den Rang k eines Gitters $L \subset \mathbb{R}^n$, so sagt man, das Gitter L hat *vollen* Rang. In dieser Arbeit werden häufig Gitter betrachtet, die vollen Rang haben. Ist eine zulässige Matrix B gegeben, so bezeichnen wir mit $L(B)$ das zu B korrespondierende Gitter. Zu gegebenem Gitter $L \subset \mathbb{R}^n$ existieren unendlich viele Basen. Denn ist B Basis des Gitters L, so ist auch $(B \cdot U)$ Basis des Gitters L für alle Matrizen $U \in \mathrm{GL}(k, \mathbb{Z})$, die als *unimodularen* Matrizen bezeichnet werden.

Gitterinvarianten

Zu jedem Gitter L gibt es Größen, die von der speziellen Wahl einer das Gitter darstellenden Basis unabhängig sind und somit aus jeder das Gitter L repräsentierenden Basis B berechnet werden können. Die Invarianten sind die *Gitterdeterminante* und die *sukzessiven Minima* des Gitters.

Definition 1.1 (Gitterdeterminante) *Sei $L \subset \mathbb{R}^n$ ein Gitter vom Rang $k \in \{1, \ldots, n\}$ und sei $B \in \mathbb{R}^{n \times k}$ eine Basis des Gitters L, dann ist die* Determinante $\Delta(L)$ *des Gitters L definiert als die positive Quadratwurzel*

$$\Delta(L) = \sqrt{B^t \cdot B}.$$

Geometrisch läßt sich die Gitterdeterminante als das Volumen des *fundamentalen Parallelepipeds*

$$P = \left\{ \mathbf{v} \in \mathbb{R}^n : \mathbf{v} = \sum_{i=1}^{k} x_i \, \mathbf{v}_i, x_i \in [0,1), \, 1 \leq i \leq k \right\}$$

interpretieren.

Wir führen die sogenannte *Hadamardschranke* $H(B)$ ein, die häufig verwendet wird. Die Hadamardschranke steht mit der Gitterdeterminante in folgendem Zusammenhang. Ist B Basis des Gitters L vom Rang k, so gilt:

$$\Delta(L)^2 \leq H(B) \leq c_k \, \Delta(L)^2.$$

Für $k \leq 8$ sind die kleinsten positiven reellen c_k, die sogenannten *Hermitekonstanten* bekannt. Für große k sind die kleinsten Werte für c_k von der Qualität $c_k \approx k^k$.

Definition 1.2 (Sukzessvive Minima) *Sei $L \subset \mathbb{R}^n$ ein Gitter vom Rang k und $i \in \{1, \ldots k\}$. Die kleinste positive reelle Zahl r mit der Eigenschaft, daß linear unabhängige Vektoren $\mathbf{v}_1, \ldots, \mathbf{v}_i \in L$ existieren mit*

$$||\mathbf{v}_j|| \leq r \;\; (1 \leq j \leq i),$$

wird das i-te sukzessives Minimum *des Gitters L genannt und mit $\lambda_i(L)$ bezeichnet.*

Wir lassen das Homonym "sukzessives Minimum" für einen Vektor $\mathbf{v} \in L$ mit $||\mathbf{v}|| = \lambda_i, i \in \{1, 2, \ldots, k\}$ zu, um umständliche Formulierungen zu vermeiden. Aus dem Kontext wird immer ersichtlich sein, ob sich der Begriff sukzessives Minimum auf den Skalar oder den Vektor bezieht.

Das erste sukzessive Minimum ist die Länge des kürzesten von Null verschiedenen Vektors in L.

Die sukzessiven Minima $\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_k$ sind geometrische Invarianten des Gitters. Sie ändern sich also unter isometrischen Transformationen nicht.

1.1.3 Gitter und quadratische Formen

Die von uns entwickelten Algorithmen TPR und FTPR können sowohl zur Reduktion von Gittern als auch zur Reduktion von Formen benutzt werden. Daher wird hier auf den Zusammenhang zwischen Gittern und positiv definiten quadratischen Formen eingegangen.

Definition 1.3 (Quadratische Form) *Sei* $k \in \mathbb{Z}^{\geq 2}$ *und* $\mathbf{x} = (x_1, \ldots, x_k)$. *Eine* quadratische Form $q(\mathbf{x})$ *ist ein symmetrisches homogenes Polynom vom Grad zwei in* k *Veränderlichen* $x_1, \ldots, x_k$*:*

$$q(\mathbf{x}) = \sum_{1 \leq i,j \leq k} q_{ij}\, x_i\, x_j.$$

Quadratische Formen werden gemäß des Vorzeichens ihrer Werte (Tabelle 1.1) klassifiziert, wobei $\mathbf{x}$ *alle nichtverschwindenden Vektoren mit reellen Einträgen durchläuft.*

Eigenschaften	*Bezeichung*
$q(\mathbf{x}) > 0$ *für alle* x	*positiv definite quadratische Form*
$q(\mathbf{x}) < 0$ *für alle* x	*negativ definite quadratische Form*
$\exists \mathbf{x}_1,\ \mathbf{x}_2 \in \mathbb{R}^k - \{0\}$ *mit* $q(\mathbf{x}_1) > 0$ *und* $q(\mathbf{x}_2) < 0$	*indefinite quadratische Form*

Tabelle 1.1: Klassifikation wichtiger quadratischer Formen

Für $k = 2$ *wird* $q(\mathbf{x})$ *als* binäre, *für* $k = 3$ *als* ternäre *quadratische Form bezeichnet.*

Es existiert eine Zuordnung von Gittern und Formen:

Abbildung eines Gitters auf eine positiv definite quadratische Form

Sei $L \subset \mathbb{R}^n$ ein Gitter vom Rang k und sei $B \in \mathbb{R}^{n \times k}$ eine Basis von L sowie $\mathbf{x} = (x_1, \ldots, x_k)$. Dann wird durch

$$q(\mathbf{x}) = \mathbf{x}^t \cdot (B^t \cdot B) \cdot \mathbf{x} \tag{1.2}$$

eine positiv definite quadratische Form definiert, da $q(\mathbf{x}) = \| B \cdot x \|^2$.

Abbildung einer positiv definiten quadratische Form auf ein Gitter

Sei $\mathbf{x} = (x_1, \ldots, x_k)$ und $q(\mathbf{x})$ eine positiv definite quadratische Form. Man kann Koeffizienten $a_{ij} \in \mathbb{R}$ bestimmt, so daß gilt:

$$q(\mathbf{x}) = \sum_{i=1}^{k} a_{ii} \cdot \left(x_i + \sum_{j=i+1}^{k} a_{ij} x_j \right)^2 .$$

Da die Form $q(\mathbf{x})$ positiv definit ist, muß $a_{ii} > 0$ sein. Die Spaltenvektoren der Matrix

$$B^t = U \cdot \mathrm{diag}(\sqrt{a_{11}}, \ldots, \sqrt{a_{kk}}) \in \mathbb{R}^{k \times k}$$

bilden eine Basis eines Gitters vom Rang k. Hierbei ist $U \in \mathbb{R}^{k\times k}$ folgendermaßen definiert:

$$U = (u_{ij})_{1\le i,j\le k} \begin{cases} a_{ij} & \text{für } 1 \le i < j \le k \\ 1 & \text{für } 1 \le i = j \le k \\ 0 & \text{sonst.} \end{cases}$$

1.1.4 Reduktionstheorie

Ein wichtiger Aspekt der Reduktionstheorie ist es, Hilfsmittel zur Verfügung zu stellen, um die sukzessiven Minima eines Gitters zu bestimmen und zu gegebenem Gitter eine Basis zu berechnen, die kurze euklidische Längen aufweist. Diese Probleme sind seit über hundert Jahren Gegenstand der Forschung.

Um von der vagen Forderung *kurzer euklidischer Längen*, die eine *reduzierte* Basis erfüllen soll, zu quantitativen Aussagen zu gelangen, wurden diverse Definition für die *Reduziertheit von Basen eines Gitters* gegeben. Bevor wir die in unserer Arbeit relevanten Definition für die Reduziertheit einer Basis eines Gitters angeben, wird der Gram-Schmidtsche-Orthogonalisierungsprozeß beschrieben, um die notwendigen Sprechweisen zur Hand zu haben.

Der Gram-Schmidtsche-Orthogonalisierungsprozeß dient dazu, aus einer Basis

$$(\mathbf{b}_1, \ldots, \mathbf{b}_k) \in \mathbb{R}^{n\times k}$$

eines $\mathbb{R}$-Vektorraums $V = \text{span}(\mathbf{b}_1, \ldots, \mathbf{b}_k)$ der Dimension k, eine Basis $(\mathbf{b}_1^\star, \ldots, \mathbf{b}_k^\star)$ des Vektorraums V zu berechnen, welche die Eigenschaften $i)$ und $ii)$ hat.

i) $\langle \mathbf{b}_i^\star, \mathbf{b}_j^\star \rangle = 0$ falls $i \ne j$

ii) $\text{span}(\mathbf{b}_1, \ldots, \mathbf{b}_i) = \text{span}(\mathbf{b}_1^\star, \ldots, \mathbf{b}_i^\star)$ $(1 \le i \le k)$

Eine solche Basis kann durch die Definition 1.4 erhalten werden.

Definition 1.4 (Gram-Schmidt) *Seien* $\mathbf{b}_1, \ldots, \mathbf{b}_k \in \mathbb{R}^n$ *linear unabhängig. Die* Gram-Schmidt-Koeffizienten μ_{ij} *und* Gram-Schmidt-Vektoren $\mathbf{b}_j^\star$ *werden durch die folgenden Rekursionsgleichungen definiert:*

$$\mathbf{b}_i^\star = \mathbf{b}_i - \sum_{j=1}^{i-1} \mu_{ij}\mathbf{b}_j^\star \ (1 \le i \le k) \tag{1.3}$$

$$\mu_{ij} = \frac{\langle \mathbf{b}_i, \mathbf{b}_j^\star \rangle}{\| \mathbf{b}_j^\star \|^2} \ (1 \le j < i \le k). \tag{1.4}$$

Zu einer gegebenen Matrix

$$B = (\mathbf{b}_1, \ldots, \mathbf{b}_k) \in \mathbb{R}^{n\times k} \ (k \le n)$$

vom Spaltenrang k, heißt die Matrix

$$B^\star = (\mathbf{b}_1^\star, \ldots, \mathbf{b}_k^\star) \in \mathbb{R}^{n\times k} \ (k \le n)$$

die Gram-Schmidt-Matrix $B^\star$ *zur Matrix B.*

Die Tatsache, daß die Vektoren der Gram-Schmidt-Basis paarweise orthogonal sind, läßt sich natürlich unmittelbar nachrechnen. Der Prozeß ist aber auch anschaulich klar, da $\mathbf{b}_i^\star$ die Differenz von $\mathbf{b}_i$ und der orthogonalen Projektion von $\mathbf{b}_i$ in das orthogonale Komplement von $\text{span}(\mathbf{b}_1, \ldots, \mathbf{b}_{i-1})$ ist. Der Gram-Schmidt-Prozeß hängt empfindlich von der Anordnung der Vektoren $(\mathbf{b}_1, \ldots, \mathbf{b}_k)$ ab, aus denen die Gram-Schmidt-Matrix berechnet wird. Jede Permutation führt im allgemeinen zu einer anderen Gram-Schmidt-Matrix.

Aus den Längen der Gram-Schmidt-Vektoren läßt sich eine untere Schranke für das erste sukzessive Minimum ableiten.

Lemma 1.5 *Sei* $(\mathbf{b}_1, \ldots, \mathbf{b}_k) \in \mathbb{R}^{n \times k}$ *Basis des Gitters* L*,* λ_1 *das erste sukzessive Minimum von* L *und* $(\mathbf{b}_1^\star, \ldots, \mathbf{b}_k^\star)$ *die Gram-Schmidt-Matrix zu* B*. Dann ist die folgende Abschätzung erfüllt:*

$$\min\{||\mathbf{b}_i^\star|| : 1 \leq i \leq k\} \leq \lambda_1.$$

Als erste Definition für die Reduziertheit einer Basis eines Gitters, stellen wir den LLL-Reduktionsbegriff vor. Die Abkürzung LLL steht für die Initialen der Autoren H. W. Lenstra, A. K. Lenstra und L. Lovász, die einen Polynomzeitalgorithmus entwickelten, der diese Reduktionsbedingungen herstellt und der in Kapitel 2 näher erläutert wird. Die Definition geht auf Lovász zurück.

Definition 1.6 (LLL(y)-Reduziertheit) *Sei* $B = (\mathbf{b}_1, \ldots, \mathbf{b}_k) \in \mathbb{R}^{n \times k}$ *eine zulässige Matrix und sei* $y \in (1/4, 1]$*. Seien* μ_{kl} *die Gram-Schmidt-Koeffizienten und* $\mathbf{b}_l^\star$ *die Gram-Schmidt-Vektoren zur Matrix* B *für* $1 \leq l < k \leq n$*.*

Die Matrix B *heißt* LLL(y)-reduziert*, wenn die Bedingungen (1.5) und (1.6) erfüllt sind.*

$$|\mu_{kl}| \leq \frac{1}{2} \ (1 \leq l < k \leq n) \tag{1.5}$$

$$y \cdot \| \mathbf{b}_{k-1}^\star \|^2 \leq \| \mathbf{b}_k^\star + \mu_{k,k-1} \mathbf{b}_{k-1}^\star \|^2 \ (1 < k \leq n). \tag{1.6}$$

In [16] wurden für eine LLL(y)-reduzierte Matrix $B = (\mathbf{b}_1, \ldots, \mathbf{b}_n) \in \mathbb{Z}^{n \times n}$ die Ungleichungen

$$\left(\frac{4}{4\,y - 1}\right)^{1-i} \lambda_i \leq \| \mathbf{b}_i \|^2 \leq \left(\frac{4}{4\,y-1}\right)^{n-1} \lambda_i \quad (1 \leq i \leq n) \tag{1.7}$$

hergeleitet. Dabei sind die λ_i $(1 \leq i \leq n)$ die sukzessiven Minima des von B repräsentierten Gitters.

Die Ungleichungen legen die Bezeichungsweise *Güteparameter* für den Parameter y nahe. Je näher y an eins ist, um so besser werden die Minima des durch B repräsentierten Gitters durch die Spaltenvektoren der LLL(y)-reduzierten Matrix B approximiert.

Eine stärkere Defintion zur Reduziertheit von Basen eines Gitters wurde von Hermann Minkowski gegeben. Basen, die diesem Reduktionsbegriff genügen, weisen die "geringsten euklidischen Längen" auf.

Die vom euklidischen Skalarprodukt induzierte Norm ermöglicht eine *partielle Ordnung* der Vektoren des $\mathbb{R}^n$ wie unten definiert:

$$\mathbf{a} < \mathbf{b} \Leftrightarrow ||\mathbf{a}|| < ||\mathbf{b}||$$

für $\mathbf{a}, \mathbf{b} \in \mathbb{R}^n$.

Dies induziert eine teilweise Ordnung "$<$" aller Basen $(\mathbf{a}_1, \ldots, \mathbf{a}_k) \in \mathbb{R}^{n \times k}$, $(\mathbf{b}_1, \ldots, \mathbf{b}_k) \in \mathbb{R}^{n \times k}$ eines Gitters $L \subset \mathbb{R}^n$ durch

$$(\mathbf{a}_1, \ldots, \mathbf{a}_k) < (\mathbf{b}_1, \ldots, \mathbf{b}_k) \Leftrightarrow \exists j \in \{1, \ldots, k\} \forall i \in \{1, \ldots, j-1\} : ||\mathbf{b}_i|| = ||\mathbf{a}_i|| \text{ und } \mathbf{b}_j < \mathbf{a}_j.$$

Definition 1.7 (Minkowski-reduziert) *Eine Basis B eines Gitters L heißt* Minkowski-reduziert, *wenn B ein minimales Element der Menge aller Basen von L ist bezüglich der oben definierten partiellen Ordnung.*

Satz 1.8 (Minkowski-reduziert) *Eine Basis $B = (\mathbf{b}_1, \ldots, \mathbf{b}_k)$ eines Gitters L vom Rang k heißt* Minkowski-reduziert, *wenn $\mathbf{b}_i$ einer der kürzesten Vektoren ist, der mit $(\mathbf{b}_1, \ldots, \mathbf{b}_{i-1})$ zu einer Basis von L erweitert werden kann für $1 \leq i \leq k$.*

An dieser Stelle möchten wir darauf hinweisen, daß das Problem der Berechnung Minkowski-reduzierter Basen eines Gitters und die Berechnung der sukzessiven Minima eines Gitters nur bis zur Dimension vier äquivalente Probleme sind (Satz 5.11). Im allgemeinen sind die beiden Probleme verschieden. Darauf wird in Kapitel 3 gründlicher eingegangen.

1.1.5 Eingabegrößen für Reduktionsalgorithmen

Ein Problem der Reduktionstheorie für Gitter ist es, obere und untere Schranken für die Laufzeit von Algorithmen anzugeben, um eine Basis eines Gitters in eine Basis des gleichen Gitters zu transformieren, die einem bestimmten Reduktionsbegriff genügt. Dazu ist es erforderlich, eine Definition für die *Länge der Eingabe* eines Basisreduktionsalgorithmus anzugeben.

In dieser Arbeit werden häufig Basisreduktionsalgorithmen für Gitter von vollem Rang n studiert. Die Analyse von Basiseduktionsalgorithmen soll die Laufzeit in Abhängigkeit der Anzahl von Bits liefern, die erforderlich sind, um die Eingabe zu kodieren. Ist die Eingabe eines Algorithmus eine zulässige $n \times n$ Matrix $B = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ mit ganzzahligen Einträgen, so definiere

$$E(B) = \max\{\| \mathbf{b}_1 \|^2, \ldots, \| \mathbf{b}_n \|^2\}$$

die *Eingabegröße* eines Basisreduktionsalgorithmus. Für $\mathbf{u} = (u_1, \ldots, u_n) \in \mathbb{R}^n$ bezeichne $||\mathbf{u}||_\infty = \max\{|u_1|, \ldots, |u_n|\}$ die *Maximumsnorm* des Vektors $\mathbf{u}$.

Die Anzahl der Bits, die notwendig sind, um die Eingabe zu kodieren, wird als *Eingabelänge* bezeichnet. Die Eingabelänge ist beschränkt durch die Funktion

$$S(B) = n^2 \cdot \log(2 \cdot \max\{||\mathbf{b}_1||_\infty, \ldots, ||\mathbf{b}_n||_\infty\}).$$

Der Faktor zwei trägt den Vorzeichenbits Rechnung, mit dem Faktor n^2 werden alle Einträge der Matrix B berücksichtigt. Offenbar ist

$$n^{3/2} \log(2\, E(B)) \leq S(B) \leq n^2 \log(2\, E(B)).$$

Ist eine obere Schranke der Laufzeit eines Basisreduktionsalgorithmus in Abhängigkeit von der Eingabegröße gezeigt, so liegt auch eine obere Schranke in Abhängigkeit der Eingabelänge vor.

Ein Algorithmus heißt *schnell* oder *effizient*, wenn seine Laufzeit polynomiell in der Eingabelänge beschränkt ist, im anderen Fall heißt der Algorithmus *nicht effizient*.

1.2 Probleme und Resultate

Dieser Abschnitt gibt einen Überblick der Probleme aus dem Bereich der algorithmischen Reduktionstheorie für Gitter, die Gegenstand der vorliegenden Dissertation sind. Das Thema dieser Dissertation sind Gitterbasisreduktionsalgorithmen, die Bedingungen herstellen, die jeweils an den Extrempunkten der Skala stehen, die für Reduktionsvoraussetzungen existieren. Zum einen haben wir uns in Kapitel 3 mit offenen Problemen im Zusammenhang mit dem LLL-Algorithmus [16] befaßt. Dieser Algorithmus stellt eine relativ schwache Reduktionsbedingung her, die jedoch für viele Anwendungen ausreichend ist. Zum anderen haben wir in Kapitel 5 und Kapitel 6 Algorithmen entwickelt und analysiert, die die Minkowski-Reduziertheit einer Basis eines Gitters vom Rang drei herstellen. Diese Reduktionsbedingung ist sehr stark. Für beliebige Dimension sind keine effizienten Algorithmen bekannt.

1.2.1 Der LLL-Algorithmus mit Reduktionsgüte $y = 1$

Es stellt sich das folgende bis heute offene Problem:

Ist der LLL-Algorithmus für die Wahl des Güteparameters $y = 1$ effizient für Gitter vom Rang $k \geq 3$ für festes k?

Die Frage konnte nicht abschließend beantwortet werden. Stattdessen haben wir in Kapitel 3 das Konzept der *parametrisierten Eingaben* entwickelt, das es erlaubt, untere Schranken für die Anzahl der Iterationen des LLL-Algorithmus für $y = 1$ und beliebigen Rang $k \geq 3$ anzugeben. Für $k = 2$ wurde das Problem von Brigitte Vallée [37] studiert.

1.2.2 Qualität der Reduktionsgüte des LLL-Algorithmus

Die LLL(y)-Reduktionsbedingungen erlauben es, die Längen der reduzierten Basisvektoren und die sukzessiven Minima des zugrundeliegenden Gitters in Relation zu setzen (1.7). Eine naheliegende Frage ist es, ob die Schranken zu pessimistisch sind. In Abschnitt 2.2 werden Beispiele für Matrizen angegeben, die zwar LLL(y)-reduziert sind, jedoch Spaltenvektoren beinhalten, deren Längen der oberen Schranke aus (1.7) entsprechen. Damit wird diese Frage verneint.

1.2.3 TPR, ein Algorithmus zur Minkowski-Reduktion für $k = 3$

Die bekannten Algorithmen zur Minkowski-Reduktion [11, 29] basieren auf der Verwendung des LLL-Algorithmus und Enumeration. In Abschnitt 4.2 wird gezeigt, daß der LLL-Algorithmus in einigen Fällen die Reduktionsqualität verringern kann. Mit Beispiel 4.2 wird verdeutlicht, daß es de facto bereits für Gitter vom Rang $k = 3$ Basen gibt, deren euklidische Längen durch den LLL-Algorithmus vergrößert werden. Paar-Reduktionsalgorithmen haben diesen Nachteil nicht. Pohst und Zassenhaus [26] schlugen Reduktionsalgorithmen vor, die immer erst alle Paare, dann alle Tripel und so fort Minkowski-reduzieren. Ausgehend von dieser Idee wird in Kapitel 5 der TPR-Algorithmus entwickelt. Für $k = 3$ wird in Satz 5.14 bewiesen, daß nach Minkowski-Reduktion aller Paare nur noch ein einfacher algorithmischer

Schritt erforderlich ist, um eine Minkowski-reduzierte Basis zu erhalten. Da zunächst alle Paare reduziert werden, wird der Algorithmus "**t**otaler-**P**aar-**R**eduktionsalgorithmus" genannt.

Die Anzahl der Iterationen des TPR-Algorithmus erweist sich als linear, die Bitkomplexität ist kubisch in der Eingabelänge beschränkt. Dies wird in Abschnitt 5.2 bewiesen. Durch parametrisierte Eingaben werden in den Abschnitten 5.3 und 5.8 auch untere Schranken für die Anzahl der Iterationen und die Bitkomplexität bewiesen.

1.2.4 FTPR, ein schneller TPR-Algorithmus

Der in Kapitel 6 vorgestellte FTPR-Algorithmus, **F**ast TPR, stellt eine Verbesserung des TPR-Algorithmus dar. In den Sätzen 6.4 und 6.6 wird gezeigt, daß die Anzahl der Interation wie im TPR-Algorithmus linear, die Bitkomplexität jedoch quadratisch in der Eingabelänge beschränkt ist. Damit ist die asymptotische Bitkomplexität dieses Algorithmus wenigstens eine Potenz geringer als die Bitkomplexität bekannter Algorithmen [11, 20, 35].

1.2.5 Sind Paar-Reduktionsalgorithmen für $k \geq 4$ sinnvoll?

Am Ende der Arbeit, in Kapitel 7, wird abschließend die Frage studiert, ob das von Pohst und Zassenhaus eingeführte Paar-Reduktionskonzept für beliebe Dimension sinvoll ist. Die Frage konnte verneint werden, denn folgende Kritikpunkte konnten gefunden werden.

i) total-Paar-reduzierte zulässige (tpr) Matrizen stellen beliebig schlechte Approximationen der sukzessiven Minima des assoziierten Gitters dar

ii) es ist nicht klar, ob es überhaupt Paar-Reduktionsalgorithmen gibt, die tpr-Matrizen effizient berechnen können

Für den Kritikpunkt i) wird in Abschnitt 7.1 eine Anregung von Hendrik Lenstra [17] ausgearbeitet.

Für den Kritikpunkt ii) wird in Abschnitt 7.2 ein naheliegender Paar-Reduktionsalgorithmus konzipiert und eine parametrisierte Eingabe für diesen angegeben, von der wir begründet annehmen können, daß sie zu einer exponentiellen Anzahl von Iterationen führt.

1.2.6 Einordnung der Resultate

In Tabelle 1.2 geben wir einen Überblick über Resultate diverser AutorInnen zu Basisreduktionsalgorithmen zur Minkowski-Reduktion gegeben. Die betrachteten Algorithmen erhalten als Eingabe eine quadratische Matrix $A \in \mathbb{Z}^{k \times k} (k = 3$ bzw. $k \geq 2)$ mit $E(A) \leq 2^b$ $(b \geq 6)$. Es bezeichne I die Anzahl von Iterationen und K die Anzahl von Bitoperationen, die die Algorithmen bei Eingabe von A ausführen, um eine Matrix B zu berechnen, die eine Minkowski-reduzierte Basis von $L(A)$ ist. Für die elementaren Operationen werden klassische Algorithmen verwendet, also keine "schnellen" Multiplikationstechniken [31].

Reduktionsalgorithmen zur Minkowski-Reduktion für Gitter vom Rang $k = 3$			
AutorIn	Name des Algorithmus	Komplexitätsresultate	
Lagarias [20]	**MGTFR**[1]	$I = O(b)$	$K = O(b^4)$
Vallée [35]	**Acute**	$I = O(b)$	$K = O(b^3)$
van Sprang	**TPR** **FTPR**	$I \leq 1654 \cdot b$ $I = O(b)$	$K \leq 1654 \cdot b^3$ $K = O(b^2)$
Reduktionsalgorithmen zur Minkowski-Reduktion für Gitter von beliebigem Rang k			
Helfrich [11]	**M-RED**	$I = O\left((5/4)^{k^3(4-o(1))} \cdot k \cdot b\right)$	$K = O\left((5/4)^{k^3(4-o(1))} \cdot k \cdot b^3\right)$

Tabelle 1.2: Komplexitätsresultate von Basisreduktionsalgorithmen für Gitter

1.3 Methoden

Um untere Schranken für die Anzahl der Iterationen des LLL-Algorithmus bei Wahl des Güteparameters $y = 1$ zu finden, wurden in Abschnitt 3.3 durch Experimentieren und Überlegen Beispiele für Eingaben gefunden, die sich als Produkt zweier zulässiger Matrizen A und $G^s (0 \leq s \in \mathbb{Z})$ schreiben ließen, bei deren Eingabe der LLL-Algorithmus periodisches Verhalten zeigt. Das heißt, eine Folge von Anweisungen wiederholte sich in der gleichen Weise. Dabei kann sowohl die Anzahl der periodischen Abläufe, die der Algorithmus ausführt, als auch die Eingabelänge über den Exponenten s gesteuert werden. Diese Methode, die wir das "Konzept der parametrisierten Eingaben" genannt haben, wird auch für die Bestimmung unterer Schranken für die Anzahl der Iterationen der anderen Gitterbasisreduktionsalgorithmen dieser Arbeit verwendet.

Um zu beweisen, daß der LLL-Algorithmus das genannte periodische Verhalten für die angegebene parametrisierte Eingaben aufweist, werden in Lemma 3.3 Rekursionsrelationen für einen Satz relevanter Größen des Algorithmus aufgestellt und gelöst . Die Suche nach einem Satz relevanter Größen des Algorithmus, die eine analytische Behandlung erlauben, erwies sich als kompliziertes Problem.

Die Entwicklung des TPR-Algorithmus basiert auf der Beobachtung, daß ein einfacher algorithmischer Schritt zur Minkowski-Reduktion einer Basis eines Gitters vom Rang drei genügt, wenn bereits alle Paare Minkowski-reduziert sind. Die Eigenschaft, daß alle Paare Minkowski-reduziert sind, wird durch wiederholte Hintereinanderausführung dreier Gauß-Algorithmen zur Reduktion von Rang zwei Gittern, hergestellt. Die bekannten Analysemethoden des Gauß-Algorithmus [37] basieren auf der Tatsache, daß dieser Algorithmus in jeder außer der letzten Iteration eine signifikante Längenverkürzung der Basisvektoren erzielt. Die Situation, daß ein Gauß-Algorithmus immer nur solche Schritte ausführt, bei denen keine Aussage über Längenverkürzung möglich ist, diese nennen wir *kritische Fälle*, kann im TPR-Algorithmus permanent auftreten, wie ausgearbeitete Beispiele in Abschnitt 5.8 dieser Arbeit belegen. Eine naive Untersuchung aller denkbaren kritischen Fälle führt zu einer kombinatorischen Explosion, die nicht zu handhaben ist. Durch die Bereitstellung umfangreicher Sätze in den

[1] **m**odified **G**auss **t**ernary **f**orm **r**eduction **a**lgorithm

Abschnitten 5.6 und 5.7, die gewisse kritische Fälle als inkonsistent ausklammern, ist es gelungen, die Anzahl der zu betrachtenden kritischen Fälle auf wenige tausend zu reduzieren. In Abschnitt 5.7 wird skizziert, wie ein Programm für das Computeralgebrasystem PARI zur Abhandlung dieser Fälle entwickelt wurde, welches alle kritischen Fälle generiert, die nach den Konsistenzregeln noch auftreten können. Für jeden Fall wird eine Reduktion der Hadamardschranke nach jeweils 22 Gaußschritten auf weniger als 15% ihres ursprünglichen Wertes bewiesen.

Die Tatsache, daß Paar-Reduktionsalgorithmen zur Reduktion von Gitter beliebigen Rangs nicht sinnvoll sind, wird durch Beispiele wird in den Abschnitten 7.1 und 7.2 belegt.

Kapitel 2

LLL - ein approximativer Reduktionsalgorithmus

In diesem Kapitel befassen wir uns mit dem sogenannten LLL-Algorithmus. Der Algorithmus berechnet aus einer Basis eines Gitters L eine neue Basis von L, die die Minima des Gitters L approximiert. Nach einer Vorstellung und kurzen Charakterisierung des LLL-Algorithmus, in der zusammengefaßt wird, was der LLL-Algorithmus leistet, werden Beispiele für Eingaben vorgelegt, bei denen der Algorithmus einen sehr geringen Reduktionserfolg erbringt. Dabei handelt es sich um zulässige Matrizen, bei denen der LLL-Algorithmus sehr schlechte Approximationen der Minima des durch diese Matrizen repräsentierten Gitter berechnet. Danach wird das offene Problem untersucht, ob der Algorithmus polynomiell in der Eingabelänge und der Dimension ist, wenn für die Ausgabe die bestmöglichen Approximationen der Minima des Gitters verlangt werden, also $y = 1$ gewählt wird.

2.1 Der LLL-Algorithmus

Es folgt eine kurze Zusammenfassung des LLL-Algorithmus, Ein- und Ausgabe des Algorithmus werden spezifiziert und eine Darstellung des LLL-Algorithmus durch Struktogramme [24] wird gegeben. Außerdem wird das bekannte Komplexitätsresultat [16], das eine obere Schranke für die Anzahl der arithmetischen Operationen in Abhängigkeit von Eingabelänge und Dimension angibt zitiert. Ferner beweisen wir eine obere Schranke für die Anzahl der Ausführungen einer für den Algorithmus zentralen Anweisung in Abhängigkeit von Eingabelänge und Dimension. Durch diese Renormierung wird eine direkte Gegenüberstellung des bekannten Komplexitätsresultats [16] mit unserem Lösungsbeitrag zum offenen Problem erleichtert.

Im folgenden bezeichne n eine natürliche Zahl. Der LLL-Algorithmus erhält als Eingabe folgende Objekte:

i) Eine Matrix $A = (\mathbf{a}_1, \ldots, \mathbf{a}_n) \in GL(n, \mathbb{Q})$ mit $\mathbf{a}_i \in \mathbb{Z}^n, 1 \leq i \leq n$, die eine Darstellung des Gitters $L = \{v \in \mathbb{Z}^n : v = \sum_{i=1}^n x_i \mathbf{a}_i, x_i \in \mathbb{Z}, 1 \leq i \leq n\} = L(A)$ ist. Mit dieser Eingabematrix erhält der Algorithmus implizit auch den Rang des Gitters, dessen Basis reduziert werden soll.

ii) Einen *Güteparameter* $y = y_1/y_2$, $y_1 \in \mathbb{N}$, $y_2 \in \mathbb{N}$, $\mathrm{ggT}(y_1, y_2) = 1$, mit den Eigenschaften $1/4 < y \leq 1$ und $y_2 < \prod_{i=1}^{n} ||\mathbf{a}_i||^{2\,i}$.

In der ursprünglichen Formulierung in [16] wurde der Güteparameter y nicht als Eingabe aufgefaßt, sondern immer auf den festen Wert 3/4 gesetzt. In einer Untersuchungen, die den Güteparameter als Eingabe zuläßt, ist die Beschränkung der Größe des Nenners unbedingt erforderlich. Der Grund ist, daß größere Werte nicht sinnvoll sind, und überdies zu einem schlechteren als dem bekannten Komplexitätsresultat führen.

Die Abschätzungen (1.7) rechtfertigen die Bezeichnungsweise "approximativen" Basisreduktionsalgorithmus für den LLL-Algorithmus. Im Gegensatz zu den Algorithmen, die Minkowski-reduzierte Gitterbasen ausgeben, berechnet der LLL-Algorithmus nur Approximationen an die Minima, deren euklidische Längen noch exponentiell weit von den Längen der Minima entfernt sein können.

Um optimale Übersichtlichkeit zu gewährleisten wurde der LLL-Algorithmus in Unteralgorithmen und damit Unterstruktogramme gegliedert. Die Gliederung ist in untenstehender Tabelle zusammengefaßt worden.

Name	**Algorithmus**	**Beschreibung**
LLL	2.1	der komplette LLL-Algorithmus
init-LLL	2.2	Initialisierung, Gram-Schmidtsche-Orthogonalisierung
Gaußstep	2.3	($\star$) genannter Teil der Originalarbeit, [16]
swap-update	2.4	dies wird mit (2) in der Originalarbeit bezeichnet, [16]

Die folgenden Ausführungen haben das Ziel, die Funktionsweise des LLL-Algorithmus darzulegen. Zunächst wird erklärt, wie die auch im Fortgang der Arbeit verwendeten Struktogramme zur Darstellung von Algorithmen zu lesen sind. Danach wird erläutert wie der LLL-Algorithmus funktioniert.

In Struktogramm 2.1 ist der LLL-Algorithmus angegeben. Hier, wie im Struktogramm der Initialisierungsroutine, steht zu Beginn des Struktogramms, was die Eingabe ist und welche Ausgabe daraus durch den im darunterstehenden Struktogramm dargestellten Algorithmus berechnet wird. Dies hat den Sinn, den Algorithmus schon alleine mit diesen Informationen verständlich zu machen. Bei den anderen Hilfsroutinen Gaußstep und swap-update wurde auf diese Darstellungselemente verzichtet, da diese Hilfsroutinen programmiertechnisch gesprochen nur globale Variablen modifizieren. Die Eingabe, die aus den Indizes (k, l) für die Hilfsroutine Gaußstep und aus dem Index k für die Hilfsroutine swap-update besteht, teilt diesen Hilfsroutinen mit, welcher Satz globaler Variablen zu modifizieren ist.

Algorithmus 2.1 (LLL(A, y))

Eingabe: Eine Matrix $A \in GL(n, \mathbb{Q})$ mit ganzzahligen Einträgen und ein Güteparameter $y = y_1/y_2 \in \mathbb{Q} \in (1/4, 1)$ mit $y_1 \in \mathbb{N}$ und $y_2 \in \mathbb{N}$ wie oben beschrieben.
Ausgabe: Eine Matrix B, die Basis des Gitters $L(A)$ und LLL(y) reduziert ist.

init-LLL(A)				(1)
$k = 2$				(2)
WHILE $k \neq n + 1$				(3)
	$l = k - 1$			(4)
	Gaußstep(k, l)			(5)
	IF $c_k < (y - \mu_{k,k-1}^2)c_{k-1}$			(6)
	THEN	swap-update(k)		(7)
	ELSE	FOR $l = k - 2$ downto 1		(8)
			Gaußstep(k, l)	(9)
		$k = k + 1$		(10)

Die Anweisung (1) des LLL-Algorithmus ruft die Initialisierungsroutine auf, deren Struktogramm unten angegeben ist.

Algorithmus 2.2 (init-LLL(A))

Eingabe: Eine Matrix $A \in GL(n, \mathbb{Q})$ mit ganzzahligen Einträgen.
Ausgabe: Die Gram-Schmidt-Koeffizienten und die quadrierten euklidischen Längen der Gram-Schmidt-Vektoren zur Matrix A.

FOR i=1 to n			(1)
	$\mathbf{a}_i^\star = \mathbf{a}_i$		(2)
	FOR j=1 to i-1		(3)
		$\mu_{ij} = \frac{\langle \mathbf{a}_i, \mathbf{a}_j^\star \rangle}{c_j}$	(4)
		$\mathbf{a}_i^\star = \mathbf{a}_i^\star - \mu_{ij}\mathbf{a}_j^\star$	(5)
	$c_i = \parallel \mathbf{a}_i^\star \parallel^2$		(6)

Der Intialisierungalgorithmus 2.2 erhält als Eingabe die reguläre Matrix $A = (\mathbf{a}_1, \ldots, \mathbf{a}_n) \in \mathbb{Z}^{n \times n}$, die der LLL-Algorithmus zuvor als Eingabe erhalten hat. Die Initialisierung besteht aus zwei FOR-Schleifen, die ineinander geschachtelt sind. Nach Abarbeitung der Anweisungen (2) bis (6) der Initialisierung wird der Index i inkrementiert und die Anweisungen (2) bis (6) wiederholt. Dies geschieht so lange, bis die Folge der Anweisugen (2) bis (6) mit dem Index $i = n$ durchlaufen wird. Die innere Schleife mit Laufindex j arbeitet in anloger Weise. Ist $i = 1$, so werden die Anweisungen (4) und (5) nicht ausgeführt und nach Anweisung (2) folgt unmittelbar die Anweisung (6). Insgesamt wird die Schleife, die aus den Anweisungen (4) und (5) besteht zu festem i genau $i-1$ mal durchlaufen. Der ganze Initialisierungsprozeß ist eine algorithmische Abbildung der Rekursionsformeln zur Berechnung der Gram-Schmidt-Koeffizienten und der Gram-Schmidt-Vektoren. Die Initialisierung ist auch so zu verstehen, daß Speicherplatz für $n(n-1)/2$ rationale Zahlen $\mu_{ij}\,(1 \leq j < i \leq n)$ und für n rationale Zahlen $c_i\,(1 \leq i \leq n)$ geschaffen wird, die nach der Initialisierung allen Hilfsroutinen des LLL-Algorithmus als globale Variablen zur Verfügung stehen. Die Gram-Schmidt-Vektoren werden nach der Initialisierung nicht mehr benötigt und fungieren in der Initialisierung nur als lokale Variablen.

Nach Abschluß der Initialisierung wird Anweisung (2) des LLL-Algorithmus ausgeführt. Hier wird der für den Ablauf des LLL-Algorithmus maßgebliche Index k, den wir auch *Masterindex* nennen, auf den Wert zwei gesetzt. Dann wird die Folge der Anweisungen(4) bis (10) des LLL-Algorithmus wie unten beschrieben iteriert. Der LLL-Algorithmus terminiert, wenn der Masterindex den Wert $n+1$, nach Abarbeitung von (10), erreicht hat.

Das einzige Strukturelement, das noch erklärt werden muß, ist das IF-THEN-ELSE Konstrukt. Ist die Bedingung erfüllt, wird (7) ausgeführt und die Bearbeitung in (4) fortgesetzt, da dann in jedem Fall $k \neq n+1$ gilt. Ansonsten werden die Anweisungen (8) und (9) $(k-2)$-mal ausgeführt. Hat k nach der Inkrementierung in (10) den Wert $n+1$, so terminiert der LLL-Algorithmus.

Nach dieser Beschreibung folgt eine kurze Erklärung des Reduktionsalgorithmus. Der LLL-Algorithmus führt zwei Typen von Operationen auf den Spaltenvektoten der Eingabematrix aus.

i) "Reduktion": In Gaußstep wird der Vektor $\mathbf{a}_k$ durch den Vektor $\mathbf{a}_k - \lfloor \mu_{kl} \rceil \mathbf{a}_l$ ersetzt.

ii) Vertauschung: Die Vektoren $\mathbf{a}_{k-1}$ und $\mathbf{a}_k$ werden vertauscht, wenn die Bedingung in (7) erfüllt ist.

Die maßgeblichen Größen im LLL-Algorithmus sind die Gram-Schmidt-Koeffizienten μ_{kl} und die quadrierten euklidischen Längen der Gram-Schmidt-Vektoren c_i, zur aktuell berechneten Matrix $(\mathbf{a}_1, \ldots, \mathbf{a}_n)$. Diese steuern durch Anweisung (6) den Ablauf des Algorithmus. Operationen vom Typ i) haben *keinen* Einfluß auf die c_i, erfordern jedoch eine Neuberechnung von μ_{kj} für $1 \leq j \leq l$. Dies geschieht in Anweisung (5) und (6) des Algorithmus Gaußstep. Nach Abarbeitung von Anweisung (6) in Gaußstep gilt offenbar $|\mu_{kl}| \leq 1/2$. Also gilt nach Reduktion der Zeile $(\mu_{k,1}, \ldots, \mu_{k,k-1})$ der Gram-Schmidt-Koeffizientenmatrix $|\mu_{kj}| \leq 1/2$ für $1 \leq j < k$. Der Masterindex k wird erst inkrementiert, wenn diese Bedingungen gelten. Die Ausführung von Operationen vom Typ i) für größere k hat keinen Einfluß auf Zeilen der Gram-Schmidt-Koeffizientenmatrix zu Indizes kleiner als k. Für $(k,l) = (2,1)$ hat die Ausführung von Opertion i) den Effekt der Verkürzung der euklidischen Länge des neuen Vektors $\mathbf{a}_k$ gegenüber des alten Vektors. In jedem anderen Fall kann unter ungünstigen Umständen sogar eine Verlängerung eintreten. Die Reduktion einer kompletten Zeile der Gram-Schmidt-Matrix zum Index k hat jedoch im allgemeinen auch eine Verkürzung zur Folge. Denn bezeichne $\mu_{kj}\,(1 \leq j < k)$ die Gram-Schmidt-Koeffizienten vor und $\nu_{kj}\,(1 \leq j < k)$ diese nach der Reduktion, sowie $\mathbf{a}_k$ den Vektor vor und $\mathbf{b}_k$ den Vektor nach der Reduktion einer ganzen Zeile der Gram-Schmidt-Koeffizientenmatrix, so gilt:

$$\| \mathbf{a}_k \|^2 = c_k + \sum_{j=1}^{k-1} \mu_{kj}^2 c_j$$

und

$$\| \mathbf{b}_k \|^2 = c_k + \sum_{j=1}^{k-1} \nu_{kj}^2 c_j.$$

Außerdem gilt $|\nu_{kj}| \leq 1/2\,(1 \leq j < k)$. Eine solche Reduktion einer kompletten Zeile tritt immer genau vor der Inkrementierung des Masterindex auf.

Operationen vom Typ ii) haben Korrekturen sowohl von Gram-Schmidt-Koeffizienten als auch der quadrierten euklidischen Längen der Gram-Schmidt-Vektoren zur Folge. Diese werden in der Hilfsroutine vorgenommen, deren Struktogramm in Algorithmus 2.4 präsentiert wird. Der dort verwendete Befehl Swap() vertauscht Speicherinhalte. Dabei ist es egal, ob es sich um Vektoren oder Skalare handelt. Ist $k > 2$, so wird k dekrementiert und die beschriebenen Vorgänge wiederholen sich. Ist $k = 2$, so wird ohne vorhergehendes Dekrementieren des Masterindex wiederholt. Es folgen noch die Struktoramme 2.3 und 2.4 der Anweisungen (5) und (7) des LLL-Algorithmus.

Algorithmus 2.3 (Gaußstep(k, l))

<table>
<tr><td colspan="3">IF $|\mu_{kl}| > \frac{1}{2}$</td><td>(1)</td></tr>
<tr><td rowspan="5">THEN</td><td colspan="2">$r = \lfloor \mu_{kl} \rceil$</td><td>(2)</td></tr>
<tr><td colspan="2">$\mathbf{a}_k = \mathbf{a}_k - r\mathbf{a}_l$</td><td>(3)</td></tr>
<tr><td colspan="2">FOR $j = 1$ to $l - 1$</td><td>(4)</td></tr>
<tr><td></td><td>$\mu_{kj} = \mu_{kj} - r\mu_{lj}$</td><td>(5)</td></tr>
<tr><td colspan="2">$\mu_{kl} = \mu_{kl} - r$</td><td>(6)</td></tr>
</table>

Algorithmus 2.4 (swap-update(k))

$\mu = \mu_{k,k-1}$		
$\mathbf{c} = c_k + \mu^2 c_{k-1}$		
$\mu_{k,k-1} = \frac{\mu\, c_{k-1}}{c}$		
$c_k = \frac{c_{k-1}\, c_k}{c}$		
$c_{k-1} = c$		
Swap($\mathbf{a}_{k-1}, \mathbf{a}_k$)		
FOR $j = 1$ to $k-2$		
	Swap($\mu_{k-1,j}, \mu_{kj}$)	
FOR $i = k+1$ to n		
	$z_1 = \mu_{k,k-1}\,(\mu_{i,k-1} - \mu_{ik}\,\mu) + \mu_{ik}$	
	$z_2 = \mu_{i,k-1} - \mu\,\mu_{ik}$	
	$\mu_{i,k-1} = z_1$	
	$\mu_{ik} = z_2$	
IF $k > 2$		
THEN	$k = k-1$	

Zum Ende der Erläuterungen sei noch erwähnt, daß der LLL-Algorithmus einen sehr alten Algorithmus, der auf Carl Friedrich Gauß zurückgeht, verallgemeinert [9]. Dieser sogenannte *Gaußalgorithmus zur Basisreduktion* von Gittern L vom Rang zwei berechnet die Minima von L. Dieser Algorithmus ist im wesentlichen der LLL(A, 1)-Algorithmus mit $A \in GL(2, \mathbb{Q})$ und ganzzahligen Einträgen. Der Gaußalgorithmus gibt bei Eingabe einer zulässigen 2×2 Matrix A mit ganzzahligen Einträgen eine Matrix $B = (\mathbf{b}_1, \mathbf{b}_2) \in GL(2, \mathbb{Q})$ aus, die Basis von $L(A)$ ist. Diese erfüllt die Eigenschaften:

$$\left| \frac{\langle \mathbf{b}_1, \mathbf{b}_2 \rangle}{\| \mathbf{b}_1 \|^2} \right| \leq \frac{1}{2} \tag{2.1}$$

und

$$\| \mathbf{b}_1 \|^2 \leq \| \mathbf{b}_2 \|^2. \tag{2.2}$$

Eine Abschwächung von Bedingung (2.2) lautet

$$y \cdot \| \mathbf{b}_1 \|^2 \leq \| \mathbf{b}_2 \|^2, \quad 1/4 < y \leq 1. \tag{2.3}$$

Die Bedingungn (2.1) und (2.3) sind gerade die LLL(y)- Reduktionsbedingungen für Gitter vom Rang zwei. Diese gehen auf Carl Friedrich Gauß zurück.

Definition 2.5 *Zwei linear unabhängige Vektoren* $(\mathbf{b}_1, \mathbf{b}_2) \in \mathbb{R}^{n \times 2}$ $(n \geq 2)$ *heißen* y-Gauß-reduziert, *wenn sie die Bedingungen (2.1) und (2.3) erfüllen.*

Eine Matrix $(\mathbf{b}_1, \ldots, \mathbf{b}_n) \in GL(n, \mathbb{R})$ ist also genau dann LLL(y)-reduziert, wenn für jedes der $(n-1)$-Paare $\beta_i = (\mathbf{b}_i, \mathbf{b}_{i+1})\,(1 \le i < n)$ gilt: Die orthogonale Projektion der Spaltenvektoren von $\beta_i\,(1 \le i < n)$ auf das orthogonale Komplement von $\mathrm{Span}(\mathbf{b}_{i-1}, \ldots, \mathbf{b}_1)$ ist y-Gauß-reduziert.

Zum Ende des Abschnitts sei nun noch das bekannte Komplexitätsresultat aus [16] vorgestellt.

Satz 2.6 *Der LLL-Algorithmus berechnet, bei Eingabe einer regulären Matrix $A \in \mathbb{Z}^{n\times n}$ und eines Güteparameters y mit $1/4 < y < 1$, in $O\left(n^4 \log E(A)/\log(1/y)\right)$ arithmetischen Operationen auf Zahlen der binären Länge $O(n \log E(A))$ eine LLL(y)-reduzierte Matrix, die Basis des Gitters $L(A)$ ist.*

In Kapitel 3 wird für den Fall, daß auch $y = 1$ als Güteparameter gestattet ist, untere Schranken für die Komplexität des LLL-Algorithmus ableiten. Diese unteren Schranken werden sich auf ein anderes Komplexitätsmaß als die Anzahl der arithmetischen Operationen beziehen, um größtmögliche Übersichtlichkeit bei den komplizierten Zusammenhängen zu behalten. Dieses Komplexitätsmaß wird folgendermaßen definiert.

Definition 2.7 *Sei $A \in \mathbb{Z}^{n\times n}$ eine reguläre Matrix und $y \in (1/4, 1]$ ein Güteparameter. Mit $T(A, y)$ bezeichnen wir die Anzahl der Aufrufe des Gaußstep - Unteralgorithmus, die der LLL-Algorithmus bei Eingabe von A und y ausführt.*

Eine obere Schranke für den LLL-Algorithmus bezüglich dieses Komplexitätsmaßes lautet:

Korollar 2.8 *Sei $A \in \mathbb{Z}^{n\times n}$ eine reguläre Matrix und sei $y \in (1/4, 1)$, dann gilt:*

$$T(A, y) = O\left(n^2 \log E(A)/\log(1/y)\right).$$

Beweis: Der Beweis verläuft analog zum Beweis von Satz 2.6. □

2.2 Beispiele für geringen Reduktionserfolg des LLL-Algorithmus

In diesem Abschnitt wird die Fragestellung untersucht, ob es Matrizen gibt, die LLL(y)-reduziert sind, aber schlechte Approximationen an die Minima des von ihnen repräsentierten Gitters darstellen. Die Frage wird konstruktiv beantwortet.

Für die Konstruktion von Beispielen für Matrizen $B \in \mathbb{R}^{n\times n}$, $n \in \mathbb{N}^{>1}$, die LLL(y)-reduziert sind, aber schlechte Approximationen an die Minima des von B repräsentierten Gitters darstellen, untersuchen wir Matrizen $B \in GL(n, \mathbb{R})$, die obere Dreiecksmatrizen mit nichtverschwindenden Diagonalelementen sind.

Lemma 2.9 *Es sei $n \in \mathbb{N}^{>2}$ und*

$$B = \begin{pmatrix} b_{11} & b_{12} & \cdots & b_{1n} \\ 0 & b_{22} & \cdots & b_{2n} \\ \mathbf{0} & 0 & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & 0 & b_{nn} \end{pmatrix} \in GL(n, \mathbb{R}).$$

Dann ist $B^\star = \mathrm{diag}(b_{11},\ldots,b_{nn})$ *die zu* B *gehörende Gram-Schmidt-Basis und die Gram-Schmidt-Koeffizienten zu* B *sind* $\mu_{ij} = b_{ji}/b_{jj}$ $(1 \le j < i \le n)$.

Beweis: Die Behauptung $B^\star = \mathrm{diag}(b_{11},\ldots,b_{nn})$ wird für beliebiges aber festes n durch Induktion über dem Spaltenindex gezeigt. Daraus folgt dann unmittelbar die Formel für die Gram-Schmidt-Koeffizienten. □

Als Korollar stellen wir die zwei Bedingungen zusammen, wann eine Matrix, wie in Lemma 2.9, LLL(y)-reduziert ist.

Korollar 2.10 *Eine Matrix* B*, die wie in Lemma 2.9 definiert ist, ist genau dann LLL*(y)*-reduziert* $(1/4 \le y \le 1)$*, wenn folgende Bedingungen gelten:*

i) $|b_{ij}/b_{ii}| \le 1/2$ $(1 \le i < j \le n)$

ii) $b_{ii}^2 + b_{i-1,i}^2 \ge y \cdot b_{i-1,i-1}^2$ $(2 \le i \le n)$.

Beweis: Die Bedingungen unter i) und ii) ergeben sich aus den Bedingungen (1.5) und (1.6) der LLL(y)-Reduktionsbedingungen, unter Verwendung der Ergebnisse von Lemma 2.9. □

Korollar 2.11 *Mit jeder Matrix* B*, die wie in Lemma 2.9 definiert ist und die LLL*(y)*-reduziert ist, ist auch jede Matrix LLL*(y)*-reduziert, in der die Elemente der Matrix* B *oberhalb der oberen Nebendiagonalen durch Werte ersetzt wurden, die im Absolutbetrag kleiner sind.*

Beweis: Korollar 2.10. □

Das Korollar 2.11 wird nun benutzt, um LLL(y)-reduzierte Matrizen anzugeben, die schlechte Approximationen der Minima der von diesen Matrizen repräsentierten Gitter sind.

Satz 2.12 *Sei* $1/4 < y \le 1$ *und die Matrix* $B = (b_{ij})_{(1 \le i,j \le n)} \in \mathbb{R}^{n \times n}$, $n \in \mathbb{N}^{>2}$ *sei wie folgt definiert:*

$$b_{ij} = \begin{cases} (y - 1/4)^{(i-1)/2} & \textit{für } i = j \ (1 \le i \le n) \\ b_{ii}/2 & \textit{für } 1 \le i < j \le n \\ 0 & \textit{sonst} . \end{cases}$$

Ferner sei $\mathbf{b}_1$ *der erste Spaltenvektor der Matrix* B*. Dann gelten folgende Aussagen:*

i) Die Matrix B *ist LLL*(y)*-reduziert.*

ii) In dem Gitter, das von den Spaltenvektoren der Matrix B *aufgespannt wird, existiert ein Vektor* $s \ne 0$ *mit* $\| \mathbf{b}_1 \|^2 \ge (4/(4y-1))^{n-2} \cdot \| s \|^2$.

Beweis: Zunächst wird die Behauptung i) des Satzes verifiziert. Die Matrix B erfüllt die Prämissen von Korollar 2.10. Es gilt $b_{ij}/b_{ii} = 1/2$ $(1 \le i < j \le n)$. Damit ist die Bedingung i) von Korollar 2.10 erfüllt. Die Gültigkeit der Bedingung ii) von Korollar 2.10 für die Matrix B ergibt sich umittelbar, beide Bedingungen des Korollars 2.10 für die Matrix B sind also erfüllt und daher ist die Matrix LLL(y)-reduziert. Damit ist die Behauptung i) des Satzes bewiesen.

Um die Behauptung ii) zu beweisen, wird der Vektor $s = (s_1, s_2, \ldots, s_n)$ mit $s_i = b_{i,n} - b_{i,n-1}$ $(1 \leq i \leq n)$ und der Skalar f als positive Quadratwurzel $f = \sqrt{y - 1/4}$ definiert. Offenbar gilt $s_i = 0$ für $1 \leq i \leq n-2$ sowie $s_{n-1} = -f^{n-2}/2$ und $s_n = f^{n-1}$. Also ist

$$\| s \|^2 = f^{2(n-1)} + \frac{f^{2(n-2)}}{4} = f^{2(n-2)}(1/4 + f^2) = f^{2(n-2)} \cdot y.$$

Da $\| \mathbf{b}_1 \|^2 = 1$ gilt, folgt unmittelbar

$$\| \mathbf{b}_1 \|^2 = \| s \|^2 \cdot \| s \|^{-2} = \| s \|^2 (4/(4y-1))^{2(n-2)} \cdot 1/y \geq (4/(4y-1))^{2(n-2)} \cdot \| s \|^2,$$

womit auch die Behauptung ii) des Satzes bewiesen ist. □

Ein Vergleich mit der Schranke aus den Ungleichungen (1.7) zeigt, daß die Matrix aus Satz 2.12 in der Tat ein Beispiel für äußerst schlechte Minimumsapproximationen durch LLL(y)-reduzierte Matrizen ist.

Zum Abschluß des Abschnitts geben wir noch ein Beispiel für eine zulässige Matrix an, die sogar LLL(1)-reduziert ist, jedoch eine sehr schlechte Approximation des ersten sukzessiven Minimums darstellt. Das Beispiel lautet:

$$B = \begin{pmatrix} 18 & 9 & 9 \\ 0 & 16 & 8 \\ 0 & 0 & 14 \end{pmatrix} \tag{2.4}$$

Es ist: $\frac{\lambda_1^2}{||b_1||^2} = \frac{8^2+14^2}{18^2} = \frac{5 \cdot 13}{3^4}$, was $\| \mathbf{b}_1 \|^2 = \frac{3^4}{5 \cdot 13}\lambda_1^2$ impliziert. Die Schranke aus den Ungleichungen 1.7 garantiert: $\| \mathbf{b}_1 \|^2 \leq \frac{4}{3}\lambda_1$. Der Unterschied ist also $\frac{5 \cdot 13}{3^4} \approx 1.2461$, gegenüber $\frac{4}{3} = 1.\overline{3}$. Das Verhältnis ist ungefähr 93.46%.

Kapitel 3

Das Konzept der parametrisierten Eingaben

Zunächst wird dargelegt, was das *Konzept der parametrisierten Eingaben* ist. Dann wird eine durch $0 \leq s \in \mathbb{Z}$ parametrisierte zulässige Matrix $A_s \in \mathbb{Z}^{3\times 3}$ für den LLL-Algorithmus angegeben, bei deren Eingabe der LLL-Algorithmus wenigsten $3s$ Iterationen ausführt. Zum Ende des Kapitels wird gezeigt, wie aus A_s eine parametrisierte Eingabe für den LLL-Algorithmus für beliebige Dimension konstruiert werden kann, um damit das Konzept auch im allgemeinen Fall anwenden zu können.

3.1 Was sind parametrisierte Eingaben?

Für die in dieser Arbeit untersuchten Reduktionsalgorithmen werden Eingaben A_s vorgelegt, die von einem Parameter $0 \leq s \in \mathbb{Z}$ abhängen. Durch diesen Parameter s kann die Laufzeit der Algorithmen und die Eingabegröße $E(A_s)$ simultan gesteuert werden.

3.2 Das Konzept

Durch die Möglichkeit, Eingabegröße und Laufzeit simultan zu steuern, können untere Schranken für die Laufzeit als Funktion der Eingabegröße bewiesen werden. Dies wird in der vorliegenden Arbeit dazu verwendet, untere Schranken für die Fälle zu beweisen, bei denen bisher für den LLL-Algorithmus keine oberen Schranken bekannt sind. Ferner werden auf diese Weise untere Laufzeitschranken für einen eigenen Algorithmus, zur Berechnung Minkowski-reduzierter Matrizen, bewiesen.

3.3 Eine parametrisierte Eingabe für den LLL-Algorithmus für Dimension drei

In den folgenden Abschnitten wird zu der unten angegebenen parametrisierten Eingabe A_s für den LLL-Algorithmus die Gültigkeit von Satz 3.1 bewiesen.

Satz 3.1 *Sei s eine positive ganze Zahl und bezeichne $A_s = A \cdot G^s$, wobei*

$$A = \begin{pmatrix} 0 & -1 & 1 \\ 1 & 0 & -2 \\ 0 & 1 & 0 \end{pmatrix}$$

und

$$G = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & -2 \\ 0 & 1 & 0 \end{pmatrix}$$

ist. Ferner sei λ die positive reelle Wurzel des Polynoms $x^3 + 4x^2 + 4x - 1$. Dann gibt es eine Folge reeller Zahlen $(\sigma_s) \in \mathbb{R}^{\mathbb{N}}$ mit $0.781 < \sigma_s < 1.122$, so daß gilt:

$$E(A_s) = (2 + \lambda)^s \cdot \sigma_s \text{ für } s > 10. \tag{3.1}$$

Außerdem gilt:

$$T(A_s, 1) = 5 + 3s. \tag{3.2}$$

Der Beweis von Satz 3.1 ist lang und kompliziert und wird daher in mehrere Lemmata untergliedert. Für den Beweis des zentralen Lemmas 3.2, das zum Beweis von Gleichung (3.2) dient, werden Hilfssätze benötigt, die als Nebenprodukt Gleichung (3.1) verifizieren.

Der LLL$(A_s, 1)$-Algorithmus zeigt *periodisches Verhalten.* Damit ist gemeint, daß sich eine Folge von Anweisung in der gleichen Weise wiederholt. Dieses periodische Verhalten wird in Lemma 3.2 formalisiert.

Lemma 3.2 *Sei $s \in \mathbb{Z} > 0$ und sei A_s wie in Satz 3.1 definiert. Dann gilt:*

$$T(A_s, 1) = T(A_{s-1}, 1) + 3. \tag{3.3}$$

Der Beweis von Lemma 3.2 erfordert einige technische Lemmata, die im folgenden angegeben werden.

Lemma 3.3 *Sei $0 \le s \in \mathbb{Z}$. Bezeichne $\mathbf{a}_s$ den ersten Spaltenvektor der Matrix A_s aus Satz 3.1 und $\mu_{ij}^{(s)}$ die Gram-Schmidt-Koeffizienten von A_s für $1 \le j < i \le 3$. Ferner sei*

$$(p_{ij}^{(s)})_{1 \le i,j \le 3} = (A_s^t \cdot A_s).$$

Dann gelten die folgenden Gleichungen und Ungleichungen:

$$A_s = (\mathbf{a}_s, \mathbf{a}_{s+1}, \mathbf{a}_{s+2}) \tag{3.4}$$

$$\mathbf{a}_{s+3} = \mathbf{a}_s - 2\,\mathbf{a}_{s+1} \tag{3.5}$$

$$|\mu_{21}^{(s)}| \le \frac{1}{2} \tag{3.6}$$

$$-\frac{5}{2} \leq \mu_{31}^{(s)} < -\frac{3}{2} \tag{3.7}$$

$$-\frac{1}{2} < \mu_{32}^{(s)} < 0 \tag{3.8}$$

$$\| \mathbf{a}_s \|^2 < \| \mathbf{a}_{s+1} \|^2 \tag{3.9}$$

$$p_{11}^{(s)} p_{33}^{(s)} - p_{13}^{(s)} p_{13}^{(s)} < p_{11}^{(s)} p_{22}^{(s)} - p_{12}^{(s)} p_{12}^{(s)} \tag{3.10}$$

Die Ungleichungen werden dadurch bewiesen, daß sogenannte geschlossene Darstellungen für die darin auftretenden Größen bestimmt werden. Dazu werden die erforderlichen Rekursionsgleichungen gefunden und gelöst.

Für den Beweis des Lemmas 3.2, wird zusätzlich noch Lemma 3.4 benötigt. Im folgenden bezeichnen wir mit 2.2.i die Anweisung i in Algorithmus 2.2. Analoge Bezeichnungen werden für die Anweisungen der anderen Algorithmen verwendet.

Lemma 3.4 *Sei $y \in (1/4, 1]$ und $A \in \mathbb{Z}^{3\times 3}$ eine reguläre Matrix. Bezeichne c_j die quadrierte euklidische Länge des j-ten Gram-Schmidt-Vektors und μ_{ij} $(1 \leq j < i \leq n)$ die Gram-Schmidt-Koeffizienten zu A. Dann gelten die folgenden Äquivalenzen:*

(a) Für $k = 2$ ist $c_2 < (y - \mu_{21}^2)c_1$ äquivalent zu $\| \mathbf{a}_2 \|^2 < y \cdot \| \mathbf{a}_1 \|^2$.

(b) Für $k = 3$ ist $c_3 < (y - \mu_{32}^2)c_2$ äquivalent zu $p_{11}^{(s)} p_{33}^{(s)} - p_{13}^{(s)} p_{13}^{(s)} < p_{11}^{(s)} p_{22}^{(s)} - p_{12}^{(s)} p_{12}^{(s)}$.

Beweis: Eine kurze Rechnung ergibt die Korrektheit des Lemmas. □

Der lange technische Beweis von Lemma 3.3 erfolgt am Ende dieses Abschnitts. An dieser Stelle wird die Gültigkeit von Lemma 3.3 benutzt, um Lemma 3.2 zu beweisen. Beim Beweis von Lemma 3.2 wird Lemma 3.3 nicht benutzt. Lemma 3.2 ergibt die Gültigkeit von Gleichung (3.2).

Beweis von Lemma 3.2 Der Beweis wird so geführt, daß Schritt für Schritt untersucht wird, was im Algorithmus 2.2 bei Eingabe von $(A_s, 1)$ passiert. Dabei wird $s > 1$ vorausgesetzt. Es zeigt sich, daß sich der Algorithmus 2.1 nach drei Gaußschritten vor Anweisung 2.1.4 befindet, und die Situation die gleiche ist, wie die Situation, die sich für LLL$(A_{s-1}, 1)$ nach der Initialisierung und vor Anweisung 2.1.4 ergibt.

Die Vorgänge werden in Tabelle 3.1 zusammengefaßt und im Anschluß erklärt.

Zeile im Algorithmus	Aktion	$(\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3)$ vorher	$(\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3)$ nachher	Referenz
2.1.2	Initialisierung	$(\mathbf{a}_s, \mathbf{a}_{s+1}, \mathbf{a}_{s+2})$	$(\mathbf{a}_s, \mathbf{a}_{s+1}, \mathbf{a}_{s+2})$	(3.4)
2.1.3	$k = 2$	$(\mathbf{a}_s, \mathbf{a}_{s+1}, \mathbf{a}_{s+2})$	$(\mathbf{a}_s, \mathbf{a}_{s+1}, \mathbf{a}_{s+2})$	
2.1.4	$k \neq 4$ weiter	$(\mathbf{a}_s, \mathbf{a}_{s+1}, \mathbf{a}_{s+2})$	$(\mathbf{a}_s, \mathbf{a}_{s+1}, \mathbf{a}_{s+2})$	
2.1.5	$l = k - 1 = 1$	$(\mathbf{a}_s, \mathbf{a}_{s+1}, \mathbf{a}_{s+2})$	$(\mathbf{a}_s, \mathbf{a}_{s+1}, \mathbf{a}_{s+2})$	
2.1.6	Gaußstep$(2, 1)$	$(\mathbf{a}_s, \mathbf{a}_{s+1}, \mathbf{a}_{s+2})$	$(\mathbf{a}_s, \mathbf{a}_{s+1}, \mathbf{a}_{s+2})$	(3.6)
2.1.7	$\parallel \mathbf{a}_{k+1} \parallel^2 < \parallel \mathbf{a}_k \parallel^2$	$(\mathbf{a}_s, \mathbf{a}_{s+1}, \mathbf{a}_{s+2})$	$(\mathbf{a}_s, \mathbf{a}_{s+1}, \mathbf{a}_{s+2})$	(3.9), Lemma 3.4 (a)
2.1.8	$l = 0$ down to 1	$(\mathbf{a}_s, \mathbf{a}_{s+1}, \mathbf{a}_{s+2})$	$(\mathbf{a}_s, \mathbf{a}_{s+1}, \mathbf{a}_{s+2})$	
2.1.10	$k = k + 1 = 3$	$(\mathbf{a}_s, \mathbf{a}_{s+1}, \mathbf{a}_{s+2})$	$(\mathbf{a}_s, \mathbf{a}_{s+1}, \mathbf{a}_{s+2})$	
2.1.3	$k \neq 4$ weiter	$(\mathbf{a}_s, \mathbf{a}_{s+1}, \mathbf{a}_{s+2})$	$(\mathbf{a}_s, \mathbf{a}_{s+1}, \mathbf{a}_{s+2})$	
2.1.4	$l = k - 1 = 2$	$(\mathbf{a}_s, \mathbf{a}_{s+1}, \mathbf{a}_{s+2})$	$(\mathbf{a}_s, \mathbf{a}_{s+1}, \mathbf{a}_{s+2})$	
2.1.5	Gaußstep(3,2)	$(\mathbf{a}_s, \mathbf{a}_{s+1}, \mathbf{a}_{s+2})$	$(\mathbf{a}_s, \mathbf{a}_{s+1}, \mathbf{a}_{s+2})$	(3.8)
2.1.6	Bedingung erfüllt	$(\mathbf{a}_s, \mathbf{a}_{s+1}, \mathbf{a}_{s+2})$	$(\mathbf{a}_s, \mathbf{a}_{s+1}, \mathbf{a}_{s+2})$	(3.10), Lemma 3.4 (b)
2.1.7	swap-update(3)	$(\mathbf{a}_s, \mathbf{a}_{s+1}, \mathbf{a}_{s+2})$	$(\mathbf{a}_s, \mathbf{a}_{s+2}, \mathbf{a}_{s+1})$	
2.1.3	$k \neq 4$ weiter	$(\mathbf{a}_s, \mathbf{a}_{s+2}, \mathbf{a}_{s+1})$	$(\mathbf{a}_s, \mathbf{a}_{s+2}, \mathbf{a}_{s+1})$	
2.1.4	$l = k - 1 = 1$	$(\mathbf{a}_s, \mathbf{a}_{s+1}, \mathbf{a}_{s+2})$	$(\mathbf{a}_s, \mathbf{a}_{s+1}, \mathbf{a}_{s+2})$	
2.1.5	Gaußstep(2,1)	$(\mathbf{a}_s, \mathbf{a}_{s+2}, \mathbf{a}_{s+1})$	$(\mathbf{a}_s, \mathbf{a}_{s+2} + 2\mathbf{a}_s, \mathbf{a}_{s+1})$	(3.7)
2.1.6	Bedingung erfüllt	$(\mathbf{a}_s, \mathbf{a}_{s+2} + 2\mathbf{a}_s, \mathbf{a}_{s+1})$	$(\mathbf{a}_s, \mathbf{a}_{s+2} + 2\mathbf{a}_s, \mathbf{a}_{s+1})$	(3.10), Lemma 3.4 (a)
2.1.7	swap-update(2)	$(\mathbf{a}_s, \mathbf{a}_{s+2} + 2\mathbf{a}_s, \mathbf{a}_{s+2})$	$(\mathbf{a}_{s+2} + 2\mathbf{a}_s, \mathbf{a}_s, \mathbf{a}_{s+1})$	
2.1.3	$k \neq 4$ weiter	$(\mathbf{a}_s, \mathbf{a}_{s+2}, \mathbf{a}_{s+1})$	$(\mathbf{a}_s, \mathbf{a}_{s+2}, \mathbf{a}_{s+1})$	
2.1.4	$k = 2$	$(\mathbf{a}_{s+2} + 2\mathbf{a}_s, \mathbf{a}_s, \mathbf{a}_{s+1})$	$(\mathbf{a}_{s-1}, \mathbf{a}_s, \mathbf{a}_{s+1})$	(3.5)

Tabelle 3.1: Drei Gaußschritte im LLL$(A_s, 1)$-Algorithmus

In der obenstehenden Tabelle wurde zusammengefaßt, wie die Eingabematrix A_s für $s \in \mathbb{Z}^{>0}$ in drei Gaußschritten zu der Matrix A_{s-1} im LLL(1)-Algorithmus modifiziert wurde. Die Initialisierung verändert die Eingabematrix nicht. Es wurde in Lemma 3.3, Gleichung (3.4) gezeigt, daß $A_s = (\mathbf{a}_s, \mathbf{a}_{s+1}, \mathbf{a}_{s+2})$ gilt. Danach wird k auf den Wert zwei gesetzt, was keinen Einfluß auf die Matrix hat. Da $k \neq 4$ gilt, wird nun die While-Schleife abgearbeitet. In Anweisung 2.1.4 wird $l = 1$ gesetzt. Auch Anweisung 2.1.5 hat keinen Einfluß auf die Matrix, da wegen der Gültigkeit von (3.6) keine Operation ausführt wird. Die Bedingung in Anweisung 2.1.6 ist nun wegen Gleichung (3.9) und Lemma 3.4 nicht erfüllt, so daß als nächstes Anweisung 2.1.8 abgearbeitet wird. Die For-Schleife 2.1.8 bis 2.1.9 wird nicht ausgeführt, da der Endlaufindex kleiner als der Startlaufindex ist. Danach wird k in 2.1.10 inkrementiert. Damit ist der erste Durchlauf durch die While-Schleife abschlossen.

Da $k = 3$ ist, wird die While-Schleife erneut durchlaufen. In 2.1.4 wird $l = 2$ gesetzt. Anweisung 2.1.5 hat keinen Effekt, wegen Gleichung (3.8). Diesmal ist die Bedingung in 2.1.6 nach Lemma 3.4 und Gleichung (3.10) erfüllt. Dies hat den Tausch der zweiten und dritten Spalte der Eingabematrix in Anweisung 2.1.7 zur Folge. Außerdem wird k dekrementiert.

Beim dritten Durchlauf durch die While-Schleife gilt $k = 2$. Anweisung 2.1.5 hat jetzt den in der Tabelle angegebenen Effekt, da zum einen für den momentanen Gram-Schmidt-Koeffizienten $\mu_{21} = \mu_{31}^{(s)}$ und zum anderen Gleichung (3.7) gilt. Nach Gleichung (3.5) ist $\mathbf{a}_{s-1} = \mathbf{a}_{s+2} + 2\,\mathbf{a}_s$. Die Bedingung, die in Anweisung 2.1.6 getestet wird, lautet daher nach Lemma 3.4 $\parallel \mathbf{a}_{s-1} \parallel^2 < \parallel \mathbf{a}_s \parallel^2$. Diese Bedingung ist nach Gleichung (3.9) erfüllt. Damit werden die erste und die zweite Spalte der bis dahin berechneten Matrix vertauscht. Die Ausführung des Algorithmus geht in 2.1.4 mit $k = 2$ weiter und die aktuelle Matrix ist

$(\mathbf{a}_{s-1}, \mathbf{a}_s, \mathbf{a}_{s+1})$. Es kann unmittelbar verifiziert werden, daß der LLL$(A_0, 1)$-Algorithmus genau fünf Gaußschritte ausführt. Damit ist Lemma 3.2 bewiesen. □

Bevor der Beweis des Lemmas 3.3 geführt wird, möchten wir einige Bemerkungen zur verwendeten beziehungsweise alternativen Beweisstrategien geben.

Die verwendete Methode, die Gültigkeit der Ungleichungen (3.6) bis (3.10) zu zeigen, ist die, Rekursionsgleichungen für $\| \mathbf{a}_s \|^2$ und $p_{11}^{(s)} p_{33}^{(s)} - p_{13}^{(s)} p_{13}^{(s)}$ aufzustellen und zu lösen.

Die *Lösung einer Rekursionsgleichung* beinhaltet die Angabe einer Formel, aus der sich das $s-te$ Folgeglied ohne die Berechnung seiner Vorgänger berechnen läßt.

Mit der geschlossenen Darstellung von $\| \mathbf{a}_s \|^2$ wird (3.9) beweisen. Ferner wird die Lösung der Rekursion für $\| \mathbf{a}_s \|^2$ dazu verwendet, Gleichung (3.6) zu verifizieren. Mit der Gültigkeit der Gleichungen (3.6) und (3.9) folgen die Ungleichungen (3.7) unmittelbar, denn es gilt:

$$\mu_{31}^{(s+1)} = -2 + \mu_{21} \frac{\| \mathbf{a}_s \|^2}{\| \mathbf{a}_{s+1} \|^2}, \tag{3.11}$$

was sich aus Gleichung (3.4) und (3.5) ergibt. Um die Gültigkeit der Gleichungen (3.8) und (3.10) zu zeigen, muß eine weitere Rekursionsgleichung für $p_{11}^{(s)} p_{33}^{(s)} - p_{13}^{(s)} p_{13}^{(s)}$ aufgestellt und gelöst werden.

Das große technische Problem im Beweis der geschlossenen Darstellungen entsteht durch Terme mit wechselndem Vorzeichen, die einen kleinen Absolutbetrag haben, aber nicht gegen Null konvergieren.

Zunächst wird begründet, warum wir **nicht** den unten erläuterten, naheliegenden Weg gegangen sind, sondern einen, auf den ersten Blick umständlicheren Weg, der aber im Endeffekt praktikabler und einfacher ist, gewählt haben.

Der naheliegende Lösungsweg ist der, die Matrix G zu diagonalisieren und die Matrix A bezüglich der Basis der Eigenvektoren (EVen) der Matrix G darzustellen. Mit der Bezeichnung V^t für die Matrix, die in ihrer i-ten Spalte den EV $(1, \lambda_i, \lambda_i^2)^t$ zum Eigenwert (EW) λ_i der Matrix G für $1 \le i \le 3$ hat, gilt dann:

$$V \cdot G^s = \begin{pmatrix} \lambda_1^s & 0 & 0 \\ 0 & \lambda_2^s & 0 \\ 0 & 0 & \lambda_3^s \end{pmatrix} \cdot V$$

und

$$A_s = X \cdot \begin{pmatrix} \lambda_1^s & 0 & 0 \\ 0 & \lambda_2^s & 0 \\ 0 & 0 & \lambda_3^s \end{pmatrix} \cdot V$$

für $s \in \mathbb{Z}^{\ge 0}$. Dabei ist $X = (x_{ij})_{1 \le i,j \le 3}$ die eindeutig bestimmte Lösung des linearen Gleichungssystems

$$A = X \cdot V. \tag{3.12}$$

Die geschlossene Form des Vektors $\mathbf{a}_s$ lautet dann

$$\mathbf{a}_s = \begin{pmatrix} x_{11}\lambda_1^s + x_{12}\lambda_2^s + x_{13}\lambda_3^s \\ x_{21}\lambda_1^s + x_{22}\lambda_2^s + x_{23}\lambda_3^s \\ x_{31}\lambda_1^s + x_{32}\lambda_2^s + x_{33}\lambda_3^s \end{pmatrix}. \tag{3.13}$$

Die Koeffizienten (x_{ij}) mit $1 \leq i,j \leq 3$, die sich aus Gleichung (3.12) ausgedrückt durch die EWe der Matrix G ergeben, haben ein äußerst kompliziertes Aussehen. Wird nun die geschlossene Form (3.13) verwendet, um die eigentlich relevanten Größen, wie etwa $\| \mathbf{a}_s \|^2$, zu bestimmen, so ergeben sich Terme, die bezüglich der gesuchten Aussagen schwer zu analysieren sind. Es erscheint daher sinnvoller, direkt die Rekursion für die Folge $\| \mathbf{a}_s \|^2$ herzuleiten und zu lösen. Diese Vorgehensweise hat zwei Vorteile: Zum einen kann die Lösung dieser Rekursiongleichung zweimal verwendet werden (beim Beweis von (3.5) und (3.9)) und zum anderen läßt sich die erhaltene Darstellung, wie sich a posterio herausstellt, sehr gut analysieren.

Es folgt nun der Beweis von Lemma 3.3:

Beweis von Lemma 3.3 Zunächst wird die Richtigkeit der Gleichungen (3.4) und (3.5) des Lemmas gezeigt. Die Gültigkeit von Gleichung (3.4) ergibt sich unmittelbar. Die Rekursionsgleichung (3.5) gilt nach dem Satz von Cayley-Hamilton. Das charakteristische Polynom $p(x)$ der Matrix G ist

$$p(x) = x^3 + 2\,x - 1.$$

Also ist nach dem Satz von Cayley-Hamilton $p(G) = 0$ und damit $G^3 = I_3 - 2\,G$. Hierin ist I_3 die Einheitsmatrix vom Rang drei. Multiplikation von links mit der Matrix $A \cdot G^s$ ergibt die Rekursionsgleichung (3.5).

Jetzt wird die Rekursionsgleichung für die Folge $(\| \mathbf{a}_s \|^2)$ hergleitet und gelöst.

Hilfssatz 3.5 *Es bezeichne $(a_s) = (\| \mathbf{a}_s \|^2)$ die Folge der quadrierten euklidischen Längen der Folge von Vektoren $(\mathbf{a}_s)$. Die Folge $(a_s) \in \mathbb{Z}^{\mathbb{N}}$ genügt der linearen Rekursion:*

$$a_{s+6} = -2\,a_{s+5} + 4\,a_{s+4} + 10\,a_{s+3} + 2\,a_{s+2} + 4\,a_{s+1} - a_s \tag{3.14}$$

mit

$$(a_0, a_1, \ldots, a_5) = (1, 2, 5, 9, 26, 41).$$

Beweis: Das Gleichungssystem

$$\begin{pmatrix} a_s \\ a_{s+1} \\ a_{s+2} \\ p_{12}^{(s)} \\ p_{13}^{(s)} \\ p_{23}^{(s)} \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 4 & 0 & -4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & -2 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -2 \end{pmatrix}^s \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ p_{12}^{(0)} \\ p_{13}^{(0)} \\ p_{23}^{(0)} \end{pmatrix} \tag{3.15}$$

ergibt sich nach den Gleichungen (3.4) und (3.5) durch Induktion. Die 6×6 Matrix aus Gleichung (3.15), die für $s = 1$ mit M bezeichne wird, hat das charakteristische Polynom $p(x) = x^6 + 2\,x^5 - 4\,x^4 - 10\,x^3 - 2\,x^2 - 4\,x + 1$. Also ist nach dem Satz von Cayley-Hamilton $p(M) = 0$, wobei dann die 1 durch die 6×6 Einheitsmatrix zu ersetzen ist. Damit ist die Rekursion (3.14) gezeigt. $\square$

Jetzt wird die Rekursion (3.14) gelöst.

Hilfssatz 3.6 *Sei $p(x) = x^3 + 4\,x^2 + 4\,x - 1$ und seien $\lambda_i \in \mathbb{C}$ $(1 \le i \le 3)$ mit $p(\lambda_i) = 0$. Sei $(a_s) \in \mathbb{Z}^{\mathbb{N}}$ die Folge aus Hilfssatz 3.5.*

Die geschlossene Darstellung der Folge (a_s) lautet:

$$a_s = \sum_{i=1}^{3} c_i\,\lambda_i^s + \sum_{i=1}^{3} c_{i+3}\,(\lambda_i + 2)^s. \tag{3.16}$$

Dabei ist $(c_1, c_2, \ldots, c_6)$ die eindeutig bestimmte Lösung des linearen inhomogenen Gleichungssystems (3.17).

$$\begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ \lambda_1 & \lambda_2 & \lambda_3 & (\lambda_1+2) & (\lambda_2+2) & (\lambda_3+2) \\ \lambda_1^2 & \lambda_2^2 & \lambda_3^2 & (\lambda_1+2)^2 & (\lambda_2+2)^2 & (\lambda_3+2)^2 \\ \lambda_1^3 & \lambda_2^3 & \lambda_3^3 & (\lambda_1+2)^3 & (\lambda_2+2)^3 & (\lambda_3+2)^3 \\ \lambda_1^4 & \lambda_2^4 & \lambda_3^4 & (\lambda_1+2)^4 & (\lambda_2+2)^4 & (\lambda_3+2)^4 \\ \lambda_1^5 & \lambda_2^5 & \lambda_3^5 & (\lambda_1+2)^5 & (\lambda_2+2)^5 & (\lambda_3+2)^5 \end{pmatrix} \cdot \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \end{pmatrix} \tag{3.17}$$

Beweis: Nach Hilfssatz 3.5 gilt:

$$\begin{pmatrix} a_s \\ a_{s+1} \\ a_{s+2} \\ a_{s+3} \\ a_{s+4} \\ a_{s+5} \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ -1 & 4 & 2 & 10 & 4 & -2 \end{pmatrix}^s \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{pmatrix}. \tag{3.18}$$

Bezeichne R die rechtsstehende 6×6 Matrix für $s = 1$. Für das charakteristische Polynom der Matrix R gilt:

$$\operatorname{char}(R, x) = p(x) \cdot q(x),$$

wobei $q(x) = x^3 - 2\,x - 1$ ist. Offenbar gilt $p(x) = q(x + 2)$, so daß

$$E = \{\lambda_i, \lambda_{i+2} : 1 \le i \le 3\}$$

die Menge der Eigenwerte der Matrix R ist. Ist $\lambda \in E$, so ist $v_\lambda = (1, \lambda, \lambda^2, \ldots, \lambda^5)^t$ der zugehörige Eigenvektor zum Eigenwert λ. Die Eigenwerte der Matrix R sind paarweise verschieden, also ist $\det(R) \neq 0$, da R eine Vandermonde Matrix ist. Daher ist (3.17) eindeutig lösbar. Die behauptete geschlossene Darstellung für a_s kann schließlich aus den Gleichungen

$$\begin{pmatrix} a_s \\ a_{s+1} \\ a_{s+2} \\ a_{s+3} \\ a_{s+4} \\ a_{s+5} \end{pmatrix} = R^s \cdot \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{pmatrix} = \begin{pmatrix} \lambda_1\,v_{\lambda_1} \\ \lambda_2\,v_{\lambda_2} \\ \lambda_3\,v_{\lambda_3} \\ (\lambda_1+2)\,v_{\lambda_1+2} \\ (\lambda_2+2)\,v_{\lambda_2+2} \\ (\lambda_3+2)\,v_{\lambda_3+2} \end{pmatrix}^t \cdot \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \end{pmatrix}$$

abgelesen werden. $\square$

Die Rekursiongleichung (3.14) wird im Beweis des Lemmas 3.3 für verschiedene Startwerte $(a_0, \ldots, a_5)$ benutzt. Ein solches geordnetes Sechstupel für eine Rekursiongleichung wird als ein *Startvektor zur Rekursion* bezeichnet. Da wir an einer Lösung des Gleichungssystem (3.17) für beliebige ganzzahlige Startvektoren interessiert sind, bestimmen wir die sechs Lösungen für die kanonischen Einheitsvektoren. Aus diesen Lösungen kann jede Lösung zu beliebigem Startvektor durch Linearkombination gewonnen werden. Die Lösungen werden in Abhängigkeit der Eigenwerte der Matrix dargestellt, welche die Rekursion (3.14) beschreibt.

Hilfssatz 3.7 *Es gelten die Bezeichnungen von Hilfssatz 3.6 und* $\mathbf{c}_i$ *bezeichne die eindeutig bestimmte Lösung des linearen inhomogenen Gleichungssystems (3.17), wobei der Spaltenvektor* $(a_0, \ldots, a_5)^t$ *durch den i-ten kanonischen Einheitsvektor für* $1 \leq i \leq 6$ *ersetzt wurde. Dann gelten die folgenden Gleichungen:*

$$\mathbf{c}_1 = \frac{1}{118} \begin{pmatrix} 16\lambda_1^2 + 67\lambda_1 + 78 \\ 16\lambda_2^2 + 67\lambda_2 + 78 \\ 16\lambda_3^2 + 67\lambda_3 + 78 \\ -8\lambda_1^2 - 19\lambda_1 + 4 \\ -8\lambda_2^2 - 19\lambda_2 + 4 \\ -8\lambda_3^2 - 19\lambda_3 + 4 \end{pmatrix}$$

$$\mathbf{c}_2 = \frac{1}{118} \begin{pmatrix} 14\lambda_1^2 + 60\lambda_1 + 67 \\ 14\lambda_2^2 + 60\lambda_2 + 67 \\ 14\lambda_3^2 + 60\lambda_3 + 67 \\ 42\lambda_1^2 + 88\lambda_1 - 19 \\ 42\lambda_2^2 + 88\lambda_2 - 19 \\ 42\lambda_3^2 + 88\lambda_3 - 19 \end{pmatrix}$$

$$\mathbf{c}_3 = \frac{1}{118} \begin{pmatrix} 35\lambda_1^2 + 148\lambda_1 + 172 \\ 35\lambda_2^2 + 148\lambda_2 + 172 \\ 35\lambda_3^2 + 148\lambda_3 + 172 \\ -11\lambda_1^2 + 26\lambda_1 - 4 \\ -11\lambda_2^2 + 26\lambda_2 - 4 \\ -11\lambda_3^2 + 26\lambda_3 - 4 \end{pmatrix}$$

$$\mathbf{c}_4 = \frac{1}{118} \begin{pmatrix} 12\lambda_1^2 + 53\lambda_1 + 56 \\ 12\lambda_2^2 + 53\lambda_2 + 56 \\ 12\lambda_3^2 + 53\lambda_3 + 56 \\ -20\lambda_1^2 - 21\lambda_1 + 8 \\ -20\lambda_2^2 - 21\lambda_2 + 8 \\ -20\lambda_3^2 - 21\lambda_3 + 8 \end{pmatrix}$$

$$\mathbf{c}_5 = \frac{1}{118} \begin{pmatrix} -8\lambda_1^2 - 32\lambda_1 - 35 \\ -8\lambda_2^2 - 32\lambda_2 - 35 \\ -8\lambda_3^2 - 32\lambda_3 - 35 \\ 2\lambda_1^2 - 4\lambda_1 + 3 \\ 2\lambda_2^2 - 4\lambda_2 + 3 \\ 2\lambda_3^2 - 4\lambda_3 + 3 \end{pmatrix}$$

$$\mathbf{c}_6 = \frac{1}{118}\begin{pmatrix} -3\lambda_1^2 - 14\lambda_1 - 16 \\ -3\lambda_2^2 - 14\lambda_2 - 16 \\ -3\lambda_3^2 - 14\lambda_3 - 16 \\ 3\lambda_1^2 + 2\lambda_1 \\ 3\lambda_2^2 + 2\lambda_2 \\ 3\lambda_3^2 + 2\lambda_3 \end{pmatrix}.$$

Beweis: Die Richtigkeit der Gleichungen wird durch Rechnung mit einem geeigneten Computeralgebrasystem (maple, mathematica, PARI) nachgeprüft. Dabei sind die Vereinfachungen der Ausdrücke, die sich durch den folgenden Hilfssatz 3.8 ergeben, unerläßlich. □

Hilfssatz 3.8 *Seien $p(x)$ und $\lambda_i\,(1 \le i \le 3)$ wie in Hilfssatz 3.6, mit dem Zusatz, daß λ_1 die reelle Wurzel von $p(x)$ ist, definiert. Dann gilt:*

$$\lambda_1 + \lambda_2 + \lambda_3 = -4 \tag{3.19}$$

und

$$\lambda_1\lambda_2\lambda_3 = 1. \tag{3.20}$$

Ferner gilt: $\lambda_1^2 + \lambda_2^2 + \lambda_3^2 = 8$ und $|\lambda_2| = |\lambda_1 + 2|$.

Beweis: Es gilt $\lambda_1 + \lambda_2 + \lambda_3 = (-1)^3 \cdot (\text{Koeffizient von } x^2)$ in $p(x) = -4$ und $\lambda_1\lambda_2\lambda_3 = (-1)^3 \cdot (\text{Koeffizient von } x^0)$ in $p(x) = (-1)^4 = 1$. Wegen (3.19) und (3.20) gilt: $(\sum_{i=1}^3 \lambda_i)^2 = \sum_{i=1}^3 \lambda_i^2 + 2\sum_{1\le i<j\le 3} \lambda_i\lambda_j = (-4)^2 = 16 = \sum_{i=1}^3 \lambda_i^2 + 2\sum_{i=1}^3 \frac{1}{\lambda_i}$. Es ist also:

$$\sum_{i=1}^{3} \lambda_i^2 + 2\sum_{i=1}^{3} \frac{1}{\lambda_i} = 16 \tag{3.21}$$

Ferner folgt, wegen (3.19) und $p(\lambda_i) = 0\,(1 \le i \le 3)$:

$$\sum_{i=1}^{3}(\lambda_i^2 - \frac{1}{\lambda_i}) = 4 \tag{3.22}$$

Bilden wir nun die Summe aus dem zweifachen von Gleichung (3.22) und Gleichung (3.21), so erhalten wir $3 \cdot \sum_{i=1}^3 \lambda_i^2 = 24$ und damit unmittelbar $\sum_{i=1}^3 \lambda_i^2 = 8$. Ferner ist $|\lambda_2|^2 = \lambda_2 \cdot \overline{\lambda_2} = \lambda_2 \cdot \lambda_3 = \frac{1}{\lambda_1} = \lambda_1^2 + 4\lambda_1 + 4 = (\lambda_1 + 2)^2$, womit der Hilfssatz bewiesen ist. □

Es sind nun die wesentlichen Hilfsmittel zusammengestellt worden, um Gleichung (3.9) zu beweisen.

Als erstes Korollar des Hilfssatzes 3.7 wird jetzt die explizite geschlossene Form der Folge $(a_s) \in \mathbb{Z}^{\mathbb{N}}$, wie in Hilfssatz 3.5 formuliert, angegeben.

Korollar 3.9 *Sei $(a_s) \in \mathbb{Z}^{\mathbb{N}}$ die Folge aus Hilfssatz 3.7 und seien $\lambda_i\ (1 \le i \le 3)$ wie in Hilfssatz 3.8 definiert. Dann gilt für die Koeffizienten c_i aus Gleichung (3.16):*

$$59 \cdot c_i = \begin{cases} -2\,\lambda_i^2 - \lambda_i + 5 & \textit{für} \quad 1 \le i \le 3 \\ 8\,\lambda_{i-3}^2 + 38\,\lambda_{i-3} + 48 & \textit{für} \quad 3 < i \le 5. \end{cases}$$

Damit kann die Folge (a_s), mit den oben definierten c_i, wie folgt geschrieben werden:

$$a_s = c_1\,\lambda_1^s + 2\,Re(\lambda_2^s \cdot c_2) + c_4\,(\lambda_1 + 2)^s + 2\,Re((\lambda_2 + 2)^s \cdot c_5) \tag{3.23}$$

Daraus ergeben sich die Ungleichungen:

$$0.781 \cdot (\lambda_1 + 2)^s - |\epsilon(s)| < a_s < 1.122 \cdot (\lambda_1 + 2)^s + |\epsilon(s)| \tag{3.24}$$

Hierbei ist $|\epsilon(s)| < 3 \cdot 0.674^{-s}$ *für* $s \in \mathbb{Z}^{\geq 0}$.

Beweis: Die Gleichungen für die c_i ergeben sich aus Hilfssatz 3.7 mit

$$\begin{pmatrix} c_1 \\ \vdots \\ c_6 \end{pmatrix} = \sum_{i=1}^{6} a_{i-1}\mathbf{c}_i.$$

Dabei ist $(a_0, a_1, \ldots, a_5)$ wie in Hilfssatz 3.5 und $\mathbf{c}_i$ $(1 \leq i \leq 6)$ wie in Hilfssatz 3.7 definiert. Da λ_1 die einzige reelle Wurzel des Polynoms $p(x)$ aus Hilfssatz 3.6 ist, gilt $\lambda_2 = \overline{\lambda_3}$ für die restlichen Wurzeln des Polynoms $p(x)$. Damit ist auch $c_2 = \overline{c_3}$ und $c_5 = \overline{c_6}$. Es gilt ganz allgemein $2\,\mathrm{Re} = z + \overline{z}$ für alle $z \in \mathbb{C}$. Damit ist Gleichung (3.23) hergeleitet.

Die Ungleichungen (3.24) ergeben sich aus untenstehenden numerischen Approximationen und der in Hilfssatz 3.8 bewiesenen Identität $|\lambda_2| = \lambda_1 + 2$.

Bezeichnung der Wurzel	Numerische Approximation
$\lambda_1 \in \mathbb{R}$	$0.205 + \epsilon$
$\lambda_2 \in \mathbb{C} - \mathbb{R}$	$-(2.102 + \epsilon) + (0.665 + \epsilon)\,i$
$\lambda_3 = \overline{\lambda_2} \in \mathbb{C} - \mathbb{R}$	$-(2.102 + \epsilon) - (0.665 + \epsilon)\,i$

Tabelle 3.2: Approximationen der Wurzeln des Polynoms $p(x) = x^3 + 4\,x^2 + 4\,x - 1$

Die Terme, die exponentiell gegen Null konvergieren, wurden in $\epsilon(s)$ zusammengefaßt. Dies sind die Terme in der geschlossenen Darstellung von (a_s), die Potenzen von λ_1, $(\lambda_2 + 2)$ und $(\lambda_3 + 2)$ enthalten. Die Ungleichungen ergeben sich aus den Abschätzungen $-|z| \leq \mathrm{Re}(z) \leq |z|$, die für alle $z \in \mathbb{C}$ gelten. Die Unschärfe ist unvermeidbar, da es keine andere Möglichkeit gibt, die Vorzeichenfolge $(s_i) \in \{\pm 1\}$ des kleinen Störungsterms $2\mathrm{Re}(\lambda_2^s \cdot c_2)$ als durch

$$s_i = \frac{2\,\mathrm{Re}(\lambda_2^s \cdot c_2)}{|2\,\mathrm{Re}(\lambda_2^s \cdot c_2)|}$$

zu beschreiben. Günstigerweise ist $|2\,\mathrm{Re}(\lambda_2^s \cdot c_2)| < 1/4 \cdot c_1 \lambda_1^s$. Daher erscheint die Bezeichnung *Störungsterm* für den linksstehenden Ausdruck gerechtfertigt. Die numerischen Approximationen wurden mit dem Computeralgebrasystem PARI berechnet. □

Mit den Ungleichungen (3.24) des Korollars 3.9 kann nun die Ungleichung (3.9) trotz der Unschärfe verifiziert werden, denn mit den obigen Bezeichungen gilt

$$1.122 \cdot (\lambda_1 + 2)^s + |\epsilon(s + 1)| < 0.781 \cdot (\lambda_1 + 2)^{s+1} - |\epsilon(s + 1)|$$

für $s \in \mathbb{Z}^{>5}$.

Mit den gleichen Abschätzungen wird auch die Gleichung (3.1) des Satzes bewiesen. Leider sind die Abschätzungen (3.24) zu grob, um daraus die Abschätzungen (3.6) abzuleiten. Deshalb wird eine andere Methode verwendet, die von der umfangreichen Vorarbeit profitiert.

Zunächst wird die Folge $(\mu_{21}^{(s)})$ mit Hilfe der Folge (a_s) dargestellt.

Hilfssatz 3.10 *Es gelten die Bezeichnungen von Korollar 3.9. Dann ergibt sich die Folge $\mu_{21}^{(s)}$ zu*

$$\mu_{21}^{(s)} = \frac{1}{4}\left(1 + 4\,\frac{a_{s+1}}{a_s} - \frac{a_{s+3}}{a_s}\right) \quad \text{für } s \geq 0. \tag{3.25}$$

Die Ungleichungen (3.6) sind äquivalent zu der Gültigkeit der beiden Ungleichungen

$$m_1(s) = \quad a_s - 4\,a_{s+1} + a_{s+3} \quad \geq 0 \tag{3.26}$$

$$m_2(s) = \quad 3\,a_s + 4\,a_{s+1} - a_{s+3} \quad \geq 0. \tag{3.27}$$

Die in (3.27) und (3.27) definierten Folgen genügen der gleichen Rekursiongleichung wie die Folge (a_s) mit den folgenden Startvektoren:

Folge	*Startvektor*					
$m_1(s)$	(2 ,	8,	10,	34,	64,	130)
$m_2(s)$	(2 ,	0,	10,	2,	40,	34)

Tabelle 3.3: Startvektoren der Folgen zum Beweis von (3.6)

Die geschlossenen Darstellungen der Folgen $m_1(s)$ und $m_2(s)$ lauten:

$$m_i(s) = d_{i1}\,\lambda_1^s + 2\,Re(d_{i2}\,\lambda_2^s) + d_{i4}\,(\lambda_1 + 2)^s + 2\,Re(d_{i5}\,(\lambda_2 + 2)^s) \tag{3.28}$$

für $1 \leq i \leq 2$ und $s \in \mathbb{Z}^{\geq 0}$. Die Koeffizienten d_{ij} sind dabei definiert als

$$d_{1j} = \begin{cases} \frac{2}{59}\,(7\,\lambda_j - 1) & \text{für} \quad 1 \leq j \leq 2 \\ \frac{2}{59}\left(12\,\lambda_{j-3}^2 + 54\,\lambda_{j-3} + 70\right) & \text{für} \quad 3 < j \leq 5 \end{cases}$$

und

$$d_{2j} = \begin{cases} \frac{2}{59}\left(-4\,\lambda_j^2 - 9\,\lambda_1 + 11\right) & \text{für} \quad 1 \leq j \leq 2 \\ \frac{4}{59}\left(2\,\lambda_{j-3}^2 + 11\,\lambda_{j-3} + 13\right) & \text{für} \quad 3 < j \leq 5. \end{cases}$$

Es gilt $m_i(s) \geq 0$ für $1 \leq i \leq 2$ und alle $s \in \mathbb{Z}^{\geq 0}$.

Beweis: Die Gleichung (3.25) folgt nach kurzer Rechnung aus (3.4) und (3.5). Die Äquivalenz der Ungleichungen (3.6) mit den beiden Ungleichungen (3.27) und (3.27) folgt nach Gleichung (3.25). Dabei ist Ungleichung (3.27) äquivalent zu der Ungleichung $\mu_{21}^{(s)} \leq 1/2$ und die Ungleichung (3.27) ist äquivalent zu der Ungleichung $\mu_{21}^{(s)} \geq -1/2$. Die geschlossenen Formen (3.28) ergeben sich mit den angegebenen Startvektoren nach Hilfssatz 3.7. □

Das folgende Korollar verifiziert die Ungleichungen (3.6):

Hilfssatz 3.11 *Mit den Bezeichnungen aus Hilfssatz 3.10 gelten die Ungleichungen*

$$m_1(s) \geq 0 \textit{ und } m_2(s) \geq 0 \textit{ für alle } s \in \mathbb{Z}^{\geq 0}.$$

Beweis: Aus Tabelle 3.1 ist ersichtlich, daß die Terme in (3.28), die Potenzen von λ_1, (λ_2+2) und (λ_3+2) enthalten, exponentiell gegen Null konvergieren. Da, wie in Hilfssatz 3.8 gezeigt, $|\lambda_2| = (\lambda_1+2)$ und $-|z| \leq \mathrm{Re}(z) \leq |z|$ für alle $z \in \mathbb{C}$ gilt, ergeben sich die Abschätzungen

$$m_i(s) = 2\,\mathrm{Re}(d_{i2}\lambda_2^s) + d_{i4}\,(\lambda_1+2)^s + \epsilon_i(s) \geq (\lambda_1+2)^s\,(d_{i4} - 2|d_{i2}|) + \epsilon_i(s)$$

für $1 \leq i \leq 2$ mit $\epsilon_i(s) < 1$ und $s \in \mathbb{Z}^{\geq 0}$. Es gilt

$$|\lambda_2|\,(d_{14} - 2\,|d_{12}|) > 1.65 \cdot 2.2^s$$

sowie

$$|\lambda_2|\,(d_{24} - 2\,|d_{22}|) > 0.02 \cdot 2.2^s.$$

Damit ist der Hilfssatz gezeigt. □

Nach Hilfssatz 3.10 und Hilfssatz 3.11 sind die Ungleichungen (3.6) bewiesen. Wegen (3.9) und (3.11) ist damit auch (3.7) bewiesen.

Es ist noch die Richtigkeit von (3.8) und (3.10) nachzuweisen. Dazu wird die Folge $(q_s) \in \mathbb{Z}^{\mathbb{N}}$, wie in Hilfssatz 3.12 angegeben, definiert.

Hilfssatz 3.12 *Bezeichne*

$$q_s = p_{11}^{(s)}\,p_{33}^{(s)} - p_{13}^{(s)}\,p_{13}^{(s)} \textit{ für } s \in \mathbb{Z}^{\geq 0}.$$

Dann ist

$$q_{s+1} = p_{11}^{(s)}\,p_{22}^{(s)} - p_{12}^{(s)}\,p_{12}^{(s)} \textit{ für } s \in \mathbb{Z}^{\geq 0}.$$

Beweis: Es gilt nach (3.15)

$$p_{13}^{(s)} = -2\,a_s + p_{12}^{(s-1)} \tag{3.29}$$

und nach (3.5)

$$4\,p_{12}^{(s)} = a_s + 4\,a_{s+1} - a_{s+3}. \tag{3.30}$$

Nach Definition ist

$$q_{s+1} = p_{11}^{(s+1)}\,p_{33}^{(s+1)} - p_{13}^{(s+1)}\,p_{13}^{(s+1)}.$$

Damit ergibt sich nach (3.29)

$$q_{s+1} = p_{11}^{(s+1)}\,p_{33}^{(s+1)} - \left(2\,a_{s+1} + p_{12}^{(s)}\right)^2 = p_{11}^{(s+1)}\,p_{33}^{(s+1)} - 4\,a_{s+1}^2 - 4\,a_{s+1}\,p_{12}^{(s)} - p_{12}^{(s)}\,p_{12}^{(s)}.$$

Wegen (3.30) kann dies zu

$$q_{s+1} = a_{s+1}\,\left(a_{s+3} - 4\,a_{s+1} - 4\,p_{12}^{(s)}\right) - p_{12}^{(s)}\,p_{12}^{(s)} = a_{s+1}\,a_s - p_{12}^{(s)}\,p_{12}^{(s)} = p_{11}^{(s)}\,p_{22}^{(s)} - p_{12}^{(s)}\,p_{12}^{(s)}$$

umgeformt werden. □

Zum Beweis von (3.10) genügt es (Hilfssatz 3.12) zu zeigen, daß die Folge (q_s) streng monoton wächst. Die Beweisstrategie ist auch hier wieder die, eine Rekursion für die Folge (q_s) aufzustellen und zu lösen. Dabei zeigt sich, daß es sich auch hier wieder als sinnvoll erweist, die Rekursion zu beliebigen Startwerten zu lösen. Der Grund dafür ist, daß sowohl der Zähler als auch der Nenner von $\mu_{32}^{(s)}$ der gleichen Rekursion zu verschiedenen Startwerten genügt. Damit kann diese Vorarbeit eingesetzt werden, um die Ungleichungen (3.8) zu beweisen. Mit Hilfssatz 3.13 wird die Rekursion für die Folge (q_s) aufgestellt:

Hilfssatz 3.13 *Sei q_s wie in Hilfssatz 3.12 definiert und bezeichne*

$$z_s = p_{23}^{(s)}\, a_s - p_{12}^{(s)}\, p_{13}^{(s)}.$$

Die Folgen q_s und z_s genügen der gleichen Rekursion zu verschiedenen Startwerten. Die Rekursionsgleichung lautet:

$$q_{s+6} = (-1, -2, 4, 10, 2, 4) \cdot (q_s, q_{s+1}, \ldots, q_{s+5})^t. \tag{3.31}$$

Die Startvektoren zu den beiden Folgen sind in Tabelle 3.4 aufgeführt.

Folge	*Startvektor*					
z_s	$(-1$	$,-4,$	$-20,$	$-99,$	$-480,$	$-2332)$
q_s	$(\ 2$	$,9,$	$45,$	$218,$	$1057,$	$5145)$

Tabelle 3.4: Startvektoren zu den Folgen zum Beweis von (3.8) und (3.10)

Beweis: Die Rekursionsgleichung wurde mit dem Algorithmus von Berlekamp und Massey bestimmt, indem jeweils hinreichend viele Folgeglieder berechnet wurden, um die Rekursion zu erhalten. Die so gewonnen Rekursiongleichung wurde dann ähnlich wie in Hilfssatz 3.12 für die Folgen z_s und q_s verifiziert. □

Zur Lösung der Rekursionsgleichung (3.31) wird zunächst die erforderliche Notation eingeführt. Bezeichne $p_1(x) = x^3 + 2\,x - 1$ und $p_2(x) = x^3 - 4\,x^2 - 4\,x - 1$. Dann gilt für das charakteristische Polynom der die Rekursion (3.31) darstellenden Matrix

$$R = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ -1 & -2 & 4 & 10 & 2 & 4 \end{pmatrix} \tag{3.32}$$

$\mathrm{char}(R, x) = p_1(x) \cdot p_2(x)$. Es bezeichne λ_{1+3i} die reelle und $(\lambda_{2+3i}, \lambda_{3+3i})$ das Paar komplexer Nullstellen des Polynoms p_{i+1} für $0 \leq i \leq 1$. Naturgemäß gilt $\lambda_{2+3i} = \overline{\lambda_{3+3i}}$ für $0 \leq i \leq 1$.

Die numerischen Approximationen an die Wurzeln von $\mathrm{char}(R, x)$ wurden in Tabelle 3.5 angegeben.

Bezeichnung der Wurzel	Numerische Approximation
$\lambda_1 \in \mathbb{R}$	$0.45339 + \epsilon$
$\lambda_2 \in \mathbb{C} - \mathbb{R}$	$-(0.226 + \epsilon) + (1.467 + \epsilon)\, i$
$\lambda_3 = \overline{\lambda_2} \in \mathbb{C} - \mathbb{R}$	$-(0.226 + \epsilon) - (1.467 + \epsilon)\, i$
$\lambda_4 \in \mathbb{R}$	$4.864 + \epsilon$
$\lambda_5 \mathbb{C} - \mathbb{R}$	$-(0.432 + \epsilon) + (0.136 + \epsilon)\, i$
$\lambda_6 = \overline{\lambda_5} \in \mathbb{C} - \mathbb{R}$	$-(0.432 + \epsilon) - (0.136 + \epsilon)\, i$

Tabelle 3.5: Approximationen der Wurzeln des charakterischen Polynoms der Matrix R

Hierbei gilt $-10^3 < \epsilon < 10^3$. Die Wurzeln $\lambda_i\,(1 \leq i \leq 6)$ genügen gewissen Relationen, die im nachfolgenden Hilfssatz zusammengefaßt wurden. Diese Relationen werden benötigt, um zu möglichst einfachen Darstellungen für die in (3.34) und (3.35) definierten Koeffizienten c_{1i} und c_{2i} $(1 \leq i \leq 6)$ zu gelangen.

Hilfssatz 3.14 *Seien λ_i $(1 \leq i \leq 6)$ die Wurzeln des charakterischen Polynoms der Matrix R der Gleichung (3.32), die wie in Tabelle 3.5 definiert sind. Dann gelten die folgenden Relationen:*

$$\begin{array}{ll} \prod_{i=1}^{3} \lambda_i = 1 & \prod_{i=4}^{6} \lambda_i = 1 \\ \sum_{i=1}^{3} \lambda_i = 0 & \sum_{i=4}^{6} \lambda_i = 4 \end{array}$$

$$\lambda_1^2 = \lambda_5 \cdot \lambda_6$$
$$\lambda_4 = \lambda_2^2 \cdot \lambda_3^2$$

$$\lambda_4 \cdot \lambda_1^2 = 1 \quad \lambda_5 \cdot \lambda_2^2 = 1 \quad \lambda_6 \cdot \lambda_3^2 = 1$$

$$\lambda_1^2 + \lambda_2^2 + \lambda_3^2 = -4$$
$$\lambda_4^2 + \lambda_5^2 + \lambda_6^2 = 24$$

Beweis: Der Beweis kann auf mannigfache Weise geführt werden. Beispielsweise analog wie in Hilfssatz 3.8 oder durch Spezialisierung allgemeiner für Polynome gültiger Sätze. □

Es bezeichne

$$V = (\lambda_j^{i-1})_{1 \leq i,j \leq 6} \tag{3.33}$$

die aus den Wurzeln λ_i gebildete 6×6 Vandermonde Matrix. Analog zu Hilfssatz 3.6 gilt

$$z_s = \sum_{i=1}^{6} c_{1i} \lambda_i^s \text{ für } s \in \mathbb{Z}^{\geq 0} \tag{3.34}$$

und

$$q_s = \sum_{i=1}^{6} c_{2i} \lambda_i^s \text{ für } s \in \mathbb{Z}^{\geq 0}. \tag{3.35}$$

Hierbei sind c_{1i} und c_{2i} für $1 \leq i \leq 6$ eindeutig aus den folgenden linearen inhomogenen Gleichungssystemen (3.34) und (3.35) zu bestimmen.

$$V \cdot \begin{pmatrix} c_{11} \\ c_{12} \\ c_{13} \\ c_{14} \\ c_{15} \\ c_{16} \end{pmatrix} = \begin{pmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \\ z_5 \\ z_6 \end{pmatrix} \tag{3.36}$$

$$V \cdot \begin{pmatrix} c_{21} \\ c_{22} \\ c_{23} \\ c_{24} \\ c_{25} \\ c_{26} \end{pmatrix} = \begin{pmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \\ q_5 \\ q_6 \end{pmatrix} \tag{3.37}$$

Hilfssatz 3.15 *Seien λ_i $(1 \leq i \leq 6)$ wie in Hilfssatz 3.14 angegeben. Bezeichne $\mathbf{e}_l$ den l-ten kanonischen Einheitsvektor. Sei V wie in (3.33) definiert. Dann gilt für die eindeutig bestimmten Lösungen $\mathbf{c}_l$ des linearen inhomogenen Gleichungssystems: $V \cdot \mathbf{c}_l = \mathbf{e}_l$ für $1 \leq l \leq 6$:*

$$\mathbf{c}_1 = \frac{1}{118} \begin{pmatrix} 3\lambda_1^2 + 4\lambda_4 + 8 \\ 3\lambda_5^2 - 8\lambda_5 - 4 \\ 3\lambda_6^2 - 8\lambda_6 - 4 \\ 109\lambda_1^2 - 56\lambda_4 + 250 \\ 109\lambda_5^2 - 492\lambda_5 - 186 \\ 109\lambda_6^2 - 492\lambda_6 - 186 \end{pmatrix}$$

$$\mathbf{c}_2 = \frac{1}{118} \begin{pmatrix} 8\lambda_1^2 + 19\lambda_4 + 24 \\ 8\lambda_5^2 - 13\lambda_5 - 8 \\ 8\lambda_6^2 - 13\lambda_6 - 8 \\ 32\lambda_1^2 - 3\lambda_4 + 8 \\ 32\lambda_5^2 - 131\lambda_5 - 120 \\ 32\lambda_6^2 - 131\lambda_6 - 120 \end{pmatrix}$$

$$\mathbf{c}_3 = \frac{1}{118} \begin{pmatrix} -4\lambda_1^2 + 30\lambda_4 + 3 \\ -4\lambda_5^2 + 46\lambda_5 + 19 \\ -4\lambda_6^2 + 46\lambda_6 + 19 \\ -556\lambda_1^2 + 256\lambda_4 - 1131 \\ -556\lambda_5^2 + 2480\lambda_5 + 1093 \\ -556\lambda_6^2 + 2480\lambda_6 + 1093 \end{pmatrix}$$

$$\mathbf{c}_4 = \frac{1}{118}\begin{pmatrix} -19\lambda_1^2 + 16\lambda_4 - 28 \\ -19\lambda_5^2 + 92\lambda_5 + 48 \\ -19\lambda_6^2 + 92\lambda_6 + 48 \\ 3\lambda_1^2 + 4\lambda_4 - 20 \\ 3\lambda_5^2 - 8\lambda_5 - 32 \\ 3\lambda_6^2 - 8\lambda_6 - 32 \end{pmatrix}$$

$$\mathbf{c}_5 = \frac{1}{118}\begin{pmatrix} 4\lambda_1^2 + 5\lambda_4 + 20 \\ 4\lambda_5^2 - 11\lambda_5 + 4 \\ 4\lambda_6^2 - 11\lambda_6 + 4 \\ -250\lambda_1^2 + 115\lambda_4 - 508 \\ -250\lambda_5^2 + 1115\lambda_5 + 492 \\ -250\lambda_6^2 + 1115\lambda_6 + 492 \end{pmatrix}$$

$$\mathbf{c}_6 = \frac{1}{118}\begin{pmatrix} -2\lambda_4 - 3 \\ -2\lambda_5 - 3 \\ -2\lambda_6 - 3 \\ 56\lambda_1^2 - 26\lambda_4 + 115 \\ 56\lambda_5^2 - 250\lambda_5 - 109 \\ 56\lambda_6^2 - 250\lambda_6 - 109 \end{pmatrix}$$

Korollar 3.16 *Seien λ_i $(1 \leq i \leq 6)$ wie in Hilfssatz 3.14 definiert. Dann gilt für die Koeffizienten aus den Gleichungen (3.34) und (3.35)*

$$\begin{pmatrix} c_{11} \\ c_{12} \\ c_{13} \\ c_{14} \\ c_{15} \\ c_{16} \end{pmatrix} = \frac{1}{59}\begin{pmatrix} 3\lambda_1^2 + 2 \\ 3\lambda_5^2 - 12\lambda_5 - 10 \\ 3\lambda_6^2 - 12\lambda_6 - 10 \\ -3\lambda_1^2 - 8\lambda_4 - 11 \\ -3\lambda_5^2 + 4\lambda_5 + 1 \\ -3\lambda_6^2 + 4\lambda_6 + 1 \end{pmatrix}$$

und

$$\begin{pmatrix} c_{21} \\ c_{22} \\ c_{23} \\ c_{24} \\ c_{25} \\ c_{26} \end{pmatrix} = \frac{1}{59}\begin{pmatrix} -8\lambda_1^2 + 6\lambda_4 - 16 \\ -8\lambda_5^2 + 38\lambda_5 + 16 \\ -8\lambda_6^2 + 38\lambda_6 + 16 \\ 5\lambda_1^2 + 19\lambda_4 + 18 \\ 5\lambda_5^2 - \lambda_5 - 2 \\ 5\lambda_6^2 - \lambda_6 - 2 \end{pmatrix}.$$

Beweis: Die Koeffizienten c_{1i} und c_{2i} ergeben sich nach Hilfssatz 3.15 mit

$$(c_{11}, c_{12}, \ldots, c_{16})^t = \sum_{i=1}^{6} z_{i-1}\mathbf{c}_i{}^t$$

und

$$(c_{21}, c_{22}, \ldots, c_{26})^t = \sum_{i=1}^{6} q_{i-1}\mathbf{c}_i{}^t.$$

□

Nach der umfangreichen Vorarbeit gestaltet sich der Beweis von (3.8) und (3.10) sehr einfach. Um (3.10) zu beweisen wird gezeigt, daß die Folge $(q_s) \in \mathbb{Z}^{\mathbb{N}}$ streng monoton wächst. Aus Tabelle 3.4 ist ersichtlich, welche Terme in der Darstellung (3.35) exponentiell wachsen und welche Terme exponentiell gegen Null konvergieren. Terme, die Potenzen von λ_1, λ_5 und λ_6 enthalten, konvergieren exponentiell gegen Null, wohingegen Terme mit Potenzen von λ_2, λ_3 und λ_4 mit exponentieller Geschwindigkeit wachsen. Nach Hilfssatz 3.14 ist $\lambda_4 = \lambda_2^2 \cdot \lambda_3^2$, also $\lambda_4 = |\lambda_2|^2 = |\lambda_3|^2$. Ferner ist, wie bereits erwähnt $-|z| \leq \mathrm{Re}(z) \leq |z|$ für alle $z \in \mathbb{C}$.

Für exponentiell gegen Null konvergierender Fehlerfunktion $\epsilon(s)$ mit $|\epsilon(s)| < 2^{-s}$ für $s \in \mathbb{Z}^{\geq 0}$ gilt die Gleichung

$$q_s = c_{24} \cdot \lambda_4^2 + 2\mathrm{Re}(c_{22} \cdot \lambda_2^s) + \epsilon(s)$$

und damit

$$\lambda_4^s \cdot (c_{24} - 2\,|c_{22}|\lambda_4^{-s/2}) - |\epsilon(s)| \leq q_s \leq \lambda_4^s \cdot (c_{24} + 2\,|c_{22}|\lambda_4^{-s/2}) + |\epsilon(s)|.$$

Mit Hilfe dieser Ungleichungen wird $q_s < q_{s+1}$ für alle $s \in \mathbb{Z}^{\geq 0}$ mit den numerischen Approximation $c_{24} \approx 1.889$ und $|c_2| \approx 0.108$ verifiziert. Damit ist (3.10) gezeigt.

Der Beweis der Ungleichung (3.8) erweist sich als sehr interessant, da sich herausstellt, daß die Folge $\mu_{32}^{(s)}$ sogar einen Grenzwert hat, der die reelle Wurzel des Polynoms $x^3 + 2\,x + 1$ ist. Dieses Polynom wurde mit der bekannten Methode [16], Seite 525, zur Bestimmung eines Polynoms $p \in \mathbb{Z}[x]$ mit $p(\lim_{s\to\infty} \mu_{32}^{(s)}) = 0$ gefunden. Diese Methode ist eine der vielen Anwendungen des LLL-Algorithmus.

Aus den Darstellungen (3.34) und (3.35) mit den Koeffizieten c_{1i} und c_{2i} für $1 \leq i \leq 6$ aus Korollar 3.16 sowie den numerischen Approximationen aus Tabelle 3.4 ergeben sich folgende Abschätzungen, wobei $\epsilon(s)$ mit $|\epsilon(s)| < 2^{-s}$ für $s \in \mathbb{Z}^{s\geq 0}$ eine Fehlerfunktion ist:

$$\frac{\lambda_4^2(c_{14} - 2\,\lambda_4^{-s/2}\,|c_{12}|) - |\epsilon(s)|}{\lambda_4^2(c_{24} + 2\,\lambda_4^{-s/2}\,|c_{22}|) + |\epsilon(s)|} \leq \mu_{32}^{(s)} \leq \frac{\lambda_4^2(c_{14} + 2\,\lambda_4^{-s/2}\,|c_{12}|) + |\epsilon(s)|}{\lambda_4^2(c_{24} - 2\,\lambda_4^{-s/2}\,|c_{22}|) - |\epsilon(s)|}$$

Aus dieser Ungleichung liest man den Grenzwert der Folge ab:

$$\lim_{s\to\infty} \mu_{32}^{(s)} = \frac{c_{14}}{c_{24}} \tag{3.38}$$

Unter Verwendung von Hilfssatz 3.14 kann dies wie folgt umgeformt werden:

$$\frac{c_{14}}{c_{24}} = \frac{-3\lambda_1^2 - 8\lambda_4 - 11}{5\lambda_1^2 + 19\lambda_4 + 18} = -\lambda_1 \tag{3.39}$$

Damit ist auch (3.8) verifiziert. Dies komplettiert den Beweis von Lemma 3.3.

3.4 Konstruktion parametrisierter Eingaben für beliebige Dimensionen

Es ist immer noch ein offenes Problem, ob der LLL-Algorithmus für die Wahl des Parameters $y = 1$ polynomiell in n und der Eingabegröße $E(A)$ ist. Wir wollen einen neuen Beitrag zu diesem Problemkreis leisten, indem wir unsere parametrisierten Eingaben dazu verwenden neue parametrisierte Eingaben für beliebige Dimension n zu kreieren.

Satz 3.17 *Der LLL-Algorithmus benötigt auf Eingabe der Matrix*

$$B_l = \begin{pmatrix} \mathbf{0} & \boxed{A_l} \\ I_{n-3} & \mathbf{0} \end{pmatrix} \in \mathbb{Z}^{n \times n}, l \geq 3 \ \mathit{mit}$$

$$A_l = \begin{pmatrix} 0 & -1 & 1 \\ 1 & 0 & -2 \\ 0 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & -2 \\ 0 & 1 & 0 \end{pmatrix}^l,$$

wenigsten $T(B_l, 1) = (2n-3)l$ Gaußschritte.

Beweis: Wir wählen die Bezeichnungweisen $\mathbf{b}_1' = \mathbf{b}_{n-2}, \mathbf{b}_2' = \mathbf{b}_{n-1}, \mathbf{b}_3' = \mathbf{b}_n$. Mit diesen Bezeichnungen wurde in den vorigen Abschnitten bereits gezeigt, daß $LLL(y=1)$ bei Eingabe der Matrix A_l genau $3l + 5$ Schritte benötigt. Für $n = 3$ ist die Aussage also richtig. Für die weiteren Erklärungen, die sich auf die Tabelle des vorigen Abschnitts beziehen, ist $\mathbf{b}_i$ $(1 \leq i \leq 3)$ in der Tabelle durch die hier definierten Bezeichnugen $\mathbf{b}_i'$ $(1 \leq i \leq 3)$ mutatis mutandis zu ersetzen. Sei nun $n > 3$. Da im linken Teil der Matrix A_l eine Minkowski-reduzierte Matrix steht, ist die Bedingung in Anweisung (7) des LLL-Algorithmus für alle Masterindizes $k \leq n-2$ verletzt. Das bedeutet aber, da die zugehörigen μ's verschwinden, daß sich die Matrix nicht verändert. Wir schildern jetzt, ausgehend vom Masterindex $k = n-2$, l identische Abläufe, die jeweils $2n-3$ Schritte (Aufrufe des Gaußstep-Algorithmus) umfassen. Anders formuliert: Wir erklären einen Ablauf, der $(2n-3)l$ Schritte umfaßt und periodisch ist mit Periodenlänge $(2n-3)$. Der Algorithmus $LLL(y = 1)$ produziert dabei aus einer Eingabe A_l in $2n-3$ Schritten eine Ausgabe $B_{l-1}, l \geq 1$. Sei also $k = n-2$. Jetzt wird Anweisung (6) des LLL-Algorithmus einmal und Anweisung (10) $(n-4)$-mal ausgeführt. Es hat also insgesamt $t_1 = (n-3)$ Aufrufe von Gaußstep-LLL(k,l) gegeben, nämlich $k = n-2, l \in \{1, \ldots, n-3\}$. Jetzt ist $k = n-1$ und es wird wieder Anweisung (6) einmal ausgeführt und Anweisung (10) $n-3$ mal, da die Bedingung (7) wieder verletzt ist (Zeile 6 der Tabelle). Insgesamt ergeben sich also $t_2 = n-2$ Gaußstep(k,l) Aufrufe: $k = n-2$, $l \in \{1, \ldots, n-2\}$. Jetzt ist $k = n$ und die Anweisung (6) wird einmal ausgeführt, ohne die Basis zu verändern (Zeile 12 der Tabelle 3.1). Die Bedingung von Abfrage (7) ist diesmal erfüllt (Zeile 13 der Tabelle 3.1), was den Tausch von $\mathbf{b}_2'$ und $\mathbf{b}_3'$ zur Folge hat. Außerdem ist jetzt $k = n-1$. Anweisung (6) bewirkt nun, daß der Basisvektor $\mathbf{b}_2' = \mathbf{b}_{n-1}$ zu $\mathbf{b}_3' + 2\mathbf{b}_1'$ wird (Zeile 17 der Tabelle). Die Bedingung in Anweisung (7) ist wieder erfüllt (Zeile 18 der Tabelle) und der update-Aufruf hat zur Folge, daß insgesamt A_l zu A_{l-1} reduziert wurde (Zeile 19 der Tabelle 3.1). Der Masterindex ist jetzt wieder $k = n-2$. Ein Ablauf von insgesamt l Zyklen ist also komplett.

Wir brauchen somit nur noch die Summe der Gaußstep Aufrufe zu bilden. Es waren: $t_1 = n-3$ Aufrufe zum Masterindex $k = n-2$, $t_2 = n-2$ Aufrufe zum Masterindex $k = n-2$ und noch je eine Aufruf in Anweisung (6). Also insgesamt $2n-3$ Aufrufe. $\square$

Kapitel 4

Algorithmen zur Minkowski-Reduktion - ein Überblick

4.1 Bekannte Algorithmen zur Minkowski-Reduktion

Das Problem der Berechnung einer Basis $B = (\mathbf{b}_1, \ldots, \mathbf{b}_n) \in \mathbb{Z}^{n \times n}$ eines Gitters $L \subseteq \mathbb{Z}^n$, mit der Eigenschaft

$$H(B) = \min\{H(B') : \ B' \text{ ist Basis von } L\},$$

ist NP-hart.

Eine Minkowski-reduzierte Basis eines Gitters ist eine minimale Basis dieses Gitters.

Das beste bekannte Komplexitätsresultat zur Minkowski-Reduktion mit dem Algorithmus M-RED von Bettina Helfrich [11] wird in Satz 4.1 zitiert.

Satz 4.1 M-RED *berechnet bei Eingabe einer regulären Matrix* $A = (\mathbf{a}_1, \ldots, \mathbf{a}_n) \in \mathbb{Z}^{d \times n}$ *ein Matrix* B *mit der Eigenschaft, daß* B *eine Minkowski-reduzierte Basis des Gitters* $L(A)$ *ist. Der Algorithmus* M-RED *benötigt dazu*

$$O\left((5/4)^{n^3(4-o(1))} \cdot d \cdot \log(E(A))\right)$$

arithmetische Operationen auf ganzen Zahlen der Länge

$$O(n^4(n \log(n) + \log(E(A)))).$$

Für feste Dimension n ergeben sich also Kosten von

$$O(\log(E(A))^3)$$

Bitoperationen bei Verwendung von klassischen Algorithmen für die elementaren Operationen $+, -, =, *$ und $/$. Bei Anwendung schneller Multiplikationstechniken (Schönhage, Strassen [31]) kann die genannte obere Schranke zu

$$O(\log(E(A))^{2+\epsilon})$$

verbessert werden.

Bettina Helfrichs Algorithmus zur Minkowski-Reduktion benutzt eine verbesserte Version eines Algorithmus von Ravi Kannan [28] zur Korkine-Zolotarev oder Hermite-Reduktion. Der letztgenannte Algorithmus wurde von Claus Peter Schnorr verbessert [29].

Eine interessante Fragestellung ist nun, ob sich das genannte Komplexitätsresulat für feste Dimension signifikant verbessern läßt. In ihrer Habilitationsschrift [35] untersucht Brigitte Vallée einen Algorithmus, der für Gitter $L \subseteq \mathbb{Z}^3$ eine Minkowski-reduzierte Basis direkt, also ohne ein zusätzliches Enumerationsverfahren, berechnet. Sie nennt diesen Algorithmus "Acute-Algorithmus". Die Analyse benötigt viele geometrische Resultate, die nur im dreidimensionalen Raum gelten. Die im Algorithmus verwendeten Operationen sind im wesentlichen die Operationen des LLL-Algorithmus, mit dem Unterschied, daß in jedem Schritt eine vollständige LLL-Neuinitialisierung und den damit verbundenen großen Kosten erforderlich ist. Der Algorithmus ist so konzipiert, daß sich die euklidischen Längen der Vektoren in jedem Schritt um den Faktor $\sqrt{7/8}$ verkürzen. Der "Acute-Algorithmus" hat aber dennoch keine bessere asymptotische Komplexität als der LLL-Algorithmus beziehungsweise der M-RED-Algorithmus. Die Komplexität des "Acute-Algorithmus" [35] ist

$$O(\log(E(A))^3)$$

Bitoperationen.

Wir präsentieren in Kapitel 5 zwei neue exakte Algorithmen zur Minkowskireduktion. Den ersten Algorithmus haben wir TPR-Algorithmus (Totaler-Paar-Reduktions-Algorithmus) genannt, da er auf dem *Paar-Reduktionskonzept* beruht. Dieser Algorithmus berechnet eine Minkowski-reduzierte Basis eines Gitters $L \subseteq L^d$ $(d \geq 3)$ vom Rang drei in

$$O(\log(E(A))^3)$$

Bitoperationen. Der Algorithmus hat eine wesentlich bessere Laufzeitkonstante als der "Acute-Algorithmus" und beinhaltet eine einfache Möglichkeit der Verallgemeinerung auf Dimension vier. Der zweite Algorithmus FTPR, für **Fast** TPR, leistet das Gleiche wie der TPR-Algorithmus, besitzt aber eine signifikant bessere asymtotische Komplexität von

$$\Theta(\log(E(A))^2)$$

Bitoperationen. Wir beweisen diese Komplexitätsresultate unter Verwendung von klassischen Algorithmen für die elementaren Operationen.

4.2 Kritik am Konzept, den LLL-Algorithmus zur Minkowski-Reduktion zu verwenden

Der Algorithmus von Ravi Kannan zur Hermite-Reduktion, der durch Claus Peter Schnorr verbessert wurde, benutzt den LLL-Algorithmus zur Basisreduktion, um eine Menge "moderater Größe" in Polynomialzeit zu berechnen, in der ein kürzester Vektor in

$$O(n^n) \text{(Ravi Kannan [28])}$$

beziehungsweise

$$O(\sqrt{n^n})\,(\text{ Claus Peter Schnorr [29]})$$

Enumerationsschritten berechnet werden kann.

Der Algorithmus von Bettina Helfrich benutzt rekursiv diesen Algorithmus zur Hermite-Reduktion, um eine Minkowski-Reduktion zu gewinnen. Für feste Dimension n ist der Algorithmus zwar polynomiell, enthält aber eine gigantische Konstante in der Laufzeitschranke.

Die Hauptkritikpunkte an diesem Verfahren sind:

1) Die Verwendung von Enumeration kann nicht optimal sein. Enumeration führt im allgemeinen zu exponentiellen Laufzeitschranken

2) Der als Subroutine verwendete LLL-Algorithmus hat den gravierenden Nachteil, daß er die "guten geometrischen" Eigenschaften einer stark reduzierten Basis eines Gitters vom Rang $n > 2$ wieder zunichte machen kann, wie am Beispiel 4.2 illustriert wird.

Beispiel 4.2 ("Verlängerung" durch den LLL-Algorithmus) *Der LLL(A, y)-Algorithmus berechnet für die in (4.1) definierte Matrix A für alle $y \in (1/4, 1]$ die in (4.2) definierte Matrix B.*

$$A = \begin{pmatrix} 17 & 8 & 0 \\ 0 & 15 & -8 \\ 0 & 0 & 14 \end{pmatrix} \tag{4.1}$$

$$B = \begin{pmatrix} 17 & 8 & 8 \\ 0 & 15 & 7 \\ 0 & 0 & 14 \end{pmatrix} \tag{4.2}$$

In Tabelle 4.1 sind die quadrierten euklidischen Längen der Spaltenvektoren der Matrizen A und B gegenübergestellt.

	1. Spalte	*2. Spalte*	*3. Spalte*
Eingabe: A	*289*	*289*	*289*
Ausgabe: B	*289*	*289*	*309*

Tabelle 4.1: "Verlängerung" von Vektoren durch LLL-Reduktion

Kapitel 5

Der TPR-Algorithmus zur Minkowski-Reduktion

In diesem Kapitel wird ein neuer deterministischer Algorithmus vorgestellt, der eine Matrix A, die Basis eines Gitters vom Rang drei ist, in

$$O(\log(E(A))^3)$$

Bitoperationen Minkowski-reduziert. Die Analyse des Algorithmus ergab eine Möglichkeit, den Algorithmus so zu modifizieren, daß für diesen verbesserten Algorithmus ein signifikant besseres asymptotisches Komplexitätsresultat, das in Kapitel 6 präsentiert wird, von

$$O(\log(E(A))^2)$$

Bitoperationen bewiesen werden konnte.

Sei im folgenden immer $d \geq 3$ und bezeichne

$$A = (\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3) \in \mathbb{Z}^{d\times 3} \tag{5.1}$$

mit

$$A^t \cdot A \in \mathrm{GL}(3, \mathbb{Q}) \quad \text{und} \quad ||\mathbf{a}_1|| \leq ||\mathbf{a}_2|| \leq ||\mathbf{a}_3|| \tag{5.2}$$

eine Eingabe des TPR-Algorithmus. Jede Matrix A, welche die Bedingungen (5.1) und (5.2) erfüllt, heißt *eine zulässige Eingabe* des TPR-Algorithmus.

Die Abkürzung "TPR" steht für *total-Paar-reduziert* oder auch *tripel-Paar-reduziert*. Der Grund dafür ist, daß der TPR-Algorithmus die Eingabematrix A im ersten Teil so lange durch unimodulare Operationen transformiert, bis man eine Matrix $B = (\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3)$ erhält, welche die Eigenschaft besitzt, daß alle Paare von Spaltenvektoren $(\mathbf{b}_1, \mathbf{b}_2)$, $(\mathbf{b}_1, \mathbf{b}_3)$ und $(\mathbf{b}_2, \mathbf{b}_3)$ Gauß-reduziert sind.

5.1 Beschreibung des TPR-Algorithmus

Der TPR-Algorithmus ist ein deterministischer Algorithmus zur Reduktion einer Basis eines Gitters. Er kann aber auch als Algorithmus zur Reduktion einer positiv definiten ternären

Form verwendet werden. Aus der Eingabe A werden zunächst die Koeffizienten

$$p_{ij} \in \mathbb{Z}\,(1 \leq i \leq j \leq 3)$$

der positiv definiten ternären Form

$$q(x_1, x_2, x_3) = (x_1, x_2, x_3) \cdot A^t A \cdot (x_1, x_2, x_3)^t = \sum_{1 \leq i \leq j \leq 3} p_{ij} x_i x_j$$

berechntet. Der Grund dafür ist, die im Algorithmus auftretenden Skalarprodukte möglichst günstig zu berechnen. Zu Beginn ist die "teuere" Operation

$$p_{ij} = \langle \mathbf{a}_i, \mathbf{a}_j \rangle$$

mit

$$O(d \cdot \log(2||\mathbf{a}_3||))$$

Bitoperationen für jedes Indexpaar (i, j) mit $i, j \in \{1, 2, 3\}$ und $i < j$ auszuführen. Bei der Abarbeitung einer Anweisung

$$\mathbf{a}'_j = \mathbf{a}_j + r\,\mathbf{a}_i \text{ mit } r \in \mathbb{Z}$$

benötigt die Bestimmung von

$$p'_{ij} = \langle \mathbf{a}'_j, \mathbf{a}_i \rangle,$$

falls die $p_{lk}\,(1 \leq l, k \leq 3)$ bekannt sind, nur

$$O(\log(2||\mathbf{a}_3||))$$

Bitoperationen, wohingegen die Berechnung aus den Vektoren

$$O(d \cdot \log(2||\mathbf{a}_3||))$$

Bitoperationen kosten würde. Dies ist allerdings nur von praktischer Relevanz und ändert nichts an der asymptotischen Komplexität, da bei der Initialisierung in Anweisung (1) des TPR-Algorithmus $O(d \cdot ||\mathbf{a}_3||^2)$ Bitoperationen verwendet werden. Wird nur die Reduktion einer ternären Form gewünscht, so entfällt dieser Schritt. Die Vektoroperationen (4) der **Gauß**−Aufrufe werden dann nicht ausgeführt. Dadurch verschwindet der Faktor d im Komplexitätsresultat.

Der TPR-Algorithmus enthält drei Gaußalgorithmen $\mathbf{Gauß}_1$, $\mathbf{Gauß}_2$ und $\mathbf{Gauß}_3$ zur Reduktion positiv definiter binärquadratischer Formen. Dabei reduziert der Subalgorithmus $\mathbf{Gauß}_l$ die zu den Spalten $(i, j)\,(1 \leq i < j \leq 3)$ gehörende positiv definite binärquadratische Form

$$q_l = (x_1, x_2) \cdot \begin{pmatrix} \langle a_i, a_i \rangle & \langle a_i, a_j \rangle \\ \langle a_i, a_j \rangle & \langle a_j, a_j \rangle \end{pmatrix} \cdot (x_1, x_2)^t$$

mit $l = i + j - 2$. Die damit assoziierten unimodularen Transformationen werden zu jedem Reduktionsschritt der Form auf den Spalten (i, j) der Matrix $(\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3)$ ausgeführt. Diese Größen sind progammiertechnisch gesehen globale Variablen. Die Subreduktionsalgorithmen

Gauß$_l$ ($1 \leq l \leq 3$) unterscheiden sich vom Standard-Gaußalgorithmus zur Gitterbasisreduktion dadurch, daß sie **alle** durch den Reduktionsschritt (4)[1] betroffenen Skalarprodukte p_{ij} aktualisieren. Die Aktualisierung wird in den Operationen (8), (9) und (10) ausgeführt. Damit gilt, wenn B die gerade durch Reduktion entstandene Matrix bezeichnet, stets

$$B^t \cdot B = (p_{ij})_{1 \leq i,j \leq 3}.$$

Im folgenden wird das Struktogramm des TPR-Algorithmus erläutert.

Algorithmus 5.1 (TPR(A))

Eingabe: Eine zulässige Matrix $A = (\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3)$, die Basis eines Gitters L vom Rang drei ist.
Ausgabe: Eine Minkowski-reduzierte Basis des Gitters L.

Initialisiere $p_{ij} = \langle \mathbf{a}_i, \mathbf{a}_j \rangle$ und $T = t_i = \hat{t}_i = 0$ für $1 \leq i \leq j \leq 3$			(1)
	Gauß$_1$		(2)
	Gauß$_2$		(3)
	IF $t_2 > 1$		(4)
	THEN	swap(2, 3)	(5)
		Gauß$_2$	(6)
		UNTIL $t_2 \leq 1$	(7)
	IF $p_{33} < p_{22}$		(8)
	THEN	swap(2,3)	(9)
	Gauß$_3$		(10)
	Sort		(11)
UNTIL $2 \cdot \lvert p_{12} \rvert \leq p_{11}$ und $2 \cdot \lvert p_{13} \rvert \leq p_{11}$ und $2 \cdot \lvert p_{23} \rvert \leq p_{22}$			(12)
LastStep			(13)

Die Prozedur swap(i, j) vertauscht die Spalten i und j der Matrix, die aus der Eingabematrix berechnet wurde. Ferner aktualisiert swap(i, j) die globalen Variablen $p_{i',j'}$, die aktualisiert werden müssen. Die zu diesem Aktualisierungsvorgang erforderlichen Operationen werden in Tabelle 5.1 angegeben. Dabei werden jeweils vier der sechs globalen Variablen p_{ij} ($1 \leq i \leq j \leq 3$) den temporären Variablen (h_1, h_2, h_3, h_4), nach dem in der Tabelle gegeben Schema, zugewiesen und dann entsprechen vertauscht zurückgeschrieben.

[1] diese Anweisungslabel beziehen sich auf die Algorithmen 5.2, 5.3 und 5.4

swap$(1,2)$		swap$(1,3)$		swap$(2,3)$	
1. Operation	2. Operation	1. Operation	2. Operation	1. Operation	2. Operation
$h_1 = p_{11}$	$p_{11} = h_2$	$h_1 = p_{11}$	$p_{11} = h_2$		
$h_2 = p_{22}$	$p_{22} = h_1$			$h_2 = p_{22}$	$p_{22} = h_3$
		$h_2 = p_{33}$	$p_{33} = h_1$	$h_3 = p_{33}$	$p_{33} = h_2$
		$h_4 = p_{12}$	$p_{12} = h_6$	$h_4 = p_{12}$	$p_{12} = h_5$
$h_3 = p_{13}$	$p_{13} = h_4$			$h_5 = p_{13}$	$p_{13} = h_4$
$h_4 = p_{23}$	$p_{23} = h_3$	$h_6 = p_{23}$	$p_{23} = h_4$		

Tabelle 5.1: Schema der swap Operationen

Es folgen die Struktogramme der Subalgorithmen **Gauß**$_l$ $(1 \leq l \leq 3)$.

Algorithmus 5.2 (Gauß$_1$)

$t_1 = 0$	(1)
SWAP$(1, 2)$	(2)
$r = -\lfloor p_{12}/p_{11} \rceil$	(3)
$\mathbf{a}_2 = \mathbf{a}_2 + r \cdot \mathbf{a}_1$	(4)
$\hat{t}_1 = \hat{t}_1 + 1,\ T = T + 1,\ t_1 = t_1 + 1$	(5)
$h_1 = p_{12} + r \cdot p_{11}$	(6)
$h_2 = p_{23} + r \cdot p_{12}$	(7)
$p_{22} = p_{22} + 2 \cdot r \cdot p_{12} + r^2 \cdot p_{11}$	(8)
$p_{12} = h_1$	(9)
$p_{13} = h_2$	(10)
UNTIL $p_{22} \geq p_{11}$	(11)

Algorithmus 5.3 (Gauß$_2$)

$t_2 = 0$	(1)
SWAP(1, 3)	(2)
$r = -\lfloor p_{13}/p_{11} \rceil$	(3)
$\mathbf{a}_3 = \mathbf{a}_3 + r \cdot \mathbf{a}_1$	(4)
$\hat{t}_2 = \hat{t}_2 + 1, T = T + 1, t_2 = t_2 + 1$	(5)
$h_1 = p_{13} + r \cdot p_{11}$	(6)
$h_2 = p_{23} + r \cdot p_{12}$	(7)
$p_{33} = p_{33} + 2 \cdot r \cdot p_{13} + r^2 \cdot p_{11}$	(8)
$p_{13} = h_1$	(9)
$p_{23} = h_2$	(10)
UNTIL $p_{33} \geq p_{11}$	(11)

Die Prozedur **SWAP**(i, j) ruft in jedem **außer** dem ersten Schleifendurchlauf die Prozedur swap(i,j) auf. Die Anweisungen in Anweisungsblock (1) und (5) in den Algorithmen 5.2 und 5.4 dienen lediglich einer einfachen Bezeichnung bei der Analyse des TPR-Algorithmus. Es enthalten $\hat{t}_i \,(1 \leq i \leq 3)$ die Gesamtzahl aller im **Gauß**$_i$–Subalgorithmus ausgeführten Gaußschritte und T die Gesamtzahl aller Gaußschritte des TPR-Algorithmus.

Algorithmus 5.4 (Gauß$_3$)

$t_3 = 0$	(1)
SWAP(2, 3)	(2)
$r = -\lfloor p_{23}/p_{22} \rceil$	(3)
$\mathbf{a}_3 = \mathbf{a}_3 + r \cdot \mathbf{a}_2$	(4)
$\hat{t}_3 = \hat{t}_3 + 1, T = T + 1, t_3 = t_3 + 1$	(5)
$h_1 = p_{23} + r \cdot p_{22}$	(6)
$h_2 = p_{13} + r \cdot p_{12}$	(7)
$p_{33} = p_{33} + 2 \cdot r \cdot p_{23} + r^2 \cdot p_{22}$	(8)
$p_{23} = h_1$	(9)
$p_{13} = h_2$	(10)
UNTIL $p_{33} \geq p_{22}$	(11)

Die Struktogramme des "Sortieralgorithmus" **Sort** und der letzten Anweisung **LastStep** komplettieren die Beschreibung des TPR-Algorithmus.

Algorithmus 5.5 (Sort)

SORT$(1,2)$	(1)
SORT$(2,3)$	(2)
SORT$(1,2)$	(3)

Algorithmus 5.6 (SORT(i,j))

IF	$p_{ii} > p_{jj}$	(1)
THEN	swap(i, j)	(2)

Algorithmus 5.7 (LastStep)

<table>
<tr><td colspan="2">$h = p_{11} + p_{22} + p_{33}$</td><td>(1)</td></tr>
<tr><td colspan="2">
<table>
<tr><th colspan="3">Vorzeichen</th><th>Operation</th></tr>
<tr><td>p_{12}</td><td>p_{13}</td><td>p_{23}</td><td></td></tr>
<tr><td>$+$</td><td>$+$</td><td>$+$</td><td>keine</td></tr>
<tr><td>$-$</td><td>$-$</td><td>$+$</td><td>keine</td></tr>
<tr><td>$-$</td><td>$+$</td><td>$-$</td><td>keine</td></tr>
<tr><td>$+$</td><td>$-$</td><td>$-$</td><td>keine</td></tr>
<tr><td>$-$</td><td>$-$</td><td>$-$</td><td>$f = ((h + 2 \cdot (p_{12} + p_{13} + p_{23})) < p_{33})$</td></tr>
<tr><td>$+$</td><td>$+$</td><td>$-$</td><td>$f = ((h - 2 \cdot (p_{12} - p_{13} + p_{23})) < p_{33})$</td></tr>
<tr><td>$+$</td><td>$-$</td><td>$+$</td><td>$f = ((h - 2 \cdot (p_{12} + p_{13} - p_{23})) < p_{33})$</td></tr>
<tr><td>$-$</td><td>$+$</td><td>$+$</td><td>$f = ((h + 2 \cdot (p_{12} - p_{13} - p_{23})) < p_{33})$</td></tr>
</table>
</td><td>(2)</td></tr>
<tr><td>IF</td><td>$f = 1$</td><td>(3)</td></tr>
<tr><td>THEN</td><td>$\mathbf{a}_3 = \mathbf{a}_3 - \mathrm{sign}(p_{13}) \cdot \mathbf{a}_1 - \mathrm{sign}(p_{23}) \cdot \mathbf{a}_2$</td><td>(4)</td></tr>
<tr><td></td><td>aktualisiere p_{ij} gemäß der durchgeführten Vektoroperation, falls Daten der reduzierten Form gewünscht werden</td><td>(5)</td></tr>
</table>

Für die im Algorithmus **LastStep** verwendete Hilfsvariable f gilt:

$$f = \begin{cases} 1 & \text{wenn die Bedingung erfüllt ist} \\ 0 & \text{sonst} \end{cases}$$

Funktionsweise des TPR-Algorithmus: Sei $B = (\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3)$ jeweils die Matrix, die aus der Eingabe unmittelbar vor Erreichen einer **Gauß**$_i$–Routine ($1 \leq i \leq 3$) berechnet wurde. Der TPR-Algorithmus ist so konzipiert, daß

$$||\mathbf{b}_1|| \leq ||\mathbf{b}_2|| \leq ||\mathbf{b}_3||$$

gilt.

Unmittelbar vor Ausführung der $\mathbf{Gauß}_3$−Subroutine gilt, daß die Vektoren $\mathbf{b}_1$ und $\mathbf{b}_2$ sowie $\mathbf{b}_1$ und $\mathbf{b}_3$ Gauß-reduziert sind. Die Bedingung in (12) gewährleistet, daß nach Verlassen der Repeat-Until Schleife, die aus den Anweisungen (2) bis (12) besteht, alle Paare von Spaltenvektoren Gauß-reduziert sind. Im Korrektheitsbeweis wird gezeigt, daß **LastStep** aus einer Basis eines Gitters L vom Rang drei, deren sämtliche Paare Gauß-reduziert sind, in einem Schritt eine Mikowski-reduzierte Basis von L berechnet.

5.2 Komplexitätsresultat

Satz 5.8 *Der TPR-Algorithmus berechnet bei Eingabe einer zulässigen Matrix $A \in \mathbb{Z}^{d\times 3}$ eine Matrix, deren Spalten eine Minkowski-reduzierte Basis des Gitters $L(A)$ bilden. Die Anzahl der Gaußschritte T genügt dabei der obern Schranke:*

$$T \leq \max\{44, 1654 \log_2 E(A)\}.$$

Die Bitkomplexität C_{bit} genügt, bei Verwendung klassischer Arithmetik für die elementaren Operationen, der oberen Schranke:

$$C_{\text{bit}} \leq 3 + d\,(\log_2 E(A))^2 \cdot \max\{44, 1654 \log_2 E(A)\}.$$

In den nächsten Abschnitten wird der TPR-Algorithmus analysiert. Die Analyse erweist sich als äußerst schwieriges Problem. Es werden im Verlauf der Komplexitätsanalyse unendliche Familien von Beispielen für Eingaben klassifiziert, die zu unteren Laufzeitschranken für die Anzahl der Gaußschritte führen. Einzelne Repräsentanten dieser Familien werden mit hoher quantitativer Präzisision studiert.

5.3 Präanalytische Studien

In diesem Abschnitt werden einige Ergebnisse vor der Analyse gegeben, die die folgenden Fragestellungen beantworten, die sich in natürlicher Weise stellen.

1) Läßt sich das Komplexitätsergebnis aus Satz 5.8 signifikant verbessern?
2) Führt eine Analysemethodik, analog zu der des LLL-Algorithmus, beim TPR-Algorithmus zum Ziel?

Die Anwort auf beide Fragen ist **nein**

Bevor wir mit der Analyse beginnen, möchten wir eine parametrisierte Eingabe A_k angeben, die verdeutlicht, daß die in Satz 5.8 behauptet Laufzeitschranke für die Bitkomplexität nicht entscheidend verbessert werden kann.

Die erste Frage wird von Lemma 5.9 beantwortet.

Lemma 5.9 (Eine untere Laufzeitschranke für die Bitkomplexität von TPR)
Sei A_k definiert als

$$A_k = \begin{pmatrix} 1 & 0 & 2^k \\ -1 & 1 & 2^k \\ 0 & -1 & 0 \end{pmatrix} \tag{5.3}$$

und sei C_{bit} die bei Eingabe von A_k benötigte Anzahl von Bitoperationen.

Der TPR-Algorithmus führt bei Eingabe der Matrix A_k wenigsten k Gaußschritte aus. Die Anzahl der Bitoperationen, bei Verwendung klassischer Algorithmen für Multiplikation und Division, kann durch

$$C_{\text{bit}} \geq \frac{1}{3^3} \cdot (\log E(A))^3 \ \text{für}\ k \geq 13$$

nach unten abgeschätzt werden.

Beweis: Es bezeichne $(r_1, r_2, \ldots, r_k)$ die Folge ganzer Zahlen, die nacheinander in Anweisung (2) der Subgaußalgorithmen **Gauß**$_2$ und **Gauß**$_3$ auftreten. Es ist

$$r_i = 2^{k-i} \text{ für } 1 \leq i \leq k.$$

Damit ergibt sich

$$\prod_{i=1}^{k} |r_i| = \prod_{i=1}^{k-1} 2^i = 2^{\sum_{i=1}^{k-1} i} = 2^{k(k-1)/2}.$$

Die kleinste Komponente der dritten Spalte der Ausgabematrix, die wir z nennen, erfüllt

$$|z| \geq \frac{2^k}{3} > 2^{k-2} \text{ für } k > 2.$$

Der Grund dafür ist, daß die Summe der Komponenten der dritten Spalte in jedem Reduktionsschritt konstant bleibt und der Algorithmus so lange nicht terminiert, bis alle Differenzen von Komponenten dieser Spalte kleiner als zwei sind. Die Multiplikation zweier Zahlen $z_1, z_2 \neq 0$ kostet wenigstens $\log(2|z_1|) \cdot \log(2|z_2|)$ Bitoperationen. Für $z_2 \neq 0$ kostet die Berechnung von $r = -\lfloor z_1/z_2 \rceil$ wenigstens $\log(2 \max\{1, |r|\}) \cdot \log(2 \max\{|z_1|, |z_2|\})$ Bitoperationen. Daraus können wir die folgende untere Schranke für die Bitkosten ableiten:

$$\begin{array}{rcl} C_{\text{bit}} & \geq & \sum_{i=2}^{k} (\log |r_i| \cdot \log 2^{k-1}) \\ & = & \log\left(2^{k(k-1)/2}\right) \log\left(2^{k-1}\right) \\ & = & k(k-1)/2 \cdot (k-1) \geq k^3/3 \text{ für } k \geq 2 \end{array}$$

Damit erhalten wir:

$$\begin{array}{rcl} 1/27 \log(E(A_k))^3 & = & 1/27 \left(\log(2^{2k+1})\right)^3 \\ & = & 1/27\,(2k+1)^3 \\ & \leq & 1/9\, k^3 \\ & \leq & C_{\text{bit}} \end{array}$$

□

Mit Lemma 5.10 wird illustriert, daß die Analysetechnik des LLL-Algorithmus beim TPR-Algorithmus nicht anwendbar ist.

Lemma 5.10 *Es existiert kein festes $y \in (0, 1)$, so daß gilt: Sind alle Paare der Matrix, die im TPR-Algorithmus nach einer Anzahl von Gaußschritten erhalten wurde, y-Gauß-reduziert, so läuft der TPR-Algorithmus nur noch konstant viele Schritte.*

Beweis: Betrachte die Matrix A_k aus Gleichung (5.3). Es bezeichne $\mathbf{a}_3$ die dritte Spalte der Eingabematrix und $\mathbf{b}_3$ die dritte Spalte der Ausgabematrix. Dann gilt:

$$\frac{||\mathbf{b}_3||^2}{||\mathbf{a}_3||^2} = \frac{2}{3} \tag{5.4}$$

Sei

$$\mathbf{b}_3^{(l)} = (b_{13}^{(l)}, b_{23}^{(l)}, b_{33}^{(l)})$$

die Matrix, die aus der Eingabematrix nach $l = \hat{t}_2 + \hat{t}_1$ Gaußschritten für $1 \leq l \leq T$ berechnet wurde.

Dann erhält man

$$\sum_{i=1}^{3} b_{i3}^{(k)} = 2^{2k+1} \, (1 \leq k \leq T).$$

Der TPR-Algorithmus terminiert genau dann, wenn

$$\max\{|b_{i3}^{(l)} - b_{j3}^{(l)}| : 1 \leq i, j \leq 3\} \leq 1$$

gilt. Nach dem Terminieren ist

$$||\mathbf{b}_3||^2 \geq 2^{2k+2}/3,$$

womit die Gültigkeit von Gleichung 5.4 gezeigt wurde. Definiere

$$f_k = \frac{||\mathbf{b}_3^{(k+1)}||^2}{||\mathbf{b}_3^{(k)}||},$$

dann ist

$$(\max\{f_i : 1 \leq i \leq T-1\})^k \geq \prod_{i=1}^{k} f_i \geq 2/3.$$

Damit ist gezeigt, daß kein $y \in (0,1)$ existieren kann, so daß alle Paare $y-$Gauß-reduziert sind und der TPR-Algorithmus nur noch konstant viele Schritte ausführt. Denn nehmen wir, an es gäbe ein derartiges y, so wähle:

$$\hat{k} = \left\lfloor \frac{\log(3/2)}{\log(1/y)} \right\rfloor + l$$

für beliebiges $l \in \mathbb{N}$.

Der TPR-Algorithmus läuft bei Eingabe der Matrix $A_{\hat{k}}$, nachdem alle Paare von Spaltenvektoren Gauß-reduziert sind, noch wenigstens l-Gaußschritte. Dies steht aber im Widerspruch zur konstanten Anzahl von Schritten.

□

5.4 Korrektheit des TPR-Algorithmus

Die Korrektheit des Algorithmus wird bewiesen, indem wir einen Satz von Hermann Minkowski benutzen, in dem Minkowski-Reduziertheit für Gitter vom Rang $n \in \{2,3,4\}$ durch eine kleine Anzahl von Bedingungen charakterisiert wird. Es wird gezeigt, daß die Ausgabe des TPR-Algorithmus diese Bedingungen erfüllt. Da alle verwendeten Matrixoperationen unimodular sind, ist die Ausgabematrix eine Basis des Gitters $L(A)$.

Satz 5.11 (Hermann Minkowski) [2] *Für $n \in \{2,3,4\}$ ist eine positiv definite Form*

$$\varphi(\xi_1, \xi_2, \ldots, \xi_n) = \sum_{1 \leq h,k \leq n} \alpha_{hk} \xi_h \xi_k$$

Minkowski-reduziert, wenn sie die Bedingungen

[2] ursprünglich in französischer Sprache abgefaßt

$\|\|\mathbf{a}_2\|\| \geq \|\|\mathbf{a}_1\|\|$	(1)
$\|\|\mathbf{a}_3\|\| \geq \|\|\mathbf{a}_2\|\|$	(4)
$\|\|\mathbf{a}_2 + \mathbf{a}_1\|\| \geq \|\|\mathbf{a}_2\|\|$	(2)
$\|\|\mathbf{a}_2 - \mathbf{a}_1\|\| \geq \|\|\mathbf{a}_2\|\|$	(3)
$\|\|\mathbf{a}_3 + \mathbf{a}_1\|\| \geq \|\|\mathbf{a}_3\|\|$	(5)
$\|\|\mathbf{a}_3 - \mathbf{a}_1\|\| \geq \|\|\mathbf{a}_3\|\|$	(6)
$\|\|\mathbf{a}_3 + \mathbf{a}_2\|\| \geq \|\|\mathbf{a}_3\|\|$	(7)
$\|\|\mathbf{a}_3 - \mathbf{a}_2\|\| \geq \|\|\mathbf{a}_3\|\|$	(8)
$\|\|\mathbf{a}_3 + \mathbf{a}_2 + \mathbf{a}_1\|\| \geq \|\|\mathbf{a}_3\|\|$	(9)
$\|\|\mathbf{a}_3 + \mathbf{a}_2 - \mathbf{a}_1\|\| \geq \|\|\mathbf{a}_3\|\|$	(10)
$\|\|\mathbf{a}_3 - \mathbf{a}_2 + \mathbf{a}_1\|\| \geq \|\|\mathbf{a}_3\|\|$	(11)
$\|\|\mathbf{a}_3 - \mathbf{a}_2 - \mathbf{a}_1\|\| \geq \|\|\mathbf{a}_3\|\|$	(12)

Tabelle 5.2: Charakterisierung von Minkowski-Reduziertheit

i) $\alpha_{11} \leq \alpha_{22} \leq \ldots \leq \alpha_{nn}$

ii) $\varphi(\epsilon_1, \epsilon_2, \ldots, \epsilon_h) \geq \alpha_{hh}$

erfüllt, wobei ϵ_h eine Einheit (± 1) ist und $\epsilon_k\,(h \neq k)$ die Werte 0 oder ± 1 annehmen kann. Sind i) *und* ii) *erfüllt, so gilt für das i-te sukzessive Minimum λ_i, $\lambda_i = \alpha_{ii}\,(1 \leq i \leq n)$. Ferner gilt: Die sukzessiven Minima jedes der Form assoziierten Gitters bilden eine Basis dieses Gitters $(n < 4)$.*

Ein fehlerfreier Nachdruck des Beweises der letzten Aussage des Satzes findet sich in [4] S. 257 ff.

Für Gitter vom Rang zwei und drei wird nun ein minimaler Satz von Bedingungen zur Charakterisierung von Minkowski-Reduziertheit abgeleitet:

Korollar 5.12 (Minkowski-Reduziertheit, minimale Bindingungsanzahl) *Sei $L_1 \subset \mathbb{R}^d$ ein Gitter vom Rang zwei mit Basis $A_1 = (\mathbf{a}_1, \mathbf{a}_2) \in \mathbb{R}^{d \times 2}$ und $L_2 \subset \mathbb{R}^d$ ein Gitter vom Rang drei mit Basis $A_2 = (\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3) \in \mathbb{R}^{d \times 2}$.*

Erfüllen die Vektoren $\mathbf{a}_1$ und $\mathbf{a}_2$ die Bedingungen (1), (2) und (3) aus Tabelle 5.2, so ist A_1 ein Minkowski-reduzierte Basis des Gitters L_1.

Erfüllen die Vektoren $\mathbf{a}_1$, $\mathbf{a}_2$ und $\mathbf{a}_3$ alle zwölf Bedingungen aus Tabelle 5.2, so ist A_2 eine Minkowski-reduzierte Basis des Gitters L_2.

Beweis: Die erste Aussage ergibt sich unmittelbar aus Satz 5.11. Für die zweite Aussage werden aus den $2 + 2 \cdot 6 + 2 \cdot (3^2 - 1) = 30$ Bedingungen von Satz 5.11

$$||\mathbf{a}_1|| \leq ||\mathbf{a}_2|| \leq ||\mathbf{a}_3||,$$

$$|| \pm \mathbf{a}_2 \pm \mathbf{a}_1|| \geq ||\mathbf{a}_2||,$$

$$|| \pm \mathbf{a}_3 \pm \mathbf{a}_1|| \geq ||\mathbf{a}_2||,$$
$$|| \pm \mathbf{a}_3 \pm \mathbf{a}_2|| \geq ||\mathbf{a}_3||$$

und

$$|| \pm \mathbf{a}_3 \pm \mathbf{a}_2 \pm \mathbf{a}_1|| \geq ||\mathbf{a}_3||,$$

die redundanten Bedingungen eliminiert. Damit ergibt sich der minimale Satz von Bedingungen (Tabelle 5.2), der erforderlich ist, um eine Minkowski-reduzierte Basis zu charakterisieren.

□

An dieser Stelle wird nun der Begriff der *totale-Paar-Reduziertheit* eingeführt.

Definition 5.13 (Paar-Reduziertheit) *Sei $L \in \mathbb{R}^d$ ein Gitter und*

$$A = (\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_n) \in \mathbb{R}^{d \times n}$$

eine Basis von L. Die Basis A von L heißt total-Paar-reduziert, *wenn alle Paare $(\mathbf{a}_i, \mathbf{a}_j)$ Gauß-reduziert sind für $1 \leq i < j \leq n$.*

Aus der Definition ist ersichtlich, daß eine Minkowski-reduzierte Basis eines Gitters auch stets total-Paar-reduziert ist.

Offenbar hat der TPR-Algorithmus unmittelbar vor Ausführung von Anweisung (13) schon eine total-Paar-reduzierte Basis des zur Eingabematrix korresponierenden Gitters $L(A)$ berechnet. Mit Satz 5.11 wird gezeigt, daß der letzte Schritt des TPR-Algorithmus die Matrix $(\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3)$ so modifiziert, daß die Bedingungen (9), (10), (11) und (12) von Korollar 5.12 hergestellt werden, **ohne** daß die anderen Bedingungen (1) bis (8) zerstört werden. Der letzte Schritt hat also keinen Einfluß auf die totale-Paar-Reduziertheit.

Satz 5.14 *Sei L ein Gitter vom Rang drei, und seien λ_1, λ_2 und λ_3 das erste, zweite und dritte sukzessive Minimum von L. Desweiteren*

$$A = (\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3) \in \mathbb{R}^{d \times 3}$$

eine Basis von L, für welche die Bedingungen (1) bis (8) aus Tabelle 5.2 erfüllt sind.

Dann gilt

$$||\mathbf{a}_1|| = \lambda_{\tau(1)},$$
$$||\mathbf{a}_2|| = \lambda_{\tau(2)}$$

und

$$\lambda_{\tau(3)} = \min\{||a_3||, ||a_3 \pm \mathbf{a}_2 \pm \mathbf{a}_1||\},$$

wobei $\tau \in S(3)$ eine geignete Permutation ist.

Beweis:

1. Fall: Es gelte

$$||\mathbf{a}_3|| \leq ||\mathbf{a}_3 \pm \mathbf{a}_2 \pm \mathbf{a}_1||.$$

In diesem Fall sind alle zwölf Bedingungen erfüllt und mit Korollar 5.12 und Satz 5.11 ergibt sich

$$||\mathbf{b}_i|| = \lambda_i \text{ für } 1 \leq i \leq 3.$$

2. Fall: Es sei

$$\min\{||\mathbf{a}_3||^2, ||\mathbf{a}_3 \pm \mathbf{a}_2 \pm \mathbf{a}_1||^2\} < ||\mathbf{a}_3||^2.$$

In diesem Fall ist wenigstens eine der Bedingungen (9) bis (12) verletzt. Sei o. B. d. A. die Bedingung (9) verletzt. Denn ist nicht die Bedingung (9) sondern etwa die Bedingung (10) verletzt, so führt eine Inversion $\mathbf{a}_1 \leftarrow -\mathbf{a}_1$ zu einer Matrix, in der jetzt die Bedingung (9) verletzt ist, wohingegegen aber (1) bis (8) unvermindert gelten. Dies ist der Fall wegen der Symmetrie der Bedingugen (2),(3) und (5) bis (8). Analoges gilt in einem Szenario, in dem die Bedingung (10) statt (9) verletzt ist. Die Inversion hat nur zur Folge, daß für die Bedingungen (2),(3) und (5) bis (8) zwei Label geswapt werden, nämlich (6) $\leftrightarrow$ (5) und (3) $\leftrightarrow$ (2).

Für den eigentlichen Beweis wird zunächst ein Hilfssatz gezeigt, der den Beweis des Satzes unmittelbar komplettiert.

Hilfssatz 5.15 (Optimalitäts-Lemma) *Sei L ein Gitter vom Rang drei mit einer Basis $(\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3) \in \mathbb{Z}^{d\times 3}$, die die Bedingungen (1) bis (8) von Tabelle 5.2 erfüllt. Sei (x, y) ein Paar ganzer Zahlen, für das $||\mathbf{a}_3 + x\mathbf{a}_1 + y\mathbf{a}_2|| < ||\mathbf{a}_3||$ erfüllt ist. Dann gilt*

$$(x, y) \in \{(1, 1), (-1, 1), (1, -1), (-1, -1)\}.$$

Beweis: Bezeichne R die im Hilfssatz auftretende Menge. Es werden nun schrittweise alle Paare ganzer Zahlen, außer den vier genannten, ausgeschlossen.

Beginnen wir mit den Paaren ganzer Zahlen (x, y), für die $x = 0$ oder $y = 0$ gilt. Der Fall $(x = 0, y \in \mathbb{Z})$ steht im Widerspruch zu (7) oder (8). Ebenso kann der Fall $(x \in \mathbb{Z}, y = 0)$ nicht eintreten, weil sonst (5) oder (6) verletzt sind.

Es verbleiben also noch die Fälle mit $(x, y) \in (\pm\mathbb{N}, \pm\mathbb{N})$. Die Bedingungen (1) bis (8) sind äquivalent zu den acht Bedingungen

$$\frac{|\langle \mathbf{a}_i, \mathbf{a}_j\rangle|}{||\mathbf{a}_i||^2} \leq \frac{1}{2}, \text{ und } ||\mathbf{a}_i|| \leq ||\mathbf{a}_j||\, (1 \leq i < j \leq 3).$$

Wählen wir die Bezeichnungen

$$p_{ij} = \langle \mathbf{a}_i, \mathbf{a}_j\rangle\, (1 \leq i < j \leq 3),$$

dann ergibt sich

$$\begin{aligned} p_{33} &> ||\mathbf{a}_3 + x\,\mathbf{a}_1 + y\,\mathbf{a}_2||^2 \\ &= p_{33} + x^2\, p_{11} + y^2\, p_{22} + 2\,(x\,y\,p_{12} + x\,p_{13} + y\,p_{23}) \\ &\geq p_{33} + x^2\, p_{11} + y^2\, p_{22} - 2\,(|xy|\,|p_{12}| - |x|\,|p_{13}| - |y|\,|p_{23}|). \end{aligned}$$

Daraus ergibt sich die Ungleichung

$$0 > x^2\, p_{11} + y^2\, p_{22} - 2\,(|x\,y|\,|p_{12}| + |x|\,|p_{13}| + |y|\,|p_{23}|).$$

Aus den Bedingungen, daß alle Paare Gauß-reduziert sind folgt

$$\begin{aligned} -2\,|p_{12}| &\geq -p_{11} \\ -2\,|p_{13}| &\geq -p_{11} \\ -2\,|p_{23}| &\geq -p_{22}. \end{aligned}$$

Damit erhält man

$$0 > p_{11}\,(x^2 - |x\,y| - |x|) + p_{22}(y^2 - |y|).$$

Da $p_{22} \geq p_{11}$ und natürlich $p_{11} > 0$ gilt, folgt

$$0 > (x^2 - |xy| - |x|) + (y^2 - |y|) = f(x,y). \tag{5.5}$$

Jetzt weisen wir nach, daß alle Paare $(x,y) \notin R$ die abgeleitete Bedingung (5.5) verletzen. Die Funktion $f(x,y)$ ist symmetrisch,

$$f(x,y) = f(y,x) \text{ für alle } x,y \in \mathbb{Z}.$$

Wir können daher $f(x,y)$ mit $d = |x-y|$ wie folgt notieren

$$\begin{aligned} f(x,y) &= f(\,|x|, |y|\,) \\ &= f(\,|x|, |x| + d\,) \\ &= x^2 + (\,|x| + d\,)^2 - |\,|x|\,(\,|x| + d\,)\,| - |x| - |\,|x| + d\,| \\ &= x^2 + x^2 + 2\,|x|\,d + d^2 - x^2 - |x|\,d - |x| - |x| - d \\ &= x^2 + |x|\,(d-2) + d^2 - d. \end{aligned}$$

Dabei ist $d > 0$ für $(x,y) \notin R$. Dann ist aber (5.5) verletzt. Es ist

$$f(x,y) > 0 \text{ für } d \geq 2$$

und

$$f(|x|, |x| + d) = f(|x|, |x| + 1) = x^2 - |x| \geq 0 \text{ für alle } x \geq 1.$$

Damit ist das Ausschlußverfahren beendet und das Lemma bewiesen. □

Der Beweis für den zweiten Teil des Satzes 5.14 ergibt sich nun wie folgt. Es gilt:

$$||\mathbf{a}_3|| > ||\mathbf{a}_3 + \mathbf{a}_2 + \mathbf{a}_1||$$

Wir haben gerade nachgewiesen, daß in dieser Situation die Wahl für einen Vektoraustausch durch die verletzte Bedingung (9) optimal war, das heißt, nach Austausch von $\mathbf{a}_3$ durch

$$\mathbf{a}_3' = \mathbf{a}_3 + \mathbf{a}_2 + \mathbf{a}_1$$

ist

$$||\mathbf{a}_3' + x\,\mathbf{a}_1 + y\,\mathbf{a}_2|| \geq ||\mathbf{a}_3'||$$

für alle ganzzahligen x, y, da sonst die Wahl des $(+1,+1)$ Paares zur Reduktion nicht optimal gewesen wäre.

Ist $||\mathbf{a}_3'|| \geq ||\mathbf{a}_2||$, so sind bereits alle Bedingungen (1) bis (12) erfüllt und damit die Aussage nach Satz 5.11 bewiesen.

Ist

$$||\mathbf{a}_2|| > ||\mathbf{a}_3'|| \geq ||\mathbf{a}_1||,$$

so führt ein Tausch von $\mathbf{a}_3'$ mit $\mathbf{a}_2$ dazu, daß alle Bedingungen (1) bis (12) erfüllt sind. Denn wäre eine Bedingung verletzt, so könnte die Wahl

$$||\mathbf{a}_3'|| = \min\{||\mathbf{a}_3 + x\,\mathbf{a}_1 + y\,\mathbf{a}_2|| : x, y \in \mathbb{Z}\} < ||\mathbf{a}_3||$$

nicht optimal gewesen sein, was im Widerspruch zum Hilfssatz 5.15 steht.

Das gleiche Argument läßt sich auch auf den letzten Fall $||\mathbf{a}_3'|| < ||\mathbf{a}_1||$ anwenden, nachdem $\mathbf{a}_3'$ und $\mathbf{a}_1$ die Plätze getauscht haben.

Damit ist dieser erstaunliche Satz bewiesen, der uns zu einem neuen exakten Algorithmus zur Berechung Minkowski-reduzierter Basen inspiriert hat. □

Die Prozedur **LastStep** ist nach Satz 5.14 so konzipiert worden, daß danach alle zwölf Bedingungen von Korollar 5.12 gelten. Damit ist der Korrektheitsbeweis des TPR-Algorithmus abgeschlossen.

5.5 Analysekonzept

In diesem Abschnitt werden einige in der Analyse verwendeten Sprechweisen eingeführt und die entscheidenden Aspekte des der Analyse zugrundeliegenden Plans vorgestellt.

Aus für die Analyse wichtigen Gründen wurden im TPR-Algorithmus die Hilfsvariablen T und t_u $(1 \leq u \leq 3)$ eingeführt, die zu Beginn des Algorithmus auf Null gesetzt und nach jedem Gaußschritt inkremtiert werden, so daß die Gesamtzahl der bereits ausgeführten Gaußschritte durch T angegeben wird. Die Anzahl von Gaußschritten einer **Gauß**$_u$-Subroutine wird mit t_u bezeichnet, für $1 \leq u \leq 3$. Im Analyseabschnitt 5.6 wird gezeigt, daß der TPR-Algorithmus terminiert.

Es bezeichne

$$A_k = (\mathbf{a}_1^{(k)}, \mathbf{a}_2^{(k)}, \mathbf{a}_3^{(k)})$$

die Matrix, die der TPR-Algorithmus unmittelbar vor Ausführung des $k-$ten $(1 \leq k \leq T)$ Gaußschritts aus der Eingabe A berechnet hat und

$$B = (\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3)$$

bezeichne die Matrix unmittelbar vor Ausführung des Kommandos LastStep.

Sei $(i, j)\,(1 \leq i < j \leq 3)$ das Paar von Indizes der Spaltenvektoren, die im $k-$ten $(1 \leq k \leq T)$ Gaußschritt des TPR-Algorithmus reduziert werden. Wir nennen die ganze Zahl

$$r_k = -\left\lfloor \frac{\langle \mathbf{a}_i^{(k)}, \mathbf{a}_j^{(k)} \rangle}{||\mathbf{a}_i^{(k)}||^2} \right\rceil,$$

die im Gaußschritt k $(1 \leq k \leq T)$ zur Reduktion

$$\mathbf{a}_{j'}^{(k+1)} = \mathbf{a}_j^{(k)} + r_k \cdot \mathbf{a}_i^{(k)}$$

verwendet wird, die $k-$te *Kohärenzzahl* des TPR-Algorithmus. Dabei ist $1 \leq j' \leq j$ der Index der Spalte, an dem sich der reduzierte Vektor $\mathbf{a}_j^{(k)} + r_k \cdot \mathbf{a}_i^{(k)}$ durch Tausch oder Sortiervorgang unmittelbar vor dem Gaußschritt $k+1$ befindet, falls $k < T$ gilt. Ist $k = T$, so bezeichnet j' den Index der Spalte, in der sich der reduzierte Vektor $\mathbf{a}_j^{(k)} + r_k \cdot \mathbf{a}_i^{(k)}$ vor Ausführung des Kommandos LastStep befindet. Der Name Kohärenzzahl trägt der Tatsache Rechnung, daß Vektoren um so stärker korreliert sind, desto größer der Absolutbetrag ihrer Kohärenzzahl ist.

Als *Kohärenzzahlenfolge* bezeichnen wir das geordnete $T-$Tupel ganzer Zahlen

$$(r_1, r_2, \ldots, r_T),$$

das aus einer zulässigen Eingabe des TPR-Algorithmus im Ablauf des Algorithmus berechnet wird.

In jedem $\mathbf{Gau\ss}_u$-Aufruf $(1 \leq u \leq 3)$ wird eine Teilfolge

$$(r_\alpha, \ldots, r_\beta) \in \mathbb{Z}^{\beta-\alpha+1} \quad 1 \leq \alpha \leq \beta \leq T$$

der Kohärenzzahlenfolge berechnet. Eine solche Teilfolge der Kohärenzzahlenfolge bezeichnen wir als *Kohärenzzahlenabschnitt* der Kohärenzzahlenfolge. Die chronologische Konkatenation der Kohärenzzahlenabschnitte ergibt die Kohärenzzahlenfolge.

In den ersten Sätzen der Analyse wird gezeigt, daß der TPR-Algorithmus terminiert, und daß $T \leq E(A)$ für jede zulässige Eingabe A gilt.

Danach wird eine obere Schranke für die Anzahl der Gaußschritte des TPR-Algorithmus bestimmt, die linear in der Eingabelänge ist. Da alle Operanden von der Größenordnung der Eingabelänge, sind ergibt sich damit das Komplexitätsergebnis.

Zunächst ist klar, daß jeder zulässigen Eingabe eine eindeutig bestimmte Folge von Kohärenzzahlen zugeordnet wird. Zwei identsiche Kohärenzzahlenfolgen müssen allerdings nicht notwendigerweise aus der gleichen Eingabe entstanden sein, die Abbildung

$$\begin{array}{rccc} \Phi: & \{A \in \mathbb{Z}^{d\times 3} : A^t \cdot A \in \mathrm{GL}(3, \mathbb{Q})\} & \longrightarrow & \mathbb{Z}^{\mathbb{N}} \\ & A & \longmapsto & \Phi(A) = (r_1, r_2, \ldots, r_T) \end{array}$$

ist nicht injektiv. Das der Analyse zugrunde liegende Konzept besteht darin, alle Kohärenzzahlenabschnitte der Kohärenzzahlenfolge zu klassifizieren, und obere Schranken für die Anzahl der Kohärenzzahlenabschnitte eines jeden Typs zu bestimmen. Aus den oberen Schranken für die Häufigkeit des Auftretens eines bestimmten Typs von Kohärenzzahlenabschnitten ergibt sich eine obere Schranke für die Länge der Kohärenzzahlenfolge durch Summation der Produkte von oberen Schranken für die Anzahl jeden Typs von Kohärenzzahlenabschnitten und dessen Länge.

Die Typisierung der Kohärenzzahlenabschnitte wurde durch genaues Studium des Gaußalgorithmus gewonnen und in Tabelle 5.3 zusammengestellt. Dabei sind $\alpha, \beta \in \mathbb{Z}$ mit $1 \leq \alpha \leq \beta \leq T$. Ein Kohärenzzahlenabschnitt vom Typ LR_1 hat die Form

$$(\rho_\alpha, \ldots, \rho_\beta) \in \mathbb{Z}^t,$$

mit der Eigenschaft, daß $|\rho_\alpha| = 1, |\rho_\beta| > 1$ und falls $t \geq 3$ auch

$$|\rho_i| \geq 2 \ (\alpha < i \leq \beta).$$

Die Bezeichnung LR bedeutet, ein Kohärenzzahlenabschnitt ist vom Typ $LR_i \, (1 \leq i \leq 3)$. Analog steht die Bezeichnung PCR beziehungsweise CSR für Abschnitte die vom Typ CSR_i respektive $PCR_i \, (1 \leq i \leq 2)$ sind. Wir verwenden die Schreibweise $LR_i(t)$, um einen Abschnitt der Länge t vom Typ LR_i zu benennen. Außerdem wird die Bezeichnungsweise

$$\#LR_i(u,t) = t \cdot (\text{Anzahl aller } \mathbf{Gauß}_u\text{-Aufrufe mit Kohärenzzahlenabschnitt vom Typ } LR_i(t))$$

verwendet.

Die Größe

$$\sum_{u=1}^{3} \#LR_1(u,t)$$

bezeichnet also die Anzahl aller Folgenglieder der Kohärenzzahlenfolge, die zu Kohärenzzahlenabschnitten vom Typ $LR_1(t)$ gehören, die in einem $\mathbf{Gauß}_u$-Aufruf berechnet wurden.

Analoge Notationen gelten für die Typen $USR, PCR_j, CSR_j, NE_j \, (1 \leq j \leq 2)$ und

$$NLR_i, \ (1 \leq i \leq 3),$$

wobei Kohärenzzahlenabschnitte vom Typ NE_j und NLR_i nicht auftreten können.

Es wird bewiesen werden, daß in Tabelle 5.3 alle Typen von Kohärenzzahlenabschnitten klassifiziert wurden.

Zur Komplexitätsanalyse des TPR-Algorithmus werden wir den Satz 5.16 im nächsten Abschnitt beweisen.

Satz 5.16 *Sei T die Gesamtanzahl aller Gaußschritte, die der TPR-Algorithmus bei Eingabe einer zulässigen Matrix $A = (\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3)$ ausführt. Es gilt:*

$$T = \sum_{i=1}^{3} \left(\#USR(i) + \sum_{j=1}^{3} \sum_{t=2}^{T} \#LR_j(i,t) + \sum_{j=1}^{2} (\#PCR_j(i) + \#CSR_j(i)) \right) \tag{5.6}$$

$$\#NLR_j(i,t) = \#NE_1(i) = \#NE_2(i) = 0 \text{ für } 1 \leq i,j \leq 3, \ 2 < t \leq T \tag{5.7}$$

$$\sum_{i=1}^{3} \left(\sum_{t=2}^{T} \sum_{j=1}^{3} \#LR_j(i,t) + \sum_{k=1}^{2} \#PCR_k(i) \right) \leq 12 \left(\log_2 ||\mathbf{a}_3|| + \log_3 ||\mathbf{a}|| \right) \tag{5.8}$$

$$\#USR(1) \leq 14 \log_2 ||\mathbf{a}_3|| + 12 \log_3 ||\mathbf{a}_3|| \tag{5.9}$$

$$\#USR(2) \leq \#USR(3) + 18 \log_2 ||\mathbf{a}_3|| + 12 \log_3 ||\mathbf{a}_3|| \tag{5.10}$$

$$\#USR(3) \leq 1 + 2\,(\log_2 ||\mathbf{a}_3|| + \log_4 ||\mathbf{a}_3||) \tag{5.11}$$

$$\sum_{i=1}^{3}\sum_{j=1}^{2} \#CSR_j(i) \leq 21\,(2 + 48\,\log_2 ||\mathbf{a}_3|| + 36\,\log_3 ||\mathbf{a}_3|| + 4\log_4 ||\mathbf{a}_3||) + 132\log_7 ||\mathbf{a}_3|| \tag{5.12}$$

Aus Satz 5.16 und dem Lemma 5.17 ergibt sich das Komplexitätsresultat von Satz 5.8.

Lemma 5.17 *Seien $z_1, z_2 \in \mathbb{Z}$ mit $z_2 \neq 0$ und sei $\log_2(2\,|z_i|) \leq m$ für eine positive ganze Zahl m und sei $d = \lfloor z_1/z_2 \rceil$ mit $\log_2(2\,\max\{1, |d|\}) \leq n$ $(1 \leq i \leq 2)$. Bezeichne C_{bit} die Anzahl der Bitoperationen, die zur Berechnung von $\lfloor z_1/z_2 \rceil$ benötigt werden. Dann gilt: $C_{\text{bit}} \leq 4\,m \cdot n$.*

Bezeichnung des Kohärenzzahlen-abschnittstyps	Erklärung der Bezeichnung des Typs	Länge $t = \beta - \alpha + 1$	Aussehen eines Kohärenzzahlenab-schnitts dieses Typs	Charakteristika der Kohärenzzahlen des Abschnitts
USR	ultra short run	$t = 1$	(r_α)	$\|r_\alpha\| > 1$
LR_1	long run	$t \geq 2$	$(r_\alpha, \ldots, r_\beta)$	$\|r_\alpha\| = 1$ $\|r_i\| > 1\,(\alpha < i \leq \beta)$ $\|r_\beta\| > 1$
LR_2	long run	$t \geq 2$	$(r_\alpha, \ldots, r_\beta)$	$\|r_\alpha\| > 1$ $\|r_i\| > 1\,(\alpha < i \leq \beta)$ $\|r_\beta\| > 1$
LR_3	long run	$t > 2$	$(r_\alpha, \ldots, r_\beta)$	$\|r_\alpha\| = 1$ $\|r_i\| > 1\,(\alpha < i \leq \beta)$ $\|r_\beta\| \leq 1$
NLR_1	not existent long run 1	$t > 2$	$(r_\alpha, \ldots, r_\beta)$	$\|r_\alpha\| = 1$ $\|r_i\| \leq 1\,(\alpha < i \leq \beta)$ $\|r_\beta\| > 1$
NLR_2	not existent long run 2	$t > 2$	$(r_\alpha, \ldots, r_\beta)$	$\|r_\alpha\| > 1$ $\|r_i\| \leq 1\,(\alpha < i \leq \beta)$ $\|r_\beta\| > 1$
NLR_3	not existent long run 3	$t > 2$	$(r_\alpha, \ldots, r_\beta)$	$\|r_\alpha\| = 1$ $\|r_i\| \leq 1\,(\alpha < i \leq \beta)$ $\|r_\beta\| \leq 1$
PCR_1	pseudo critical 1	$t = 2$	(r_α, r_β)	$r_\alpha = 1 = r_\beta$
PCR_2	pseudo critical 2	$t = 2$	(r_α, r_β)	$r_\alpha = -1 = r_\beta$
CSR_1	critical short run 1	$t = 1$	(r_α)	$\|r_\beta\| \leq 1$
CSR_2	critical short run 2	$t = 2$	(r_α, r_β)	$\|r_\alpha\| = 1$ $r_\beta = 0$
NE_1	not existent	$t = 2$	(r_α, r_β)	$r_\alpha = 1$ $r_\beta = -1$
NE_2	not existent	$t = 2$	(r_α, r_β)	$r_\alpha = -1$ $r_\beta = 1$

Tabelle 5.3: Klassifikation von Kohärenzzahlenabschnitten

5.6 Analyse I

Sei im folgenden $A = (\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3)$ eine zulässige Eingabe des TPR-Algorithmus. In diesem Abschnitt werden die Gleichungen (5.6) und (5.7) sowie die Ungleichungen (5.8) bis (5.11) von Satz 5.16 bewiesen. Der Beweis von Ungleichung (5.12) wird in Abschnitt 5.7 abgehandelt. Zum Beweis werden zahlreiche Hilfssätze benötigt. Der Beweis wird wie folgt strukturiert.

I) **Satz 5.18** *Die Anzahl T der Gaußschritte, die der TPR-Algorithmus bei Eingabe einer zulässigen Matrix $A = (\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3)$ ausführt, genügt der folgenden Abschätzung:*

$$T \leq 3 \cdot \sum_{i=1}^{3} ||\mathbf{a}_i||^2$$

Der Satz 5.18 dient dazu, Konvergenzproblemen vorzubeugen. Zum Beweis sind Satz 5.21 und Lemma 5.22 erforderlich.

II) **Satz 5.19** *Das Klassifikationsschema aus Tabelle 5.3 ist vollständig. Es gelten die Gleichungen (5.6) und (5.7).*

III) **Satz 5.20** *Es gilt die Ungleichung (5.8).*

IV) Zum Abschluß des Abschnitts werden nacheinander die Ungleichungen (5.11), (5.9) und (5.10) bewiesen.

I) Beweis von Satz 5.18 Die Aussage gilt, da eine Vektoroperation mit nichtverschwindender Kohärenzzahl eine Reduktion der quadrierten euklidischen Länge um wenigstens eins bewirkt. Diese Tatsache wird in Lemma 5.22 gezeigt. Treten drei nichtverschwindende Kohärenzzahlen in Folge auf, so terminiert der TPR-Algorithmus. Ferner gilt $||v||^2 \geq 1$ für $v \in \mathbb{Z}^d$. Damit ist Satz 5.18 bewiesen.

Es werden nun noch zwei Aussagen gezeigt, die zum Beweis des Satzes 5.18 benötigt wurden. Der Satz 5.21 wird zum Beweis von Lemma 5.22 benötigt. Der erste Satz besagt, daß die Kohärenzzahl die beste Wahl einer ganzen Zahl zur Reduktion zweier Vektoren ist.

Satz 5.21 (Optimalität der Reduktion) *Seien $\mathbf{a}_1, \mathbf{a}_2 \in \mathbb{R}^d$ zwei linear unabhängige Vektoren. Die Kohärenzzahl*

$$r = -\left\lfloor \frac{\langle \mathbf{a}_1, \mathbf{a}_2 \rangle}{|| \ \mathbf{a}_1 \ ||^2} \right\rceil$$

hat die Eigenschaft:

$$||\mathbf{a}_2 + r \cdot \mathbf{a}_1||^2 = \min\{||\mathbf{a}_2 + z \cdot \mathbf{a}_1||^2 : z \in \mathbb{Z}\}$$

Beweis: Sei $f(x) = ||\mathbf{a}_2 + x \cdot \mathbf{a}_1||^2$. Die Funktion $f : \mathbb{R} \to \mathbb{R}^+$ ist stetig differenzierbar und besitzt das absolute Minimum

$$f(\overline{r}) = ||\mathbf{a}_2||^2 - \langle \mathbf{a}_1, \mathbf{a}_2 \rangle^2$$

mit

$$\overline{r} = -\frac{\langle \mathbf{a}_1, \mathbf{a}_2 \rangle}{\| \mathbf{a}_1 \|^2}.$$

Die Funktion f ist symmetrisch bezüglich ihres Minimums:

$$f(\overline{r} + x) = f(\overline{r} - x) \text{ für alle } x \in \mathbb{R}.$$

Ferner wächst die Funktion streng monoton für alle $x > \overline{r}$. Daher gilt:

$$f(\overline{r} - \lfloor r \rceil) \leq f(\overline{r} + z) \text{ für alle } z \in \mathbb{Z}.$$

□

Die Aussage des Satzes 5.19 läßt sich auch sehr gut geometrisch einsehen, da der Absolutbetrag der Kohärenzzahl r gerade die Länge der orthogonalen Projektion des Vektors $\mathbf{a}_2$ auf den Vektor $\mathbf{a}_1$ ist. Das Lot von $\mathbf{a}_2$ auf $\mathbf{a}_1$ ist die kürzeste Verbindung zwischen $\mathbf{a}_2$ und dem Vektor $\mathbf{a}_1$. Der nächste Gitterpunkt an den Punkt $\mathbf{a}_2 - \overline{r}\mathbf{a}_1$ im Gitter $\mathbb{Z}\mathbf{a}_1$ ist der kürzeste Vektor der Form $\mathbf{a}_2 + z \cdot \mathbf{a}_1$ mit einem $z \in \mathbb{Z}$. Für die Wahl dieses *nächsten Nachbarn* gibt es zwei Möglichkeiten, wenn $r \in \mathbb{Z}/2$, und ansonsten genau eine Möglichkeit. Die Ambiguität der Kohärenzzahlwahl wird durch unsere Definition von $\lfloor\ \rceil$ eliminiert. Unsere Motivation für diese Definition war es, nicht nur die Doppeldeutigkeit zu vermeinden, sondern auch zu garantieren, daß bei jeder Reduktionsoperation mit **nichtverschwindender** Kohärenzzahl, die euklidische Länge des reduzierten Vektors kleiner wird.

Diese Aussage wird in Lemma 5.22 bewiesen.

Lemma 5.22 (Verkürzungs-Lemma) *Seien $\mathbf{a}_1$ und $\mathbf{a}_2 \in \mathbb{R}^d$ linear unabhängig. Es gelte $||\mathbf{a}_2|| \geq ||\mathbf{a}_1||$ und*

$$\left| \frac{\langle \mathbf{a}_1, \mathbf{a}_2 \rangle}{||\mathbf{a}_1||^2} \right| > \frac{1}{2}.$$

Dann gilt

$$||\mathbf{a}_2 - \left\lfloor \frac{\langle \mathbf{a}_1, \mathbf{a}_2 \rangle}{||\mathbf{a}_1||} \right\rceil \mathbf{a}_1||^2 < ||\mathbf{a}_2||^2.$$

Beweis: Nach dem Optimalitätslemma ist

$$||\mathbf{a}_2||^2 \geq ||\mathbf{a}_2 - \left\lfloor \frac{\langle \mathbf{a}_1, \mathbf{a}_2 \rangle}{||\mathbf{a}_1||^2} \right\rceil \mathbf{a}_1||^2.$$

Annahme:

$$||\mathbf{a}_2||^2 = ||\mathbf{a}_2 - \left\lfloor \frac{\langle \mathbf{a}_1, \mathbf{a}_2 \rangle}{||\mathbf{a}_1||^2} \right\rceil \mathbf{a}_1|| = ||\mathbf{a}_2 + x\,\mathbf{a}_1||^2$$

mit

$$x = -\left\lfloor \frac{\langle \mathbf{a}_1, \mathbf{a}_2 \rangle}{||\mathbf{a}_1||^2} \right\rceil.$$

Dann gilt

$$||\mathbf{a}_2||^2 = ||\mathbf{a}_2||^2 + 2x \langle \mathbf{a}_1, \mathbf{a}_2 \rangle + x^2\, ||\mathbf{a}_1||^2.$$

Dies impliziert $-2x\langle \mathbf{a}_1, \mathbf{a}_2\rangle = x^2\,||\mathbf{a}_1||^2$. Da $x \neq 0$ folgt

$$x = -2\frac{\langle \mathbf{a}_1, \mathbf{a}_2\rangle}{||\mathbf{a}_1||^2} = -\left\lfloor \frac{\langle \mathbf{a}_1, \mathbf{a}_2\rangle}{||\mathbf{a}_1||^2} \right\rceil.$$

Nach der Definition der nächsten ganzen Zahl ist aber

$$\left|\left\lfloor \frac{\langle \mathbf{a}_1, \mathbf{a}_2\rangle}{||\mathbf{a}_1||^2} \right\rceil\right| < \frac{|\langle \mathbf{a}_1, \mathbf{a}_2\rangle|}{||\mathbf{a}_1||^2} + \frac{1}{2}.$$

Dies führt zu $|\langle \mathbf{a}_1, \mathbf{a}_2\rangle|/||\mathbf{a}_1||^2 < 1/2$, im Widerspruch zur Voraussetzung. □

II) Der Beweis wird in die Lemmata 5.23 und 5.24 untergliedert.

Im folgenden wird häufig ein einzelner Gaußalgorithmus, etwa Algorithmus 5.2, isoliert untersucht. Es bezeichne für diese Untersuchungen t die Anzahl von Gaußschritten, die der Gaußalgorithmus hintereinander ausführt. Die Matrix

$$(\mathbf{a}_1, \mathbf{a}_2) \in \mathbb{Z}^{d\times 2}$$

mit

$$(\mathbf{a}_1, \mathbf{a}_2)^t \cdot (\mathbf{a}_1, \mathbf{a}_2) \in GL(2, \mathbb{Q}) \text{ und } ||\mathbf{a}_1|| \leq ||\mathbf{a}_2||$$

wird *eine zulässige Eingabematrix des Gaußalgorithmus* genannt. Der TPR-Algorithmus ist so konzipiert, daß die **Gauß**$_i$-Subalgorithmen $(1 \leq i \leq 3)$ immer zulässige Eingaben erhalten. Ein Kohärenzzahlenabschnitt

$$(r_1, \ldots, r_t) \in \mathbb{Z}^t$$

ist die Folge von Kohärenzzahlen, die beim Aufruf eines Gaußalgorithmus, etwa Algorithmus 5.2 berechnet wird.

Es wird nun gezeigt, welche Kohärenzzahlenabschnitte für den Gaußalgorithmus nicht auftreten können, wenn der Algorithmus weniger als drei Schritte läuft.

Lemma 5.23 (Ausschluß-Lemma) *Es gibt keine zulässige Eingabe* $(\mathbf{a}_1, \mathbf{a}_2) \in \mathbb{Z}^{d\times 2}$*, auf die ein Gaußalgorithmus einen Kohärenzzahlenabschnitt vom Typ* NE *berechnet.*

Beweis: Wir nehmen an es gäbe eine derartige Eingabe für die $r_1 r_2 = -1$ gilt. Da $|r_1| = |r_2| = 1$, impliziert dies $r_1 = -r_2$. Es bezeichnen nun die einfachgestrichenen Vektoren die Vektoren nach dem ersten und die doppelt gestrichenen Vektoren die Vektoren nach dem zweiten Reduktionsschritt.

1. Schritt: Es ist $||\mathbf{a}_1||^2 \leq ||\mathbf{a}_2||^2$ und $\mathbf{a}_2' = \mathbf{a}_2 + r_1\mathbf{a}_1$. Da die Schrittzahl größer eins ist, folgt $||\mathbf{a}_2'|| < ||\mathbf{a}_1||$.

2. Schritt: Es ist $||\mathbf{a}_2'|| < ||\mathbf{a}_1|| \leq ||\mathbf{a}_2||$.

$$\mathbf{a}_2'' = \mathbf{a}_1 + r_2\mathbf{a}_2' = \mathbf{a}_1 - r_1(\mathbf{a}_2 + r_1\mathbf{a}_1) = \mathbf{a}_1 - r_1\mathbf{a}_2 - r_1^2 = -r_1\mathbf{a}_2.$$

Aus dem Verkürzungslemma folgt, da $r_2 \neq 0$ nun $||\mathbf{a}_2''|| < ||\mathbf{a}_1||$, aber $||\mathbf{a}_2''|| = ||-r_1\mathbf{a}_2|| = ||\mathbf{a}_2||$ und es war $||\mathbf{a}_2|| \geq ||\mathbf{a}_1||$, was ein Widerspruch ist. □

Falls ein Gaußalgorithmus also genau zwei Schritte läuft, können nur die Kohärenzzahlenabschnitte

$$(r_1, r_2) \in \mathbb{Z}^2 - \{(0,0), (1,-1), (-1,1)\}$$

auftreten. Die Klassifikation der verbleibenden Abschnittstypen gemäß Tabelle 5.3, ist in Tabelle 5.4 gegeben.

Kohärenzzahlenabschnitt	Typ
$\left.\begin{array}{r}(-1,0)\\ (\ \ 1,0)\end{array}\right\}$	$CSR_2(i), 1 \leq i \leq 3$
$(0, \pm 1)$	tritt trivialerweise nicht auf
$\left.\begin{array}{r}(-1,-1)\\ (\ \ 1,1)\end{array}\right\}$	$PSR_{1,2}(i), 1 \leq i \leq 3$

Tabelle 5.4: Kohärenzzahlen in Kohärenzzahlenabschnitten der Länge zwei

Es wird nun gezeigt, daß durch die Typbezeichnungen

$$LR_j(t)\ (1 \leq j \leq 2,\ t \geq 2),\ \text{und}\ LR_3(t)\ t \geq 3$$

alle Kohärenzzahlenabschnitte von Gaußalgorithmen klassifiziert werden, deren Abschnittslänge $t \geq 2$ ist, und die wenigstens eine Kohärenzzahl vom Absolutbetrag größer eins enthalten. Es wird bewiesen, daß Kohärenzzahlenabschnitte der Länge $t \geq 3$, außer an erster und letzter Position, nur Kohärenzzahlen vom Absolutbetrag größer eins enthalten, was beweist, daß es keine Kohärenzzahlenabschnitte vom Typ NLR gibt.

Lemma 5.24 (1-Lemma) *Sei $t > 1$ und $(r_1, r_2, \ldots, r_t)$ der Kohärenzzahlenabschnitt eines Gaußalgorithmus, bei Eingabe einer zulässigen Matrix $A = (\mathbf{a}_1, \mathbf{a}_2)$. Aus*

$$|r_s| \leq 1 \text{ für ein } s \text{ mit } 2 \leq s \leq t$$

folgt $t = s$.

Beweis: Sei $(\mathbf{c}_1, \mathbf{c}_2)$ die Matrix, die aus $(\mathbf{a}_1, \mathbf{a}_2)$ nach exakt $s-1$ Gaußschritten berechnet wurde. Da $s - 1 \geq 1$ folgt

$$2 \cdot |\langle \mathbf{c}_1, \mathbf{c}_2 \rangle| \leq ||\mathbf{c}_1||^2.$$

Ist $r_s = 0$, so ist trivialerweise $t = s$. Im anderen Fall ($|r_s| = 1$) gilt

$$||\mathbf{c}_2||^2 < ||\mathbf{c}_1||^2.$$

Nach dem Tausch gelten die Bezeichnungen

$$\mathbf{c}_1' = \mathbf{c}_2 \text{ und } \mathbf{c}_2' = \mathbf{c}_1.$$

Im s-ten Gaußschritt erfolgt die Reduktion

$$\mathbf{c}'' = \mathbf{c}_2' + r_s \mathbf{c}_1' = \mathbf{c}_1 + r_s \mathbf{c}_2.$$

Es ist

$$||\mathbf{c}_2''||^2 = ||\mathbf{c}_1||^2 + ||\mathbf{c}_2||^2 - 2 \cdot |\langle \mathbf{c}_1, \mathbf{c}_2 \rangle| \geq ||\mathbf{c}_2||^2 = ||\mathbf{c}_1'||^2,$$

und damit $t = s$. □

III) Beweis von Satz 5.20: Der Satz wird unter Verwendung der Lemmata 5.25 und 5.27, welche im Anschluß bewiesen werden, geführt.

Bezeichne

$$I_1 = \{\tau \in \{3, \ldots, T\} : l_\tau \neq 0\}$$

und

$$l_t = \sum_{u=1}^{3} \sum_{i=1}^{3} \# LR_i(u,t).$$

Es gilt $H(C) \geq 1$ für jede im Verlauf des TPR-Algorithmus berechnete Matrix C. Damit ist, gemäß den unten bewiesenen Ungleichungen (5.13), (5.14) und (5.17),

$$H(A) \cdot 2^{-l_2} \cdot 3^{-p} \cdot \prod_{\tau=3}^{T} \left(3^{-\tau+2}\right)^{l_\tau} \geq 1.$$

Daraus folgt

$$\log_2 H(A) \geq l_2 + p + \sum_{\tau \in I_1} l_\tau \cdot (\tau - 2)$$

und

$$\log_3 H(A) \geq p + \sum_{\tau \in I_1} l_\tau \cdot (\tau - 2) \geq \sum_{\tau \in I_1} l_\tau.$$

Damit ist

$$2\,l + 2\,p + \sum_{\tau=3}^{T} \tau \cdot l_\tau = 2\,(l_2 + p) + \sum_{\tau \in I_1} \tau \cdot l_\tau \leq 2 \cdot \left(\log_2 H(A) + \sum_{\tau \in I_1} l_\tau\right) \leq 2\,(\log_2 H(A) + \log_3 H(A)).$$

Es folgen nun die für den gerade geführten Beweis erforderlichen Lemmata. Dies sind Lemmata, die Aussagen über den Gaußalgorithmus bei Eingabe einer zulässigen Matrix beinhalten, also die Standardsituation im TPR-Algorithmus.

Lemma 5.25 *Sei $1 \leq u \leq 3$, $2 \leq t \leq T$. Bezeichne $C = (\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3)$ eine Matrix im TPR-Algorithmus vor und $C' = (\mathbf{c}'_1, \mathbf{c}'_2, \mathbf{c}'_3)$ die Matrix nach dem Aufruf eines* **Gauß**$_u$*-Algorithmus, dessen Kohärenzzahlenabschnitt vom Typ LR und der Länge t ist. Es gilt: Es gilt:*

$$H(C') \leq \frac{1}{2} \cdot H(C), \text{ falls } t = 2 \tag{5.13}$$

und

$$H(C') \leq 3^{-t+2} \cdot H(C), \text{ falls } t > 2. \tag{5.14}$$

Beweis: Sei (i,j) das Paar von Indizes mit $1 \leq i < j \leq 3$, so daß $u = i + j - 2$ gilt. Für $t > 2$ kann ein Resultat aus [35], Seite C 13 verwendet werden. Damit ergibt sich

$$||\mathbf{c}'_j||^2 \leq 3^{-t+2} \cdot ||\mathbf{c}_j||^2 \text{ für } t > 2,$$

woraus

$$\prod_{l=1}^{3} ||\mathbf{c}'_l||^2 \leq 3^{-t+2} \cdot \prod_{l=1}^{3} ||\mathbf{c}_l||^2$$

folgt.

Für den Fall, daß $t = 2$ gilt, benötigen wir das Lemma 5.26.

Dieses Lemma beleuchtet eine Ursache für die Effektivität des TPR-Algorithmus. Kohärenzzahlen, die einen Absolutbetrag größer eins haben induzierten eine signifikante Längenkorrelation. Dieses Lemma begründet auch die Wahl des Namens *Kohärenzzahl.*

Lemma 5.26 (Verhältnis-Lemma) *Seien* $\mathbf{c}_1$ *und* $\mathbf{c}_2$ *zwei linear unabhängige Vektoren und sei*

$$r = -\left\lceil \frac{\langle \mathbf{c}_1, \mathbf{c}_2 \rangle}{\| \mathbf{c}_1 \|^2} \right\rfloor .$$

Dann gilt

$$||\mathbf{c}_2||^2 > (r^2 - |r|) \cdot ||\mathbf{c}_1||^2. \tag{5.15}$$

Beweis: Offenbar ist

$$\left| \frac{\langle \mathbf{c}_1, \mathbf{c}_2 \rangle}{\| \mathbf{c}_1 \|^2} \right| \geq |r| - \frac{1}{2} = \frac{2\,|r| - 1}{2},$$

und somit

$$2 \cdot |\langle \mathbf{c}_1, \mathbf{c}_2 \rangle| \geq (2\,|r| - 1)\,||\mathbf{c}_1||^2. \tag{5.16}$$

Ferner ist $||\mathbf{c}_2 + r\mathbf{c}_1||^2 > 0$, da $\mathbf{c}_1$ und $\mathbf{c}_2$ linear unabhängig sind. Es ist

$$||\mathbf{c}_2||^2 > 2\,|r|\,|\langle \mathbf{c}_1, \mathbf{c}_2 \rangle| - r^2||\mathbf{c}_1||^2,$$

also gilt wegen Ungleichung (5.16)

$$||\mathbf{c}_2||^2 > |r|\,(2\,|r| - 1)\,||\mathbf{c}_1||^2 - r^2||\mathbf{c}_1||^2 \geq (r^2 - |r|)\,||\mathbf{c}_1||^2.$$

□

Es wird jetzt ausgeführt, wie die Aussage des Lemmas 5.25 für $t = 2$ folgt.

Seien wiederum (i, j) Indizes mit $i + j - 2 = u$. Wir setzen:

$$\begin{array}{ll} \mathbf{u} = \mathbf{c}_i, & \mathbf{u}' = \mathbf{c}'_i \\ \mathbf{v} = \mathbf{c}_j, & \mathbf{v}' = \mathbf{c}'_j. \end{array}$$

Bezeichne (ρ_1, ρ_2) einen Kohärenzzahlenabschnitt vom *LR*-Typ. Dann ist

$$\max\{|\rho_1|, |\rho_2|\} \geq 2.$$

Ein Gaußalgorithmus berechnet bei Eingabe von $(\mathbf{u}, \mathbf{v})$:

Zeitpunkt	aktuell berechnete Spaltenvektoren
vor dem ersten Gaußschritt	$(\mathbf{u}, \mathbf{v})$
vor dem zweiten Gaußschritt	$(\underbrace{\mathbf{v} + \rho_1\,\mathbf{u}}_{=\mathbf{u}'}, \mathbf{u})$
nach dem zweiten Gaußschritt	$(\mathbf{u}', \underbrace{\mathbf{u} + \rho_2\,\mathbf{u}'}_{=\mathbf{v}'})$

Definiere

$$M(\rho) = \max\{1, \rho^2 - |\rho|\} \text{ für } \rho \in \mathbb{R}.$$

Dann ist nach dem Verhältnislemma 5.26

$$||\mathbf{v}||^2 \geq M(\rho_1) \cdot ||\mathbf{u}||^2 \text{ und}$$

$$||\mathbf{u}||^2 \geq M(\rho_2) \cdot ||\mathbf{u}'||^2, \text{ also}$$

$$||\mathbf{v}||^2 \geq M(\rho_1) \cdot M(\rho_2) \cdot ||\mathbf{u}'||^2 \geq 2 \cdot ||\mathbf{u}'||^2.$$

Da nach dem Verkürzungslemma 5.22

$$||\mathbf{v}'||^2 < ||\mathbf{u}||^2 \text{ gilt, folgt nun}$$

$$||\mathbf{u}||^2 \cdot ||\mathbf{v}||^2 > 2 \cdot ||\mathbf{u}'||^2 \cdot ||\mathbf{v}'||^2 \text{ und schließlich}$$

$$H(C') < \frac{1}{2} \cdot H(C).$$

Damit ist Ungleichung (5.13) und somit Lemma 5.25 bewiesen. □

Wir benötigen noch Lemma 5.27, zu dessen Beweis der im Anschluß gezeigte Hilfssatz 5.28 erforderlich ist.

Lemma 5.27 *Sei $1 \leq u \leq 3$ und bezeichne $C = (\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3)$ die Matrix im TPR-Algorithmus unmittelbar vor und $C' = (\mathbf{c}'_1, \mathbf{c}'_2, \mathbf{c}'_3)$ die Matrix unmittelbar nach dem Auftreten eines* **Gauß**$_u$*-Aufrufs, dessen Kohärenzzahlenabschnitt vom Typ PC ist. Es gilt:*

$$H(C') \leq \frac{1}{3} \cdot H(C) \tag{5.17}$$

Beweis: Bezeichne (ρ_1, ρ_2) das Kohärenzzahlenpaar, das zu dem **Gauß**$_l$-Aufruf gehört. Es sei

$$\begin{array}{rclcrcl} \mathbf{u} & = & \mathbf{c}_i, & & \mathbf{v} & = & \mathbf{c}_j \\ \mathbf{u}' & = & \mathbf{c}'_i, & & \mathbf{v}' & = & \mathbf{c}'_j \end{array}$$

mit $i + j - 2 = l$ und $1 \leq i < j \leq 3$. Ist $k = \{1, 2, 3\} - \{i, j\}$, so ist $\mathbf{c}_k = \mathbf{c}'_k$. Es genügt daher zu zeigen, daß

$$||\mathbf{u}'||^2 \cdot ||\mathbf{v}'||^2 \leq \frac{1}{3} \cdot ||\mathbf{u}||^2 \cdot ||\mathbf{v}||^2$$

gilt. Das Vektorpaar $(\mathbf{u}', \mathbf{v}')$ erfüllt die Prämissen von Hilssatz 5.28. Ferner gilt

$$\begin{pmatrix} \mathbf{u} \\ \mathbf{v} \end{pmatrix} = \begin{pmatrix} -1 & 1 \\ -2 & 1 \end{pmatrix} \cdot \begin{pmatrix} \mathbf{u}' \\ \mathbf{v}' \end{pmatrix}, \text{ falls } (\rho_1, \rho_2) = (1, 1)$$

beziehungsweise

$$\begin{pmatrix} \mathbf{u} \\ \mathbf{v} \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 2 & 1 \end{pmatrix} \cdot \begin{pmatrix} \mathbf{u}' \\ \mathbf{v}' \end{pmatrix}, \text{ falls } (\rho_1, \rho_2) = (-1, -1).$$

Daher ist nach Hilfssatz 5.28

$$\begin{aligned} ||\mathbf{v}||^2 = ||\mathbf{v}' \pm 2\,\mathbf{u}'||^2 &\geq ||\mathbf{v}'||^2 + 2\,||\mathbf{u}'||^2 \\ &\geq 3\,||\mathbf{u}'||^2. \end{aligned}$$

Da $(\mathbf{u}', \mathbf{v}')$ Gauß-reduziert sind, ist

$$\begin{aligned} ||\mathbf{u}||^2 = ||\mathbf{u}' \pm \mathbf{v}'||^2 &\geq ||\mathbf{u}'||^2 + ||\mathbf{v}'||^2 - 2\,|\langle \mathbf{u}', \mathbf{v}' \rangle| \\ &\geq ||\mathbf{v}'||^2. \end{aligned}$$

Damit ist das Lemma gezeigt. □

Hilfssatz 5.28 *Seien die linear unabhängigen Vektoren* $\mathbf{c}_1, \mathbf{c}_2 \in \mathbb{R}^{d \times 2}$ *mit* $||\mathbf{c}_1|| \leq ||\mathbf{c}_2||$ *Gauß-reduziert. Dann gilt*

$$||2\mathbf{c}_1 + \mathbf{c}_2||^2 \geq ||\mathbf{c}_2||^2 + 2\,||\mathbf{c}_1||^2.$$

Beweis: Da die Vektoren Gauß-reduziert sind, gilt:

$$-2\,|\langle \mathbf{c}_1, \mathbf{c}_2 \rangle| \geq -||\mathbf{c}_1||^2$$

und deshalb ist

$$\begin{aligned} ||2\,\mathbf{c}_1 + \mathbf{c}_2||^2 &= 4\,||\mathbf{c}_1||^2 + ||\mathbf{c}_2||^2 + 4\,\langle \mathbf{c}_1, \mathbf{c}_2 \rangle \\ &\geq 4\,||\mathbf{c}_1||^2 + ||\mathbf{c}_2||^2 - 4\,|\langle \mathbf{c}_1, \mathbf{c}_2 \rangle| \\ &\geq 4\,||\mathbf{c}_1||^2 + ||\mathbf{c}_2||^2 - 2\,||\mathbf{c}_1||^2 \\ &\geq ||\mathbf{c}_2||^2 + 2\,||\mathbf{c}_1||^2. \end{aligned}$$

□

IV) Beweis der Ungleichungen (5.9) bis (5.11): Wir beginnen mit dem Beweis von Ungleichung (5.11). Aus dieser Ungleichung sowie den bereits bewiesenen Ergebnissen von Satz 5.16 werden die Ungleichungen (5.9) und (5.10) abgeleitet.

Für die folgenden Untersuchungen gelten die hier angegebenen Bezeichnungen für die im TPR-Algorithmus aufeinander folgend berechneten Matrizen eines kompletten Durchlaufs.

Zeitpunkt	Bezeichnung der Matrix
vor **Gauß**$_3$	$(\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3)$
nach **Sort**	$(\mathbf{c}_1', \mathbf{c}_2', \mathbf{c}_3')$
nach **Gauß**$_1$	$(\mathbf{c}_1'', \mathbf{c}_2'', \mathbf{c}_3'')$
nach **Gauß**$_2$ in Anweisung (3)	$(\mathbf{c}_1''', \mathbf{c}_2''', \mathbf{c}_3''')$
vor **Gauß**$_2$ in Anweisung (6)	$(\mathbf{c}_1^{iv}, \mathbf{c}_2^{iv}, \mathbf{c}_3^{iv})$
vor **Gauß**$_3$	$(\mathbf{c}_1^{v}, \mathbf{c}_2^{v}, \mathbf{c}_3^{v})$

Satz 5.29 *Sei* $A = (\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3)$ *eine zulässige Eingabe des TPR-Algorithmus. Es gilt*

$$\#USR(3) \leq 1 + 2\,(\log_2 ||\mathbf{a}_3|| + \log_4 ||\mathbf{a}_3||).$$

Beweis: Es bezeichne

$$(\rho_{11}, \ldots, \rho_{1\tau_1}, \ldots, \rho_{i1}, \ldots, \rho_{i\tau_i}, \ldots \rho_{l1}, \ldots, \rho_{l\tau_l}) \tag{5.18}$$

die Konkatenation von Kohärenzzahlenabschnitten $\rho_{ij}\,(1 \leq i \leq l, 1 \leq j \leq \tau_i)$, vom Typ USR, im **Gauß**$_3$-Algorithmus in der Reihenfolge ihres Auftretens.

Es bezeichne $\mathbf{v}_i\,(1 \leq i \leq l)$ den dritten Spaltenvektor in der Matrix, die der TPR-Algorithmus bis unmittelbar vor der Berechnung der Kohärenzzahl ρ_{i1} reduziert hat.

Es wird nun erklärt, warum (5.18) in Sequenzen

$$(\rho_{i1}, \ldots, \rho_{i\tau_i}) \tag{5.19}$$

unterteilt wurde. Dann beweisen wir, daß

$$\#USR(3) = \sum_{i=1}^{l} \sum_{j=1}^{\tau_i} 1 \leq 1 + 2\,(\log_2 ||\mathbf{a}_3|| + \log_4 ||\mathbf{a}_3||)$$

gilt.

Wir haben die aufeinanderfolgenden Kohärenzzahlenabschnitte im **Gauß**$_3$-Algorithmus in sogenannte *Tauschfreiabschnitte* unterteilt. Ein *Tauschfreiabschnitt* $(\rho_{i1}, \ldots, \rho_{i\tau_i})$ ist die Konkatenation von aufeinanderfolgenden Kohärenzzahlenabschnitten (ρ_{ij}), $(1 \leq i \leq l, 1 \leq j \leq \tau_i)$ bis jeweils zu dem Zeitpunkt, an dem eine Operation des TPR-Algorithmus die erste oder die zweite Spalte der zu reduzierenden Matrix verändert hat.

Sei im folgenden i beliebig aber fest, mit $1 \leq i \leq l$. Mit der zuvor eingeführten Notation können wir die Situationen, die bei Aufruf des **Gauß**$_3$-Algorithmus bei Auftreten einer USR Abschnittsfolge auftreten, wie folgt charakterisieren.

Situation	$(\mathbf{c}'_1, \mathbf{c}'_2, \mathbf{c}'_3) =$
1)	$(\mathbf{c}_1, \mathbf{c}_3 + \rho_{i1}\,\mathbf{c}_2, \mathbf{c}_2)$
$2a$)	$(\mathbf{c}_1, \mathbf{c}_3 + \rho_{i1}\,\mathbf{c}_2, \mathbf{c}_2)$
$2b$)	$(\mathbf{c}_3 + \rho_{i1}\,\mathbf{c}_2, \mathbf{c}_1, \mathbf{c}_2)$

Für Fall $2a$) und $2b$) hat der Tauschfreiabschnitt die Länge $\tau_i = 1$, und es gilt nach dem Verhältnislemma:

$$||\mathbf{c}_3||^2 = (\rho_{i1}^2 - |\rho_{i1}|)\,||\mathbf{c}_2||^2 = (\rho_{i1}^2 - |\rho_{i1}|)\,||\mathbf{c}'_3||^2.$$

Damit ist $||\mathbf{c}'_3||^2 \leq \frac{1}{2}\,||\mathbf{c}_3||^2$.

Wir werden nun den Fall 1) näher untersuchen und die Ungleichung

$$||\mathbf{v}_{i+1}||^2 \leq 2^{-\tau_i} ||\mathbf{v}_i||^2\,(1 \leq i < l) \tag{5.20}$$

herleiten, die für die Fälle $2a$) und $2b$) soeben bewiesen wurde.

Fall 1) ist derart, daß im darauffolgenden **Gauß**$_1$-Aufruf immer ein CSR_1-Abschnitt mit verschwindender Kohärenzzahl berechnet wird, da $(\mathbf{c}_1, \mathbf{c}_2)$ Gauß-reduziert sind. Ist der Kohärenzzahlenabschnitt des nun folgenden **Gauß**$_3$-Algorithmus nicht vom Typ USR oder CSR_1, so ist $\tau_i = 1$ und es gilt:

$$||\mathbf{c}_3||^2 \geq (\rho_{i1}^2 - |\rho_{i1}|) \cdot ||\mathbf{c}_2||^2 \geq 2 \cdot ||\mathbf{c}_3'''||^2$$

und damit (5.20). Somit bleiben als einzige Fälle, bei denen $\tau_i > 1$ gilt, die übrig, bei denen die Kohärenzzahlenabschnitte (σ_{i1}) im **Gauß**$_2$-Aufruf vom Typ USR oder CSR_1 sind. Wir führen die Bezeichnung (σ_{ij}) für derartige Kohärenzzahlenabschnitte ein, die nach dem Auftreten von (ρ_{ij}) im **Gauß**$_3$ berechnet werden. Wir behaupten:

Lemma 5.30

$$|\sigma_{ij}| < 1 + \frac{|\rho_{ij}|}{2} \quad \textit{und falls} \quad \tau_i \geq 2 \tag{5.21}$$

$$|\rho_{i,j+1}| < 1 + \frac{|\sigma_{ij}|}{2} \quad \textit{und folglich} \tag{5.22}$$

$$|\rho_{i,j+1}| < \frac{3}{2} + \frac{|\rho_{ij}|}{4}. \tag{5.23}$$

Beweis:

Zu Ungleichung (5.21) Es ist $(\mathbf{c}_1'', \mathbf{c}_2'', \mathbf{c}_3'') = (\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3 + \rho_{ij}\mathbf{c}_2)$, wobei $(\mathbf{c}_1'', \mathbf{c}_2'')$ und $(\mathbf{c}_2'', \mathbf{c}_3'')$ Gauß-reduziert sind. Daher gilt:

$$\begin{aligned} |\sigma_{ij}| &= \left|\left\lfloor \tfrac{\langle \mathbf{c}_1'', \mathbf{c}_3''\rangle}{\langle \mathbf{c}_1'', \mathbf{c}_1''\rangle} \right\rceil\right| &&= \left|\left\lfloor \tfrac{\langle \mathbf{c}_1'', \mathbf{c}_3 + \rho_{ij}\mathbf{c}_2\rangle}{\langle \mathbf{c}_1'', \mathbf{c}_1''\rangle} \right\rceil\right| \\ &= \left|\left\lfloor \tfrac{\langle \mathbf{c}_1, \mathbf{c}_3\rangle}{\langle \mathbf{c}_1, \mathbf{c}_1\rangle} + \rho_{ij} \cdot \tfrac{\langle \mathbf{c}_2, \mathbf{c}_3\rangle}{\langle \mathbf{c}_1, \mathbf{c}_1\rangle} \right\rceil\right| &&< 1 + \tfrac{|\rho_{ij}|}{2} \end{aligned}$$

Zu Ungleichung (5.22) Die Paare $(\mathbf{c}_1'', \mathbf{c}_2'')$ sowie $(\mathbf{c}_2'', \mathbf{c}_3'')$ sind Gauß-reduziert, also $||\frac{1}{\mathbf{c}_2''}|| \leq ||\frac{1}{\mathbf{c}_1''}||$ und damit

$$\begin{aligned} |\rho_{i,j+1}| &= \left|\left\lfloor \tfrac{\langle \mathbf{c}_2'', \mathbf{c}_3 + \sigma_{ij}\mathbf{c}_1\rangle}{\langle \mathbf{c}_2'', \mathbf{c}_2''\rangle} \right\rceil\right| \\ &\leq \left|\left\lfloor \tfrac{\langle \mathbf{c}_2'', \mathbf{c}_3''\rangle}{\langle \mathbf{c}_2'', \mathbf{c}_2''\rangle} + \sigma_{ij} \cdot \tfrac{\langle \mathbf{c}_2, \mathbf{c}_3\rangle}{\langle \mathbf{c}_1'', \mathbf{c}_1''\rangle} \right\rceil\right| \\ &< 1 + \tfrac{|\sigma_{ij}|}{2}. \end{aligned}$$

Zu Ungleichung (5.23) Aus (5.21) und (5.22) folgt (5.23). □

Ein Szenario für eine Tauschfreifolge von parametrisierbarer Länge wurde in einem Beispiel bereits gegeben.

Sei $\tau_i \geq 2$. Dazu definieren wir die Folge

$$s_k = 4s_{k-1} - 5 \text{ mit } s_0 = 2.$$

Es gilt:

$$\begin{array}{lcl} |\rho_{i,\tau_i}| & \geq & 2 = s_0 \\ |\rho_{i,\tau_i-1}| & > & 4\,|\rho_{i,\tau_i}| - 6 \end{array}$$

Letzteres gilt wegen (5.23). Da $\rho_{i,\tau_i-1} \in \mathbb{Z}$ ist

$$|\rho_{i,\tau_i-1}| \geq 4\,|\rho_{i,\tau_i}| - 5 \geq s_1.$$

Durch vollständige Induktion auf k ergibt sich

$$|\rho_{i,\tau_i-k}| \geq s_k \quad 0 \leq k \leq \tau_i - 1.$$

Die geschlossene Form von s_k ergibt sich zu

$$s_k = 4^k s_0 - 5\sum_{n=0}^{k-1} 4^n = \frac{4^k}{3} + \frac{5}{3}.$$

Folglich ist:

$$(\rho_{i1}^2 - |\rho_{i1}|) \geq |\rho_{i1}| \geq s_{\tau_i-1} > 2^{\tau_i} \tag{5.24}$$

für $\tau_i \geq 4$.

Für $\tau_i \in \{1, 2, 3\}$ rechnen wir nach:

$$\begin{array}{lcrcccc} s_0 & = & 2 & \geq & 2^1 & & \\ s_1 & = & 6 & \geq & 2^2 & = & 4 \\ s_2 & = & 22 & \geq & 2^3 & = & 8 \end{array}$$

Damit ergibt sich auch für Tauschfreifolgen der Länge $\tau_i > 1$ analog zu Fällen $2a, b)$ mit (5.24): $||\mathbf{v}_i||^2 \geq 2^{\tau_i}||\mathbf{v}_{i+1}||^2$ und damit

$$1 \leq ||\mathbf{v}_l||^2 \leq 2^{-\sum_{i=1}^{l-1}\tau_i}||\mathbf{v}_1||^2$$

also

$$\sum_{i=1}^{l-1} \tau_i \leq 2\,\log_2 ||\mathbf{a}_3||.$$

Wegen Ungleichung (5.23) ist $\tau_l \leq 1 + 2\,\log_4 ||\mathbf{a}_3||$ und somit folgt insgesamt

$$\#USR(3) \leq 1 + 2\,(\log_2 ||\mathbf{a}_3|| + \log_4 ||\mathbf{a}_3||)\,.$$

$\square$

Beweis von Ungleichung (5.9): Wir beweisen nun die Schranke aus Ungleichung (5.9). Dazu studieren wir alle Kombinationen von Kohärenzzahlenabschnitten im **Gauß**$_3$-Aufrufe, denen ein **Gauß**$_1$-Aufruf mit Kohärenzzahlenabschnittstyp USR folgt. Die nachstehende Tabelle gibt einen Überblick über die zu untersuchenden Fälle.

Kohärenzzahlenabschnittstyp in **Gauß**$_3$	obere Schranke für $USR(1)$-Abschnitte
$USR(3)$	0
$LR(3)$ oder $PCR(3)$	$\#LR(3) + \#PCR(3)$
$CSR(1)$	0
$CSR(2)$	$2 \log_2 \|\|\mathbf{a}_3\|\|$

Die Fälle werden nun einzeln abgehandelt:

1. Fall: Der Kohärenzzahlenabschnitt im **Gauß**$_3$-Algorithmus sei vom Typ USR. Offenbar sind dann die Spalten $(1,2)$ im darauffolgenden **Gauß**$_1$-Algorithmus bereits Gauß-reduziert und damit ist der Kohärenzzahlenabschnitt nicht vom Typ USR sondern vom Typ CSR.

2. Fall: Ist der Kohärenzzahlenabschnitt im **Gauß**$_3$-Algorithmus vom Typ LR oder PCR, so ist die Anzahl der Typ USR Abschnitte im darauffolgenden **Gauß**$_1$-Algorithmus naturgemäß durch $\#LR(3) + \#PCR(3)$ beschränkt. Dafür wurden bereits Schranken hergeleitet.

3. Fall: Ist der **Gauß**$_3$-Kohärenzzahlenabschnitt vom Typ CSR_1, so kann kann mit dem gleichen Argument wie im 1. Fall die Nichtexistenz von einem nachfolgenden $USR(1)$-Abschnittstyp nachgewiesen werden.

4. Fall: Sei $|\sigma| = 1$. Es können zwei Unterfälle eintreten:

a) $(\mathbf{c}'_1, \mathbf{c}'_2, \mathbf{c}'_3) = (\mathbf{c}_1, \mathbf{c}_3 + \sigma \cdot \mathbf{c}_1, \mathbf{c}_2)$: Sei r die erste Kohärenzzahl des auf den **Gauß**$_3$-Algorithmus folgenden **Gauß**$_1$-Abschnitts. Es ist:

$$|r| = \left\lceil \frac{\langle \mathbf{c}_1, \mathbf{c}'_2 \rangle}{||\mathbf{c}_1||^2} \right\rfloor \leq 1.$$

Damit ist gezeigt, daß dieser Abschnitt nicht vom Typ USR ist.

b) $(\mathbf{c}'_1, \mathbf{c}'_2, \mathbf{c}'_3) = (\mathbf{c}_3 + \sigma \cdot \mathbf{c}_1, \mathbf{c}_1, \mathbf{c}_2)$: Sei (r) vom Typ USR der Kohärenzzahlenabschnitt des auf den **Gauß**$_3$-Aufruf folgenden **Gauß**$_1$-Aufrufs. Dann gilt nach dem Verhältnislemma

$$||\mathbf{c}_1||^2 \geq (r^2 - |r|) \cdot ||\mathbf{c}'_1||^2 \geq 2 \cdot ||\mathbf{c}'_1||^2.$$

Folglich kann dies insgesamt nur $2 \log_2 ||\mathbf{a}_3||$ - mal passieren.

Somit ist die Schranke aus Ungleichung (5.9) bewiesen worden.

Beweis von Ungleichung (5.10) Zum Beweis von Ungleichung (5.10) benötigen wir den folgenden Hilfssatz.

Hilfssatz 5.31 *Sei ρ eine ganze Zahl mit $|\rho| > 1$ und sei $C = (\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3)$ eine zulässige Matrix. Ferner sei $C' = (\mathbf{c}'_1, \mathbf{c}'_2, \mathbf{c}'_3)$ eine zulässige Matrix mit $L(C') = L(C)$ und*

$$||\mathbf{c}'_3||^2 \geq (\rho^2 - |\rho|)\, ||\mathbf{c}'_1||^2$$

und $||\mathbf{c}_2'|| \leq ||\mathbf{c}_1||$ *sowie* $||\mathbf{c}_3'|| \leq ||\mathbf{c}_2||$. *Dann gilt:*

$$H(C') \leq \frac{1}{2} H(C).$$

Beweis: Es ist

$$H(C) \geq (\rho^2 - |\rho|)\, ||\mathbf{c}_1'||^2\, ||\mathbf{c}_1||^2\, ||\mathbf{c}_3||^2 \geq 2\, H(C').$$

□

Um obere Schranken für $\#USR(2)$ herzuleiten, sind zwei Fälle zu untersuchen. Zum einen muß untersucht werden, wie oft USR-Abschnitte im **Gauß**$_2$-Algorithmus von Anweisung (3), zum anderen, wie oft USR-Abschnitte im **Gauß**$_2$-Algorithmus von Anweisung (6) auftreten können.

Eine obere Schranke für USR-Abschnitte in Anweisung (3): Es wird untersucht, wie oft in Anweisung (3) ein USR-Abschnitt auftritt, wenn im **Gauß**$_3$-Aufruf zuvor einer der folgenden Abschnittstypen auftritt:

a) USR

b) LR oder PCR

c) CSR_1

d) CSR_2

Es folgt die detaillierte Behandlung der Fälle.

a) Offenbar kann nach Auftreten eines USR-Abschnitts in **Gauß**$_3$ nur ein CSR_1-Abschnitt im nachfolgenden **Gauß**$_1$-Algorithmus auftreten, denn es ist

$$(\mathbf{c}_1', \mathbf{c}_2', \mathbf{c}_3') = (\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3 + \rho\, \mathbf{c}_2)$$

und $(\mathbf{c}_1, \mathbf{c}_2)$ sind Gauß-reduziert. Damit sind die danach auftretenden USR-Abschnitte durch $\#USR(3)$ beschränkt.

b) Für diese Situation ist trivialerweise $\#LR + \#PCR$.

c) Hierbei gilt mit $\sigma \in \{\pm 1\}$:

$$(\mathbf{c}_1'', \mathbf{c}_2'', \mathbf{c}_3'') = (\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3 + \sigma\, \mathbf{c}_2).$$

Bezeichne ρ die erste Kohärenzzahl des Kohärenzzahlenabschnitts in **Gauß**$_2$ von Anweisung (3), dann gilt:

$$|\rho| = \left| \left\lfloor \frac{\langle \mathbf{c}_1, \mathbf{c}_3 + \sigma\, \mathbf{c}_2 \rangle}{\langle \mathbf{c}_1, \mathbf{c}_1 \rangle} \right\rceil \right| \leq 1,$$

da $(\mathbf{c}_1, \mathbf{c}_2)$ und $(\mathbf{c}_1, \mathbf{c}_3)$ Gauß-reduziert sind.

d) Sei wieder $\sigma \in \{\pm 1\}$. Es können zwei Fälle eintreten:

Fall	$(\mathbf{c}_1'', \mathbf{c}_2'', \mathbf{c}_3'') =$
$i)$	$(\mathbf{c}_1, \mathbf{c}_3 + \sigma\, \mathbf{c}_2, \mathbf{c}_2)$
$ii)$	$(\mathbf{c}_3 + \sigma\, \mathbf{c}_2, \mathbf{c}_1, \mathbf{c}_2)$

Ist ρ die erste Kohärenzzahl der Kohärenzzahlenabschnitts von **Gauß**$_2$, so ist in Fall i) $\rho = 0$, denn $(\mathbf{c}_1, \mathbf{c}_2)$ sind Gauß-reduziert. In Fall ii) ist für $|\rho| > 1$ nach Hilfssatz 5.31

$$H(C'') \leq \frac{1}{2}\, H(C).$$

Eine obere Schranke für USR-Abschnitte in Anweisung (6): Hier gilt $||\mathbf{c}_1^{iv}|| < ||\mathbf{c}_1'||$, $||\mathbf{c}_2^{iv}|| \leq ||\mathbf{c}_1'||$, $||\mathbf{c}_3^{iv}|| \leq ||\mathbf{c}_2||$. Ferner bezeichne ρ die erste Kohärenzzahl im **Gauß**$_2$-Algorithmus von Anweisung (6) mit $|\rho| > 1$. Dann gilt nach dem Verhältnislemma 5.26 $||\mathbf{c}_3^{iv}||^2 \geq 2\, ||\mathbf{c}_1^{iv}||^2$. Damit folgt

$$H(C^{iv}) \leq \frac{1}{2}\, H(C').$$

Fazit:

$$\#USR(2) \leq \#USR(3) + \#LR + \#PCR + 6\, \log_2 ||\mathbf{a}_3||.$$

5.7 Analyse II

In diesem Abschitt wird die Ungleichung (5.12) von Satz 5.16 bewiesen. Dazu analysieren wir die Situation, daß ab einem bestimmten Zeitpunkt viele aufeinanderfolgende Kohärenzzahlenabschnitte ausnahmslos vom Typ CSR sind. Diese Kohärenzzahlen nennen wir im folgenden kritische Kohärenzzahlen. Es wird nun der folgende Satz bewiesen.

Satz 5.32 *Sei A die Eingabematrix, oder die Matrix, welche im Verlauf des TPR-Algorithmus aus der Eingabe berechnet wurde. Sei A derart, daß wenigstens die folgenden 22 Gaußschritte zur Reduktion von A zu Kohärenzzahlenabschnitten vom Typ CSR_1 oder CSR_2 gehören. Bezeichne A' die Matrix, die aus A nach dem Auftreten von 22 Kohärenzzahlen des genannten Typs berechnet wurde. Es gilt:*

$$H(A') \leq \frac{1}{7} \cdot H(A). \tag{5.25}$$

Beweis: Durch Betrachtung aller Fälle ergibt sich, daß nach der Berechnung von höchstens vier aufeinanderfolgenden Kohärenzzahlen die zu Kohärenzzahlenabschnitten vom Typ CSR_1 oder CSR_2 gehören, eine der folgenden Situationen vorliegt:

I) die Spalten $(1,2)$ und $(1,3)$ der Matrix des TPR-Algorithmus sind Gauß-reduziert und der Algorithmus befindet sich vor Anweisung (10)

II) die Spalten $(1,2)$ und $(2,3)$ der Matrix des TPR-Algorithmus sind Gauß-reduziert und der Algorithmus befindet sich vor Anweisung (3) oder (6)

Die zuletzt geschilderten Situationen bilden den Ausgangspunkt für den Beweis. In Tabelle 5.5 wurden alle unimodularen Transformationen zusammengestellt durch die Transformationen beschrieben werden, die zu Reduktionsschritten korrespondieren, die nicht trivialerweise zum Terminieren des Algorithmus führen. Die Bezeichnungsweise $(\sigma_1, \sigma_2, \sigma_3)$ bedeutet:

$$\sigma_l = \begin{cases} -1 & \text{falls die Vektoren } (i,j) \text{ mit } l = i+j-2 \text{ nicht Gauß-reduziert sind} \\ 0 & \text{falls die Vektoren } (i,j) \text{ mit } l = i+j-2 \text{ Gauß-reduziert sind} \end{cases}$$

Die Tabelle 5.5 gibt den Reduktionsstatus der Matrix, vor und nach der Reduktionsoperation an.

Lemma 5.33 *Seien $\sigma_1, \sigma_2, \sigma_3$ drei aufeinanderfolgende Kohärenzzahlen, die zu Kohärenzzahlenabschnitten vom CSR_1 oder CSR_2 gehören, und die unmittelbar im Anschluß an eine in I) oder II) geschilderte Situation berechnet werden. Es gilt:*

σ_3 ist eindeutig durch σ_1 und σ_2 bestimmt.

Der Beweis des Lemmas wird durch Nachprüfen von Tabelle 5.6 erbracht. Dort ist zusammengestellt, wie σ_3 aus σ_1 und σ_2 berechnet wird. Wir definieren aus Gründen einer übersichtlichen Notation: $\gamma = \sigma_1 \cdot \sigma_2$.

Computerbeweis Zum Beweis des Satzes 5.32 wurde ein Algorithmus CB implementiert, der die Probleme 1) und 2) löst:

Probleme: Sei $I_1 = (i_5, i_6, \ldots, i_{22})$ und sei

$$(M_{l_i}^{s_i})_{i \in I_1},\, l \in \{1, 2, \ldots, 8\},\, s_i \in \{\pm 1\},$$

die zu den Kohärenzzahlen $(r_i)_{i \in I_1}$ assoziierte Folge unimodularer Transformationen.

1) Generierung einer Menge von Folgen MF, so daß für jede Folge vom mehr als 22 kritischen Kohärenzzahlen, die Teilfolge einer Kohärenzzahlenfolge einer zulässigen Eingabe ist, ein Folge unimodularer Transformationen in MF existiert.

2) Nachweis von Ungleichung (5.25) für jede Folge in MF, dabei sei

$$A^{(i_{22})} = A^{(i_4)} \cdot \prod_{i \in I_1} M_{l_i}^{s_i}.$$

Zur Lösung des Problems 1) wurde eine Ordnung auf der Menge MF wie folgt eingeführt: Seien $\mathbf{l}_j = (l_j^{(1)}, \ldots, l_j^{(18)})\, (1 \leq j \leq 2)$ die Indexfolgen von F_1, $F_2 \in MF$ mit der gleichen Vorzeicheneinstellung in den ersten zwei Positionen. Sei $F_1 \neq F_2$ und $i = \min\{j : l_1^{(j)} \neq l_2^{(j)}\}$. Wir definieren:

$$\begin{array}{l} F_1 < F_2, \text{falls } l_1^{(i)} < l_2^{(i)} \\ F_1 > F_2, \text{falls } l_1^{(i)} > l_2^{(i)}. \end{array}$$

Diese Ordnung ermöglicht es, alle Elemente in MF inkrementell zur erzeugen.

Zum Nachweis von 2) wurde folgendes Lemma benutzt:

Lemma 5.34 *Seien $(\mathbf{c}_1, \mathbf{c}_2)$ Gauß-reduziert. Dann gilt für alle $x, y \in \mathbb{Z}$:*

$$||x\, \mathbf{c}_1 + y\, \mathbf{c}_2||^2 \geq (x^2 - |xy|)\, ||\mathbf{c}_1||^2 + y^2\, ||\mathbf{c}_2||.$$

Durch die Berechnung der inversen Transformation und die jeweilige Kenntnis des Reduktionszustands, wurde mit Lemma 5.34 die Ungleichung (5.25) für alle Fälle nachgewiesen.

$\square$

Start: $(-1, 0, 0)$	
Transformation	Ziel
$M_1^{\pm} = \begin{pmatrix} 1 & \pm 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$	$(0, 0, -1)$
$M_2^{\pm} = \begin{pmatrix} \pm 1 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$	$(0, -1, 0)$
Start: $(0, -1, 0)$	
Transformation	Ziel
$M_3^{\pm} = \begin{pmatrix} 1 & 0 & \pm 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$	$(0, 0, -1)$
$M_4^{\pm} = \begin{pmatrix} 1 & \pm 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$	$(0, 0, -1)$
$M_5^{\pm} = \begin{pmatrix} \pm 1 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$	$(0, -1, 0)$
Start: $(0, 0, -1)$	
Transformation	Ziel
$M_6^{\pm} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & \pm 1 \\ 0 & 0 & 1 \end{pmatrix}$	$(0, -1, 0)$
$M_7^{\pm} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \pm 1 & 1 \\ 0 & 1 & 0 \end{pmatrix}$	$(-1, 0, 0)$
$M_8^{\pm} = \begin{pmatrix} 0 & 1 & 0 \\ \pm 1 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$	$(-1, 0, 0)$

Tabelle 5.5: Transformationen mit kritischen Kohärenzzahlen

1. Transformation	2. Transformation	3. Transformation
$M_1^{\sigma_1}$	$M_6^{\sigma_2}$	M_l^{γ}, $l \in \{3,4,5\}$
$M_1^{\sigma_1}$	$M_7^{\sigma_2}$	M_l^{γ}, $l \in \{1,2\}$
$M_1^{\sigma_1}$	$M_8^{\sigma_2}$	M_l^{γ}, $l \in \{1,2\}$
$M_2^{\sigma_1}$	$M_3^{\sigma_2}$	M_l^{γ}, $l \in \{6,7,8\}$
$M_2^{\sigma_1}$	$M_4^{\sigma_2}$	M_l^{γ}, $l \in \{6,7,8\}$
$M_2^{\sigma_1}$	$M_5^{\sigma_2}$	M_l^{γ}, $l \in \{3,4,5\}$
$M_3^{\sigma_1}$	$M_6^{\sigma_2}$	$M_l^{\sigma_1}$, $l \in \{3,4,5\}$
$M_3^{\sigma_1}$	$M_7^{\sigma_2}$	$M_l^{\sigma_1}$, $l \in \{1,2\}$
$M_3^{\sigma_1}$	$M_8^{\sigma_2}$	$M_l^{\sigma_1}$, $l \in \{1,2\}$
$M_4^{\sigma_1}$	$M_6^{\sigma_2}$	M_l^{γ}, $l \in \{3,4,5\}$
$M_4^{\sigma_1}$	$M_7^{\sigma_2}$	M_l^{γ}, $l \in \{1,2\}$
$M_4^{\sigma_1}$	$M_8^{\sigma_2}$	M_l^{γ}, $l \in \{1,2\}$
$M_5^{\sigma_1}$	$M_3^{\sigma_2}$	M_l^{γ}, $l \in \{6,7,8\}$
$M_5^{\sigma_1}$	$M_4^{\sigma_2}$	M_l^{γ}, $l \in \{6,7,8\}$
$M_5^{\sigma_1}$	$M_5^{\sigma_2}$	M_l^{γ}, $l \in \{3,4,5\}$
$M_6^{\sigma_1}$	$M_3^{\sigma_2}$	$M_l^{\sigma_1}$, $l \in \{6,7,8\}$
$M_6^{\sigma_1}$	$M_4^{\sigma_2}$	$M_l^{\sigma_1}$, $l \in \{6,7,8\}$
$M_6^{\sigma_1}$	$M_5^{\sigma_2}$	$M_l^{\sigma_1}$, $l \in \{3,4,5\}$
$M_7^{\sigma_1}$	$M_1^{\sigma_2}$	$M_l^{\sigma_1}$, $l \in \{6,7,8\}$
$M_7^{\sigma_1}$	$M_2^{\sigma_2}$	$M_l^{\sigma_1}$, $l \in \{3,4,5\}$
$M_8^{\sigma_1}$	$M_1^{\sigma_2}$	M_l^{γ}, $l \in \{6,7,8\}$
$M_8^{\sigma_1}$	$M_2^{\sigma_2}$	M_l^{γ}, $l \in \{3,4,5\}$

Tabelle 5.6: Interdependenzen zwischen kritischen Kohärenzzahlen

5.8 Untersuchung einer Klasse von Eingaben

In diesem Abschnitt werden zulässige Eingaben des TPR-Algorithmus studiert, die die Eigenschaft haben, daß die Repeat-Until Schleife, die aus den Anweisungen (5) bis (7) des TPR-Algorithmus besteht (im folgenden kurz RU genannt) l-mal ($l \in \mathbb{N}$) hintereinander durchlaufen wird und das alle während dieser RU-Durchläufe berechneten Kohärenzzahlen Elemente von $CSR_2(2)$ sind.

Lemma 5.35 *Sei $l \in \mathbb{N}$, $l' = \lfloor l/3 \rfloor$ und $l'' = l - l'$ und C_0 die Matrix, die der TPR-Algorithmus vor Ausführung von RU berechnet hat. Desweiteren bezeichne C_l die Matrix, die aus C_0 nach l RU-Durchläufen mit Kohärenzzahlen in $CSR_2(2)$ berechnet wurde. Dann existieren genau vier Möglichkeiten, wie C_0 durch die in Tabelle 5.5 definierten Matrizen in C_l transformiert werden kann. Diese lauten:*

$$
\begin{aligned}
C_l &= (M_5^+)^l \cdot C && (5.26)\\
C_l &= \Big((M_5^+\, M_5^-\, M_5^-)^{l'} \cdot (M_5^+)^{\epsilon_1} (M_5^-)^{\epsilon_2}\Big) \cdot C && (5.27)\\
C_l &= \Big((M_5^-\, M_5^+\, M_5^-)^{l'} \cdot (M_5^-)^{\epsilon_1} (M_5^+)^{\epsilon_2}\Big) \cdot C && (5.28)\\
C_l &= \Big((M_5^-\, M_5^-\, M_5^+)^{l'} \cdot (M_5^-)^{\epsilon_1} (M_5^-)^{\epsilon_2}\Big) \cdot C && (5.29)
\end{aligned}
$$

Die Werte von ϵ_i $(1 \le i \le 2)$ in Abhängigkeit von l'' wurden in Tabelle 5.7 zusammengestellt.

l''	ϵ_1	ϵ_2
0	0	0
1	1	0
2	1	1

Tabelle 5.7: Untersuchung einer Klasse von zulässigen Eingaben

Für die durch Gleichung (5.26) dargestellte Situation gilt, falls $l \ge 5$:

$$2\,||\mathbf{c}_k|| \le ||\mathbf{c}_{k+5}|| \le 3\,||\mathbf{c}_k|| \text{ für } 1 \le l \le k-5 \qquad (5.30)$$

Beweis: Die Gleichungen (5.28) bis (5.31) wurden bereits gezeigt. Die Ungleichung (5.30) wird wie folgt bewiesen: Das charakterische Polynom der Matrix M_5^+ lautet

$$p(x) = x^3 - x^2 - 1.$$

Die Polynome $p_i(x)\,(1 \le i \le 2)$ faktorisieren wie folgt

$$
\begin{aligned}
p_1(x) &= x^8 - 3\,x^5 - 1 = p(x) \cdot (x^5 + x^4 + x^3 - x^2 + 1) && (5.31)\\
p_2(x) &= x^7 - 2\,x^5 - 1 = p(x) \cdot (x^3 - x + 1) \cdot (x + 1). && (5.32)
\end{aligned}
$$

Daher ist nach dem Satz von Cayley-Hamilton

$$p_i(M_5^+) = 0 \; (1 \leq i \leq 2).$$

Damit folgen die Ungleichungen (5.30) nach der Dreiecksungleichung und dem Verkürzungslemma 5.22, denn es ist

$$||\mathbf{c}_k|| = ||3\,\mathbf{c}_{k+5} - \mathbf{c}_{k+8}|| \geq 2\,||\mathbf{c}_{k+5}||$$

und

$$||\mathbf{c}_k|| = ||\mathbf{c}_{k+7} - 2\,\mathbf{c}_{k+5}|| \leq 3\,||\mathbf{c}_{k+5}||.$$

□

Für die geschilderte Situation folgt aus der Ungleichung (5.30) sowohl die obere als auch die untere Schranke für die Anzahl von Gaußschritten und für die durch die Gleichungen (5.27) bis (5.29) dargestellten Situationen lassen sich analoge Abschätzungen aufstellen.

Wir geben nun noch zwei parametrierte Eingaben für den TPR-Algorithmus an, bei denen sich die in den Gleichungen (5.27) bis (5.29) geschilderten Reduktionsabläufe abspielen. Wir verzichten auf die beiden Beweise dieser Behauptung, da die Beweise ähnlich lang sind, wie der in Kapitel 3 geführte Beweis für die dort gegeben parametrisierte Eingabe zum LLL-Algorithmus.

Ein Repräsentant für die Situation aus Gleichung (5.26)

$$C_0 = \begin{pmatrix} 1 & -1 & -1 \\ 1 & 0 & -2 \\ 0 & 2 & -1 \end{pmatrix} \cdot \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix}^l \text{ mit } l \in \mathbb{N}$$

Ein Repräsentant für die Situationen der Gleichungen (5.27) bis (5.29)

$$C_0 = \begin{pmatrix} -1 & 1 & -1 \\ 0 & 2 & 1 \\ -1 & 0 & -2 \end{pmatrix} \cdot \left(\begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \right)^l \text{ mit } l \in \mathbb{N}$$

Kapitel 6

FTPR, die Verbesserung des TPR-Algorithmus

Der FTPR-Algorithmus (Fast TPR) ist ein Algorithmus zur Berechnung einer Minkowski-reduzierten Basis eines Gitters vom Rang drei. Dies geschieht durch die Reduktion der assoziierten positiv definiten ternären quadratischen Form. Der FTPR-Algorithmus ist eine Weiterentwicklung des TPR-Algorithmus. Die Worst-Case Schranken für die Anzahl der Iterationen ist bei beiden Algorithmen bis auf einen konstanten Faktor gleich. Die Worst-Case Bitkomplexität des FTPR-Algorithmus unter Verwendung klassischer Algorithmen für die elementare Arithmetik, ist quadratisch, wohingegen die Worst-Case Bitkomplexität des TPR-Algorithmus kubisch ist. Der FTPR-Algorithmus besitzt damit eine signifikant bessere asymptotische Bitkomplexität als die bekannten Algorithmen von Brigitte Vallée und Jeff Lagarias.

Zunächst geben wir das Struktogramm des FTPR-Algorithmus an. Dann erklären wir die wesentlichen Unterschiede zu dem schon bekannten TPR-Algorithmus. Schließlich folgt die detaillierte Beschreibung der Funktionen.

Algorithmus 6.1 (FTPR(A))

Eingabe: Eine zulässige Matrix $A = (\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3)$, die Basis eines Gitters L vom Rang drei ist.

Ausgabe: Eine Minkowski-reduzierte Basis des Gitters L.

Initialisiere $t_2 = 0$ und die temporären Variablen $h_i^{(j)}\,(1 \le j \le 2)$, $(1 \le i \le 27)$ und $(\kappa_1, \kappa_2, \kappa_3) = (1, 2, 3)$, g_1, h_1, z_1, z_2, l_3, l_{test}, $d_i^{\min}\,(1 \le i \le 2)$, $r_i\,(1 \le i \le 3)$, $\mathbf{u}, \mathbf{v}$ (1)

Initialisierung von $h_{i1}\,(1 \le i \le 27)$, wie in der Beschreibung des Algorithmus definiert (2)

- **FGauß**$_1$ (3)
- **FGauß**$_2$ (4)
- IF $t_2 > 1$ (5)
- THEN
  - fswap(2,3) (6)
  - **FGauß**$_2$ (7)
  - UNTIL $t_2 \le 1$ (8)
- IF $p_{33} < p_{22}$ (9)
- THEN fswap(2,3) (10)
  - $r_3 = -\lfloor p_{23}/p_{22} \rceil$ (11)
  - IF $|r_3| \le 1$ (12)
  - THEN
    - Exit$= 1$ (13)
    - **SFGauß**$_3$ (14)
    - **FSort** (15)
  - ELSE
    - Exit$= 0$ (16)
    - **optimum** (17)
    - **indsort** (18)
  - UNTIL Exit$= 1$ oder $\kappa_2 \neq 3$ (19)

UNTIL $2 \cdot |p_{12}| \le p_{11}$ und $2 \cdot |p_{13}| \le p_{11}$ und $2 \cdot |p_{23}| \le p_{22}$ (20)

LastStep (21)

Der TPR- und der FTPR-Algorithmus unterscheiden sich in der Anzahl der Hilfsvariablen, die sie benötigen. Beim TPR-Algorithmus werden zwei Sätze zu je 6 Hilfsvariablen verwendet, um die Koeffizienten der des Gitters korrespondierenden Form zu speichern. Beim FTPR-Algorithmus hingegen sind zwei Sätze von je 27 Hilfsvariablen erforderlich. Sie dienen der schnellen Berechnung der für den Algorithmus wichtigen Größe μ_{32} in der Funktion **optimum**. Sie enthalten die Koeffizienten der assoziierten Form und sämtliche Produkte von je zwei Koeffizienten der Form und sind wie folgt definiert:

$$\begin{array}{lll}
h_{1,1} = \langle \mathbf{a}_1, \mathbf{a}_1 \rangle & h_{2,1} = \langle \mathbf{a}_2, \mathbf{a}_2 \rangle & h_{3,1} = \langle \mathbf{a}_3, \mathbf{a}_3 \rangle \\
h_{4,1} = \langle \mathbf{a}_1, \mathbf{a}_2 \rangle & h_{5,1} = \langle \mathbf{a}_1, \mathbf{a}_3 \rangle & h_{6,1} = \langle \mathbf{a}_2, \mathbf{a}_3 \rangle \\
h_{7,1} = \langle \mathbf{a}_1, \mathbf{a}_1 \rangle^2 & h_{8,1} = \langle \mathbf{a}_2, \mathbf{a}_2 \rangle^2 & h_{9,1} = \langle \mathbf{a}_3, \mathbf{a}_3 \rangle^2 \\
h_{10,1} = \langle \mathbf{a}_1, \mathbf{a}_2 \rangle^2 & h_{11,1} = \langle \mathbf{a}_1, \mathbf{a}_3 \rangle^2 & h_{12,1} = \langle \mathbf{a}_2, \mathbf{a}_3 \rangle^2 \\
h_{13,1} = \langle \mathbf{a}_1, \mathbf{a}_1 \rangle \cdot \langle \mathbf{a}_2, \mathbf{a}_2 \rangle & h_{14,1} = \langle \mathbf{a}_1, \mathbf{a}_1 \rangle \cdot \langle \mathbf{a}_3, \mathbf{a}_3 \rangle & h_{15,1} = \langle \mathbf{a}_1, \mathbf{a}_1 \rangle \cdot \langle \mathbf{a}_1, \mathbf{a}_2 \rangle \\
h_{16,1} = \langle \mathbf{a}_1, \mathbf{a}_1 \rangle \cdot \langle \mathbf{a}_1, \mathbf{a}_3 \rangle & h_{17,1} = \langle \mathbf{a}_1, \mathbf{a}_1 \rangle \cdot \langle \mathbf{a}_2, \mathbf{a}_3 \rangle & h_{18,1} = \langle \mathbf{a}_2, \mathbf{a}_2 \rangle \cdot \langle \mathbf{a}_3, \mathbf{a}_3 \rangle \\
h_{19,1} = \langle \mathbf{a}_2, \mathbf{a}_2 \rangle \cdot \langle \mathbf{a}_1, \mathbf{a}_2 \rangle & h_{20,1} = \langle \mathbf{a}_2, \mathbf{a}_2 \rangle \cdot \langle \mathbf{a}_1, \mathbf{a}_3 \rangle & h_{21,1} = \langle \mathbf{a}_2, \mathbf{a}_2 \rangle \cdot \langle \mathbf{a}_2, \mathbf{a}_3 \rangle \\
h_{22,1} = \langle \mathbf{a}_3, \mathbf{a}_3 \rangle \cdot \langle \mathbf{a}_1, \mathbf{a}_2 \rangle & h_{23,1} = \langle \mathbf{a}_3, \mathbf{a}_3 \rangle \cdot \langle \mathbf{a}_1, \mathbf{a}_3 \rangle & h_{24,1} = \langle \mathbf{a}_3, \mathbf{a}_3 \rangle \cdot \langle \mathbf{a}_2, \mathbf{a}_3 \rangle \\
h_{25,1} = \langle \mathbf{a}_1, \mathbf{a}_2 \rangle \cdot \langle \mathbf{a}_1, \mathbf{a}_3 \rangle & h_{26,1} = \langle \mathbf{a}_1, \mathbf{a}_2 \rangle \cdot \langle \mathbf{a}_2, \mathbf{a}_3 \rangle & h_{27,1} = \langle \mathbf{a}_1, \mathbf{a}_3 \rangle \cdot \langle \mathbf{a}_2, \mathbf{a}_3 \rangle .
\end{array}$$

Allen zum TPR-Algorithmus analogen Funktionen des FTPR-Algorithmus wurde ein F vorangestellt. So bezeichnet **FGauß** die zu **Gauß** analoge Funktion. Im Struktogramm von Algorithmus 6.2 sind die zu einem **FGauß**-Schritt gehörenden Anweisungen am Beispiel eines **FGauß**$_1$-Schritt dargestellt.

Algorithmus 6.2 (FGauß$_1$-Schritt)

$\mathbf{a}_2 = \mathbf{a}_2 + x\,\mathbf{a}_1$	(1)
FOR $i = 1$ to 27	(2)
$\quad h_{i2} = h_{i1}$	(3)
$h_{21} = h_{22} + x^2\,h_{12} + 2\,x\,h_{42}$	(4)
$h_{41} = h_{42} + x\,h_{12}$	(5)
$h_{61} = h_{62} + x\,h_{52}$	(6)
$h_{71} = h_{22} + x^2\,h_{12}$	(7)
$h_{81} = h_{82} + x^4\,h_{72} + 4\,x^2\,h_{10,2} + 2\,(x^2\,h_{13,2} + 2\,x\,h_{19,2} + 2\,x^3\,h_{15,2})$	(8)
$h_{10,1} = h_{10,2} + 2\,x\,h_{15,2} + x^2\,h_{7,2}$	(9)
$h_{12,1} = h_{12,2} + 2\,x\,h_{27,2}$	(10)
$h_{13,1} = h_{13,2} + x^2\,h_{7,2} + 2\,x\,h_{15,2}$	(11)
$h_{15,1} = h_{15,2} + x\,h_{7,2}$	(12)
$h_{17,1} = h_{17,2} + x\,h_{16,2}$	(13)
$h_{18,1} = h_{18,2} + x^2\,h_{14,2} + 2\,x\,h_{22,2}$	(14)
$h_{19,1} = h_{19,2} + x^2\,h_{15,2} + 2\,x\,h_{10,2}$	(15)
$h_{20,1} = h_{20,2} + x^2\,h_{16,2} + 2\,x\,h_{25,2}$	(16)
$h_{21,1} = h_{21,2} + x^2\,h_{17,2} + 2\,x\,h_{26,2}$	(17)
$h_{22,1} = h_{22,2} + x\,h_{14,2}$	(18)
$h_{24,1} = h_{24,2} + x\,h_{23,2}$	(19)
$h_{25,1} = h_{25,2} + x\,h_{16,2}$	(20)
$h_{26,1} = h_{26,2} + x\,(h_{25,2} + h_{17,2}) + x^2\,h_{16,2}$	(21)
$h_{27,1} = h_{27,2} + x\,h_{11,2}$	(22)

Abgesehen von den Unterschieden der beiden Algorithmen, die von der Berechnung der Hilfsgrößen herrühren, unterscheiden sich die Algorithmen im ersten Teil nicht. Präziser: Anweisung (2) bis (9) des TPR-Algorithmus entspricht der Folge von Anweisungen (3) bis (10) des FTPR-Algorithmus.

Unterschiede treten auf für Kohärenzzahlenabschnitte im **Gauß$_3$**-Algorithmus vom Typ USR, LR_1, $LR_2(t \geq 3)$ und LR_3.

Ist der Kohärenzzahlenabschnitt des **FGauß$_3$**-Aufrufs vom Typ USR oder LR_2, so wird in die Funktion **optimum** verzweigt, ansonsten in die Funktion **SFGauß$_3$**. Wurde in die Funktion **SFGauß$_3$** verzweigt, so wird die Ausführung nach Abarbeitung dieser Funktion in **SFGauß$_1$** fortgesetzt, falls die Paar-Reduktionsbedingung noch nicht erfüllt ist.

In der Funktion **optimum** werden durch eine kleine Enumeration, von maximal drei Versuchen $(x_1, x_2) \in \mathbb{Z}^2$ bestimmt, so daß der Vektor $w = \mathbf{a}_3 + x_1\mathbf{a}_1 + x_2\mathbf{a}_2$ die Eigenschaft besitzt, daß

$$||\mathbf{w}|| = \min\{||\mathbf{a}_3 + x\mathbf{a}_1 + y\mathbf{a}_2|| : x, y \in \mathbb{Z}\}.$$

Das Paar (x_1, x_2) wird ein *optimales Kohärenzzahlenpaar* genannt. Dieser **optimum**-Schritt führt zu einem Vektor, der wenigstens so kurz ist wie der, der aus der gleichen Eingabe in einem **Gauß**$_2$- oder **Gauß**$_3$-Schritt berechnet würde. Ist $||\mathbf{w}|| \geq ||\mathbf{a}_3||$, so sind die Paare $(\mathbf{w}, \mathbf{a}_i)$ $(1 \leq i \leq 2)$ Gauß-reduziert.

Im Fall, daß **optimum** aufgerufen wird, können nach **indsort** folgende Situationen auftreten:

	Situation	Paare, die garantiert Gauß-reduziert sind
i)	$(\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3')$	$(1,2)\,(1,3),(2,3)$
ii)	$(\mathbf{a}_1, \mathbf{a}_3', \mathbf{a}_2)$	$(1,2)\,(1,3)$
iii)	$(\mathbf{a}_3', \mathbf{a}_1, \mathbf{a}_2)$	$(2,3)$

Die Funktion **indsort** arbeitet dabei wie **FSort** mit dem Unterschied, daß zusätzlich ein Indexvektor $\boldsymbol{\kappa} = (\kappa_1, \kappa_2, \kappa_3)$ nach Abarbeitung von **indsort** indiziert, wie sich die Anordnung der Spalten verändert hat. So bedeutet etwa $\boldsymbol{\kappa} = (1, 3, 2)$ am Ausgang von **indsort**, daß die Spalten 3 und 2 vertauscht wurden.

In der Situation i) terminiert der Algorithmus. In der Situation ii) wird ab Anweisung (11) wiederholt. In Situation iii) wird ab **FGauß**$_1$ wiederholt, falls die Paar-Reduktionsbedingung noch nicht erzielt wurde.

Die Funktion **SFGauß**$_3$ arbeitet wie eine **FGauß**$_3$ mit dem Unterschied, daß der **SFGauß**$_3$ nach höchstens zwei **FGauß**$_3$-Schritten terminiert.

Das Struktogramm wird nun gegeben.

Algorithmus 6.3 (optimum)

$z_1 = h_{17,1} - h_{25,1}$			(1)
$z_2 = h_{13,1} - h_{7,1}$			(2)
$g_1 = -\lfloor z_1 / z_2 \rceil$			(3)
$h_1 = h_{5,1} + g_1 \cdot h_{4,1}$			(4)
$l_3 = h_{3,1}, d_1^{\min} = d_2^{\min} = 0$			(5)
FOR $i = -1$ to $+1$			(6)
	$d_2 = g_1 + i$		(7)
	$d_1 = -\lfloor (h_1 + i \cdot h_{4,1}) / h_{1,1} \rceil$		(8)
	$l_{\text{test}} = h_{3,1} + d_1^2 \cdot h_{1,1} + d_2^2 \cdot h_{2,1} + 2\,(d_1 \cdot d_2 \cdot h_{4,1} + d_1 \cdot h_{5,1} + d_2 \cdot h_{6,1})$		(9)
	IF $\min\{l_{\text{test}}, l_3\} < l_3$		(10)
	THEN	$(d_1^{\min}, d_2^{\min}) = (d_1, d_2)$	(11)
		$l_3 = l_{\text{test}}$	(12)
FGauß$_2$ – Schritt mit Kohärenzzahl $d_1^{\min}$			(13)
FGauß$_3$ – Schritt mit Kohärenzzahl $d_2^{\min}$			(14)

6.1 Komplexitätsresultat

Das Komplexitätsresultat wird in Satz 6.4 gegeben.

Satz 6.4 *Sei C_{bit} die Anzahl von Bitoperationen des FTPR-Algorithmus bei Eingabe einer zulässigen Matrix*

$$A = (\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3)$$

unter Verwendung klassischer Algorithmen für Division und Multiplikation. Es gilt:

$$C_{\text{bit}} = \Theta(\log(E(A))^2).$$

6.2 Korrektheit des FTPR-Algorithmus

Für den Beweis der Korrektheit des FTPR-Algorithmus verweisen wir auf den Korrektheitsbeweis des TPR-Algorithmus.

6.3 Analyse

Im diesem Abschnitt werden wir Satz 6.4 beweisen.

Die untere Schranke für die Bitkomplexität ist trivial, da in der Initialisierung, die aus den Anweisungen (1) bis (5) besteht, wenigsten $\lfloor \log(E(A))^2 \rfloor$ Bitoperationen benötigt werden.

Es folgt nun der nichttriviale Teil der Analyse.

Zunächst ist es nötig die Terminologie aus Kapitel 5 zu erweitern, da der FTPR-Algorithmus Gaußschritte und die Anweisung **optimum** ausführt. Letzters nennen wir einen **optimum**-Schritt.

Ist

$$B = (\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3)$$

eine zulässige Matrix und sind $(\mathbf{b}_1, \mathbf{b}_2)$ Gauß-reduziert, so nennen wir B eine *semi-reduzierte Matrix.* Es sei

$$((\delta_1^{(1)}, \delta_1^{(2)}), \ldots, (\delta_{\#OCP}^{(1)}, \delta_{\#OCP}^{(2)}))$$

die Folge aller Paare optimaler Kohärenzzahlen mit der Bezeichnung $\#OCP$ für die Anzahl der optimalen Kohärenzzahlen (Tupbezeichnung OCP).

Die Folge von Kohärenzzahlen, die in Anweisung (11) berechnet werden sei

$$(\rho_1, \ldots, \rho_{\#TC}).$$

Wir bezeichnen diese $\#TC$ Kohärenzzahlen als *transparente Kohärenzzahlen.*

Für die in den **FGauß**$_i$–Aufrufen $(1 \leq i \leq 3)$ auftretenden Kohärenzzahlen gelten die Bezeichnungen von Kapitel 5.

Eine Kohärenzzahl r eines Kohärenzzahlenabschnitts eines **FGauß**- oder **FGauß**-Aufrufs sei vom Typ

i) UC, wenn $|r| > 1$ und

ii) CC, wenn $|r| \leq 1$.

Mit $UC(i)$ und $CC(i)$ wird das Auftreten der Typen von Abschnitten in einem **FGauß**$_i$- oder SFGauß$_3$-Aufruf bezeichnet.

Es folgt ein Überblick über die Beweisschritte:

- Zuerst wird gezeigt, daß die Funktion **optimum** wenigstens so viel Reduktionserfolg erzielt wie ein Gaußschritt der **Gauß**$_3$-Funktion.
- Dann werden obere Schranken für die Produkte der unkritischen Kohärenzzahlen und der nichtverschwindenden optimierten Kohärenzzahlen gezeigt, die polynomiell in $||\mathbf{a}_3||$ sind.
- Unter Verwendung eines bekannten Komplexitätsresultats für die Operation der "Division zweier ganzer Zahlen mit Rest" folgt Satz 6.4

Satz 6.5 *Sei* $B = (\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3)$ *eine semi-reduzierte Matrix mit Gram-Schmidt-Koeffizienten* $\mu_{ij}\,(1 \le j < i \le 3)$. *Sei* $\mathbf{w}$ *einer der kürzesten von folgenden Vektoren*

$$\mathbf{b}_3 + d_1\,\mathbf{b}_1 + d_2\,\mathbf{b}_2,\ \textit{mit}\ d_2 = -\lfloor \mu_{32} \rceil + i,\, d_1 = -\lfloor \mu_{31} + d_2\,\mu_{32} \rceil,\, i \in [\![-1, 1]\!].$$

Dann gilt:

$$||\mathbf{w}|| \le \min\{||\mathbf{b}_3 + x_1\,\mathbf{b}_1 + x_2 \mathbf{b}_2|| : x_1, x_2 \in \mathbb{Z}\}.$$

Beweis: Es sei $B^\star$ die Gram-Schmidt-Matrix zu B. Dann ist:

$$\begin{aligned}&||\mathbf{b}_3 + x_1\,\mathbf{b}_1 + x_2\,\mathbf{b}_2||^2 = \\ &||\mathbf{b}_3^\star||^2 + ||\mathbf{b}_2^\star||^2\,(\mu_{32} + x_2)^2 + ||\mathbf{b}_1^\star||^2\,(\mu_{31} + x_2\,\mu_{21} + x_1)^2.\end{aligned}$$

Da $(\mathbf{b}_1, \mathbf{b}_2)$ Gauß-reduziert sind ist

$$||\mathbf{b}_2^\star||^2 \ge 3/4\,||\mathbf{b}_1^\star||^2.$$

Damit gilt für $\mathbf{w}$ und für alle $i \in \mathbb{Z}$ mit $|i + \lfloor \mu_{32} \rceil| > 1$ und für alle $x_1 \in \mathbb{Z}$:

$$\begin{aligned}||\mathbf{b}_3 + i\,\mathbf{b}_2 + x_1\,\mathbf{b}_1||^2 & \\ &\ge ||\mathbf{b}_3^\star||^2 + (i + \mu_{32})^2\,||\mathbf{b}_2^\star||^2 \\ &\ge ||\mathbf{b}_3^\star||^2 + (|i + \lfloor \mu_{32} \rceil| - 1/2)^2\,||\mathbf{b}_2^\star||^2 \\ &\ge ||\mathbf{b}_3^\star||^2 + (9/4)\,||\mathbf{b}_2^\star||^2 \\ &\ge ||\mathbf{b}_3^\star||^2 + 1/4||\mathbf{b}_2^\star||^2 + 3/2\,||\mathbf{b}_1^\star||^2 \\ &> ||\mathbf{b}_3^\star||^2 + 1/4||\mathbf{b}_2^\star||^2 + 1/4\,||\mathbf{b}_1^\star||^2|| \ge ||\mathbf{w}||^2.\end{aligned}$$

□

Es folgt nun der für den Beweis von Satz 6.4 entscheidend wichtige Satz 6.6.

Satz 6.6 *Sei* $A = (\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3)$ *eine zulässige Eingabe des FTPR-Algorithmus. Mit den gegeben Bezeichnungen gilt:*

$$\prod_{i=1}^{3} \prod_{r\ \mathrm{Typ}\ UC(i)} |r| \le ||\mathbf{a}_3||^6\,(1 + ||\mathbf{a}_3||^{12}) \tag{6.1}$$

$$\sum_{i=1}^{3} \#UC(i) \le 2 + 18\,\log_2 ||\mathbf{a}_3|| \tag{6.2}$$

$$\sum_{i=1}^{3} \#CC(i) \le 2 + 6\,\log_3 ||\mathbf{a}_3|| \tag{6.3}$$

$$\prod_{(\delta_1,\delta_2)\mathrm{Typ}\ OCP} \max\{1, |\delta_1|\} \cdot \max\{1, |\delta_2|\} \le ||\mathbf{a}_3||^{28} \tag{6.4}$$

$$\#OCP \le 6\,\log_2 ||\mathbf{a}_3|| \tag{6.5}$$

Beweis von Satz 6.6 Der Beweis der Ungleichungen (6.1), (6.2), (6.3) und (6.5) wird in Analogie zu den in Kapitel 5 präsentierten Beweistechniken geführt. Zum Beweis von Ungleichung (6.4) benötigen wir Lemma 6.7. Nach dem Lemma 6.7 und Ungleichung (6.5) ergibt sich die Ungleichung (6.4) unmittelbar. Ungleichung (6.5) ist gültig, da ein **optimum**-Schritt die Hadamardschranke wenigstens halbiert.

Lemma 6.7 *Sei*

$$A = (\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3)$$

eine semi-reduzierte Matrix und gelte ferner $\langle \mathbf{a}_1, \mathbf{a}_3 \rangle / ||\mathbf{a}_1||^2 \leq 1/2$ *und seien*

$$\mu_{ij}\,(1 \leq j < i \leq 3)$$

die Gram-Schmidt Koeffizienten der Matrix A *und*

$$A^\star = (\mathbf{a}_1^\star, \mathbf{a}_2^\star, \mathbf{a}_3^\star)$$

die Gram-Schmidt Matrix zur Matrix A*. Sei*

$$r = -\left\lfloor \frac{\langle \mathbf{a}_2, \mathbf{a}_3 \rangle}{||\mathbf{a}_2||^2} \right\rceil$$

und bezeichne

$$p_{ij} = \langle \mathbf{a}_i, \mathbf{a}_j \rangle\,(1 \leq i \leq j \leq 3).$$

Es gilt:

$$\mu_{32} = \frac{p_{11} \cdot p_{23} - p_{12} \cdot p_{13}}{p_{11} \cdot p_{22} - p_{12}^2} \tag{6.6}$$

und

$$|\mu_{32}| \leq \frac{4}{3}\,|r| + 1 \tag{6.7}$$

Beweis: Zunächst gilt

$$1 \leq \frac{||\mathbf{a}_2||^2}{||\mathbf{a}_2^\star||^2} \leq \frac{4}{3},$$

da $(\mathbf{a}_1, \mathbf{a}_2)$ Gauß-reduziert sind, denn

$$||\mathbf{a}_2^\star||^2 = ||\mathbf{a}_2||^2 - \mu_{21}^2\,||\mathbf{a}_1||^2 \geq \frac{3}{4}\,||\mathbf{a}_2||^2.$$

Es ist

$$\frac{|p_{23}|}{p_{22}} \leq \frac{2\,|r| + 1}{2}$$

und damit

$$|\mu_{32}| = \frac{|\langle \mathbf{a}_3, \mathbf{a}_2^\star \rangle|}{||\mathbf{a}_2^\star||^2} \leq \frac{|p_{23}|}{||\mathbf{a}_2^\star||^2} + \frac{|p_{13}\,p_{12}|}{p_{11}\,||\mathbf{a}_2^\star||^2} \leq \frac{4}{3}\,\frac{|p_{23}|}{||\mathbf{a}_2||^2} + \frac{1}{4}\,\frac{p_{11}}{||\mathbf{a}_2^\star||^2} \leq \frac{8\,|r| + 4}{6} + \frac{1}{3} \leq \frac{4}{3}\,|r| + 1$$

□

Herleitung des Komplexitätsresultats Wir schätzen zunächst die Bitkosten für **FGauß**-Schritte und dann die Bitkosten für **optimum**-Schritte ab. Die Bitlänge aller Operanden im TPR-Algorithmus, die nicht Kohärenzzahlen sind, ist durch $O(\log ||\mathbf{a}_3||)$ beschränkt. Dies gilt aufgrund der Gültigkeit der Cauchy-Schwarzschen Ungleichung sowie der Ungleichung

$$|\langle \mathbf{v}, \mathbf{v} \rangle| \geq ||\mathbf{v}||_\infty$$

für $\mathbf{v} \in \mathbb{R}^d$ und der Tatsache, daß sich die euklidische Länge der Vektoren, die im TPR-Algorithmus reduziert werden, nur verringern kann.

Kosten für FGauß-Schritte Hier können zwei Kostenkategorien auftreten, die beide schlußendlich eine quadratische obere Schranke besitzen.

a) Die Operationen mit Kohärenzzahlen in Abschnitten vom Typ UC kosten nach Lemma 5.17 $O(\log ||\mathbf{a}_3||)$-Bitoperationen und treten nach Ungleichung (6.3) $O(\log ||\mathbf{a}_3||)$-fach auf, so daß sich per saldo Gesamtkosten von $O(\log^2 ||\mathbf{a}_3||)$ Bitoperationen ergeben.

b) In Algorithmus 6.2 wurde dargestellt, wie ein **FGauß**-Schritt auszuführen ist. Danach ist nun ersichtlich, daß maximal die vierte Potenz einer Kohärenzzahl berechnet werden muß. Wegen der Gültigkeit von (6.1) und (6.2) und Lemma 5.17 sind die Kosten für Operationen mit unkritischen Kohärenzzahlen beschränkt durch $O(\log^2 ||\mathbf{a}_3||)$.

   Erklärung: Für $c > 8$ ist

$$\left(\prod_{r \text{ vom Typ UC}} |r| \right)^4 \leq c \cdot ||\mathbf{a}_3||^{72},$$

   also folgt nach Lemma 5.17 für die Bitkomplexität aller Operationen mit Kohärenzzahlen, die in Kohärenzzahlenabschnitten vom UC-Typus auftreten:

$$\begin{aligned} 2 \log ||\mathbf{a}_3|| \cdot 4 \cdot \textstyle\sum_{r \text{ vom Typ UC}} (\log(|r|) + 1) &= 8 \log ||\mathbf{a}_3|| \cdot \log \left(\textstyle\prod_{r \text{ vom Typ UC}} 2\,|r| \right) \\ &\leq 8 \log ||\mathbf{a}_3|| \cdot 2 \cdot (1 + 24 \log ||\mathbf{a}_3||). \end{aligned}$$

Kosten für optimum-Schritte Die Berechnung von Zähler und Nenner eines optimalen Kohärenzzahlenpaares kostet $O(\log ||\mathbf{a}_3||)$-Bitoperationen, da diese aus der Subtraktion zweier ganzer Zahlen der Länge $O(\log ||\mathbf{a}_3||)$ berechnet werden. Nach Ungleichung (6.5) werden insgesamt $O(\log ||\mathbf{a}_3||)$ optimale Kohärenzzahlenpaare (δ_1, δ_2) berechnet. Mit einem optimalen Kohärenzzahlenpaar (δ_1, δ_2) wird jeweils ein **FGauß**$_2$-Schritt, $(\mathbf{b}_3 = \mathbf{b}_3 + \delta_1\, \mathbf{b}_1)$, und ein **FGauß**$_3$-Schritt, $(\mathbf{b}_3 = \mathbf{b}_3 + \delta_2\, \mathbf{b}_2)$, ausgeführt. Alle Operanden, die keine Kohärenzzahlen sind, etwa die h_{ij}, haben eine durch $O(\log ||\mathbf{a}_3||)$ begrenzte Bitlänge. Nach den Ungleichungen (6.4) und (6.5) ergibt sich

$$\begin{aligned} \underbrace{2 \log ||\mathbf{a}_3||}_{\text{Schranke für Bitlänge der } (h_{ij})} &\cdot \underbrace{\sum_{\text{alle optimalen } (\delta_1,\, \delta_2)} \log(\max\{2, 2\,|\delta_1|\} \max\{2, 2\,|\delta_2|\})}_{\text{Summation der Bitschranken für optimale Kohärenzzahlen}} &= \\ 2 \log ||\mathbf{a}_3|| &\cdot \log \textstyle\prod_{\text{alle optimalen } (\delta_1,\, \delta_2)} 2 \log(\max\{1, |\delta_1|\} \max\{1, |\delta_2|\}) &\leq \\ 2 \log ||\mathbf{a}_3|| &\cdot (\#OCP + 28 \log ||\mathbf{a}_3||) \leq 68 \log^2 ||\mathbf{a}_3||. \end{aligned}$$

Kapitel 7

Sind naive Paar-Reduktionsalgorithmen sinnvoll?

Bei der Entwicklung neuer Reduktionsalgorithmen für Formen und Gitter wurde von Pohst, Zassenhaus [26] et. al. vorgeschlagen, erst alle Paare, dann alle Tripel etcetera zu reduzieren. In diesem Kapitel wird nachgewiesen, daß dieses Konzept **nicht** zu guten Gitterbasisreduktionsalgorithmen führt, da es in seinem Kern einen naiven Paar-Reduktionsalgorithmus beinhaltet. Darunter verstehen wir eine Verallgemeinerung des TPR-Algorithmus auf beliebig Dimension, der erst alle Paare Gauß-reduziert. Der erste Negativaspekt des Verfahrens wird in Abschnitt 7.1 untersucht. Es wird gezeigt, daß bei einer Basis eines Gitters, deren sämtliche Paare Gauß-reduziert sind, keine Aussage über die Approximationsgüte gemacht werden kann. Dies ist noch kein schwerwiegender Mangel, da die vorgeschlagenen Algorithmen ja noch nicht terminieren, sondern nun alle Tripel etcetera reduzieren würden. Der zweite und gravierendere Negativaspekt wird in Abschnitt 7.2 dargestellt. Es wird ein Beispiel für eine Basis eines Gitters gegeben, bei der wir vermuten, daß jeder Paar-Reduktionsalgorithmus ineffektiv ist.

7.1 Fehlende Approximationsgüte

In diesem Abschnitt wird ein Beispiel einer Basis eines Gitters von beliebigem Rang n angegeben, deren sämtliche Paare Gauß-reduziert sind, deren kürzester Vektor jedoch beliebig weit vom ersten sukzessiven Minimum des Gitters entfernt ist.

Definition 7.1 (Paareduziertheit) *Sei $n \geq 2$ eine natürliche Zahl. Eine reguläre Matrix $(\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_n) \in \mathbb{R}^{n \times n}$ heißt* Paar-reduziert, *wenn die Bedingungen 7.1 und 7.2 erfüllt sind.*

$$\| \mathbf{a}_i \|^2 \leq \| \mathbf{a}_{i+1} \|^2 \text{ für } 1 \leq i < n \tag{7.1}$$

$$|\langle \mathbf{a}_i, \mathbf{a}_j \rangle| \leq 2 \| \mathbf{a}_i \|^2 \text{ für } 1 \leq i < j \leq n \tag{7.2}$$

Es wird nun gezeigt, daß für eine Paar-reduzierte Matrix

$$A = (\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_n) \in \mathrm{GL}(n, \mathbb{R})$$

keine obere Schranke der Form

$$\| \mathbf{a}_1 \|^2 \leq c_n \, \lambda_1^2$$

existieren kann, wenn λ_1 das erste sukzessive Minimum des Gitters $L(A)$ und c_n eine positive reelle Zahl bezeichnet. Für eine LLL(y)-reduzierte Matrix kann eine derartige Schranke gemäß Ungleichung (1.7) gegeben werden.

Wir konstruieren nach einer Anregung von Hendrik W. Lenstra [17] Matrizen A, die Paar-reduziert sind und deren kürzester Spaltenvektor eine beliebig schlechte Approximation des ersten sukzessiven Minimums des Gitters L(A) ist.

Satz 7.2 *Für jede natürliche Zahl $n > 2$ und $t > 1$ existieren Gitter $\Lambda_t \subseteq \mathbb{Z}^n$ von vollem Rang n, mit Paar-reduzierten Basen*

$$A_t = (\mathbf{a}_1(t), \mathbf{a}_2(t), \ldots, \mathbf{a}_n(t)) \in \mathbb{Z}^{n \times n},$$

so daß, wenn $\lambda_1(t)$ das erste sukzessive Minimum des Gitters Λ_t bezeichnet,

$$||\mathbf{a}_1(t)|| > \frac{t}{\sqrt{n^3}} \lambda_1(t) \tag{7.3}$$

gilt.

Beweis: Es werden nun Matrizen mit den gewünschten Eigenschaften angegeben. Dazu bezeichne $\mathbf{e}_i$, für $1 \leq i \leq n$ die kanonische Basis des $\mathbb{R}^n$, also $\mathbf{e}_i = (\delta_{1i}, \delta_{2i}, \ldots, \delta_{ni})$ mit dem Kroneckersymbol

$$\delta_{ij} = \begin{cases} 1 \text{ für } i = j \\ 0 \text{ sonst.} \end{cases}$$

Sei $\mathbf{E} = \sum_{i=1}^{n} \mathbf{e}_i = (1, 1, \ldots, 1)$ die Summe der kanonischen Basisvektoren und mit

$$H = \{\mathbf{x} \in \mathbb{R}^n : \langle \mathbf{x}, \mathbf{E} \rangle = 0\}$$

die Hyperebene mit Normalenvektor $\mathbf{E}/\sqrt{n}$. Wir projizieren nun die um den Faktor n gestreckte Standardbasis $(\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_n)$ orthogonal auf die Hyperebene H und erhalten n linear abhängige Vektoren

$$\mathbf{c}_i = n \left(\mathbf{e}_i - \frac{< \mathbf{e}_i, \mathbf{E} >}{\| \mathbf{E} \|^2} \mathbf{E} \right) = n \cdot \mathbf{e}_i - \mathbf{E} \; (1 \leq i \leq n)$$

in H. Streckung dieser Vektoren um den Faktor t und Addition des Vektors $\mathbf{E}$ führt zu den n Vektoren $\mathbf{b}_i = t\,\mathbf{c}_i + \mathbf{E}$ $(1 \leq i \leq n)$, deren Eigenschaften die Kernaussage des folgenden Lemmas bilden:

Lemma 7.3 *Die Vektoren*

$$\mathbf{b}_i = t\,\mathbf{c}_i + \mathbf{E} \quad (1 \leq i \leq n)$$

haben ganzzahlige Komponenten und sind linear unabhängig. Das von ihnen erzeugte Gitter $\Lambda_t = \sum_{i=1}^{n} \mathbb{Z} b_i$ hat also den vollen Rang n.

Beweis: Daß die Vektoren ganzzahlige Komponenten besitzen, ergibt sich daraus, daß sowohl $\mathbf{c}_i$ als auch $t \cdot \mathbf{c}_i$ $(1 \leq i \leq n)$ ein Vektor mit ganzzahligen Komponenten ist. Da $\mathbf{E}$ nur ganzzahlige Komponenten aufweist, ist auch die Summe $t \cdot \mathbf{c}_i + \mathbf{E}$ ein Vektor, der nur ganzzahlige Einträge hat.

Die lineare Unabhängigkeit wird bewiesen, indem wir zuerst zeigen, daß die Gleichung $\sum_{i=1}^n \mathbf{b}_i y_i = 0$ zur Folge hat, daß $\sum_{i=0}^n y_i = 0$ ist.

Die Bildung des Skalarprodukts von $\sum_{i=1}^n \mathbf{b}_i y_i$ mit $\mathbf{E}$ ergibt

$$\sum_{i=1}^n \mathbf{b}_i \mathbf{E} y_i = (tc_i + \mathbf{E})\mathbf{E} y_i = \| \mathbf{E} \|^2 \sum_{i=1} y_i = 0.$$

Dabei verschwinden die Terme $t\, \mathbf{c}_i\, \mathbf{E}\, y_i$ für $1 \leq i \leq n$, da $\mathbf{c}_i\, \mathbf{E} = 0$ wegen $\mathbf{c}_i \in H$ gilt. Da $\| \mathbf{E} \|^2 = n > 2$ ist, folgt $\sum_{i=1}^n y_i = 0$ und damit gilt die behauptete Implikation.

Nun wird die lineare Unabhängigkeit der Vektoren $(\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n)$ durch einen indirekten Beweis gezeigt. Wir beziehen die Kontraposition und nehmen an, der Ausdruck $\sum_{i=1}^n y_i \mathbf{b}_i = 0$ ließe sich sich auf nichttriviale Weise zu Null kombinieren. Dann muß aber ein Index $k \in \{1, 2, \ldots, n\}$ mit $y_k \neq 0$ existieren. Wir bilden nun das euklidische Skalarprodukt der besagten Vektorsumme mit dem Vektor $\mathbf{c}_k$. Dies führt zu:

$$\begin{aligned}
0 &= \langle \textstyle\sum_{i=1}^n y_i\, \mathbf{b}_i, \mathbf{c}_k \rangle = \sum_{i=1}^n y_i \langle \mathbf{b}_i, \mathbf{c}_k \rangle & &= \textstyle\sum_{i=1}^n y_i \langle (t\, \mathbf{c}_i + \mathbf{E}), \mathbf{c}_k \rangle \\
&= \textstyle\sum_{i=1}^n y_i\, t \langle \mathbf{c}_i, \mathbf{c}_k \rangle & &= y_k\, t \| \mathbf{c}_k \|^2 + \textstyle\sum_{\substack{i=1 \\ i \neq k}}^n y_i\, t \langle \mathbf{c}_i, \mathbf{c}_k \rangle \\
&= y_k\, t \| \mathbf{c}_k \|^2 + \textstyle\sum_{\substack{i=1 \\ i \neq k}}^n y_i\, t\, (n\, \mathbf{e}_i - \mathbf{E})(n \mathbf{e}_k - \mathbf{E}) & & \\
&= y_k\, t \| \mathbf{c}_k \|^2 + \textstyle\sum_{\substack{i=1 \\ i \neq k}}^n y_i\, t\, (-n - n + \| \mathbf{E} \|^2) & &= y_k\, t \| \mathbf{c}_k \|^2 + \textstyle\sum_{\substack{i=1 \\ i \neq k}}^n y_i\, t\, (-n) \\
&= y_k\, (t \| \mathbf{c}_k \|^2 + t\, n) = 0
\end{aligned}$$

Die vorletzte Identität ergibt sich, da wie oben gezeigt, $\sum_i y_i\, \mathbf{b}_i = 0$ zur Folge hat, daß $\sum_i y_i = 0$ gilt.

Mit $t(||c_k|| + n) > 0$ ist $y_k = 0$, was im Widerspruch zur Annahme $y_k \neq 0$ steht. Damit haben wir die Kontraposition ad absurdum geführt und die lineare Unabhängigkeit bewiesen, was den Beweis von Lemma 7.3 vervollständigt. □

Es folgt nun der Bweis von Ungleichung (7.3) aus Satz 7.2. Es ist

$$\sum_{i=1}^n \mathbf{c}_i = n \cdot \sum_{i=1}^n \mathbf{e}_i - \sum_{i=1}^n \mathbf{E} = 0.$$

Definieren wir $\mathbf{x} = \sum_{i=1}^n \mathbf{b}_i$, so gilt

$$\mathbf{x} = \sum_{i=1}^n t \mathbf{c}_i + \sum_{i=1}^n \mathbf{E} = n \cdot \mathbf{E}.$$

Damit ist nun

$$\frac{\| \mathbf{b}_1 \|^2}{||\mathbf{x}||^2} = \frac{\| t\, \mathbf{c}_1 + \mathbf{E} \|^2}{n^3} = \frac{t^2 \| \mathbf{c}_1 \|^2 + n}{n^3} > \frac{t^2}{n^3}.$$

Dabei gilt die vorletzte Gleichung wegen $\mathbf{c}_1 \in H$ und die letzte Ungleichung, da $\mathbf{c}_1$ ein nichtverschwindender Vektor mit ganzzahligen Komponenten ist, der also wenigstens die euklische Länge eins besitzt. Damit ist sogar noch eine stärkere Ungleichung als die Ungleichung (7.3) bewiesen worden, denn $\mathbf{x} \in \Lambda_t - \{0\}$ und $\| \mathbf{x} \|^2 \geq \lambda_1$.

Es bleibt noch zu zeigen, daß die konstruierte Basis tatsächlich die Eigenschaft erfüllt, Paar-reduziert zu sein. Dies zeigen wir in untenstehendem Lemma 7.4.

Lemma 7.4 *Mit den bereits eingeführten Bezeichnungen gilt für* $1 \leq i < j \leq n$*:*

$$\begin{aligned} &i) \quad ||\mathbf{b}_i|| = ||\mathbf{b}_j|| \\ &ii) \quad |\langle \mathbf{b}_i, \mathbf{b}_j \rangle| \cdot \| \mathbf{b}_i \|^{-2} \leq 1/2 \end{aligned}$$

Beweis: Eine kurze Rechnung ergibt

$$\| \mathbf{b}_i \|^2 = n(t^2\, n - t^2 + 1),$$

womit i) gezeigt ist. Die zweite Ungleichung folgt, da

$$\langle \mathbf{b}_i, \mathbf{b}_j \rangle = -n(t^2 - 1)$$

ist. Also ist ii) äquivalent zu $2(t^2 - 1) \leq t^2\, n - t^2 + 1$. Dies kann zur Ungleichung

$$3t^2 \leq t^2\, n + 3$$

umgeformt werden. □

Damit ist der Satz 7.2 bewiesen. □

Fazit Für $n \in \mathbb{Z}^{>2}$ ergibt die orthogonale Projektion der Spalten der $n \times n$ Einheitsmatrix eine singuläre Matrix B. Wird zu jedem Eintrag der Matrix B eine kleine positive reelle und frei wählbare Größe ϵ addiert, so ist die so gebildete Matrix regulär und Paar-reduziert. Das von den Spalten dieser regulären Matrix gebildete Gitter besitzt einen Vektor der Länge $\sqrt{n}\,\epsilon$. Da die Länge des ersten Vektors dieser regulären Matrix unabhängig von ϵ ist, gibt es keine obere Schranke für die Länge dieses Vektors in Abhängigkeit von n.

7.2 Exponentielle Laufzeit möglich?

In diesem Abschnitt wird ein Beispiel einer Matrix angegeben für die unklar ist, ob die Anzahl der Gaußschritte von Paar-Reduktionsalgorithmen überhaupt polynomiell in der Dimension n der Matrix und der Anzahl der Bits, die die Eingabe kodieren, ist.

Beispiel 7.5

$$M(n,k) = \begin{pmatrix} -1 & 0 & \cdots & 0 & 0 & 2^k \\ 0 & -1 & \cdots & 0 & \vdots & 2^k \\ 0 & 0 & -1 & 0 & \vdots & \vdots \\ \vdots & \vdots & 0 & \ddots & 0 & \vdots \\ 1 & 1 & \cdots & 1 & -1 & 2^k \\ 0 & 0 & \cdots & 0 & 1 & 0 \end{pmatrix}$$

Probleme

1) Gibt es einen in $E(M(n,k))$ und n polynomiellen Paar-Reduktionsalgorithmus?

2) Ist Algorithmus 7.6 polynomiell in $E(M(n,k))$ und n?

Wir geben nun das Struktogramm von Algorithmus 7.6 als ein Beispiel eines Paar-reduktionsalgorithmus an, für den unklar ist, ob die Anzahl der Iterationen polynomiell in der Eingabelänge und Dimension beschränkt ist.

Algorithmus 7.6 (PR(A))

Eingabe: Eine zulässige Matrix A, die Basis eines Gitters L vom Rang n ist
Ausgabe: Eine Paar-reduzierte Basis des Gitters L

<table>
<tr><td colspan="5">$i = 1$</td><td>(1)</td></tr>
<tr><td colspan="5">$j = 2$</td><td>(2)</td></tr>
<tr><td colspan="5">$f = 0$</td><td>(3)</td></tr>
<tr><td rowspan="18"></td><td colspan="4">$r = \lfloor \langle \mathbf{a}_i, \mathbf{a}_j \rangle / \langle \mathbf{a}_i, \mathbf{a}_i \rangle \rceil$</td><td>(4)</td></tr>
<tr><td colspan="4">$\mathbf{a}_j = \mathbf{a}_j - r \cdot \mathbf{a}_i$</td><td>(5)</td></tr>
<tr><td colspan="4">IF $r \neq 0$</td><td>(6)</td></tr>
<tr><td>THEN</td><td colspan="3">$f = 1$</td><td>(7)</td></tr>
<tr><td colspan="4">$j_1 = j$</td><td>(8)</td></tr>
<tr><td colspan="4">$j = \tau(j)$</td><td>(9)</td></tr>
<tr><td colspan="4">IF $j = j_1$</td><td>(10)</td></tr>
<tr><td rowspan="8">THEN</td><td colspan="3">IF $i = j - 1$</td><td>(11)</td></tr>
<tr><td rowspan="6">THEN</td><td colspan="2">IF $f = 0$</td><td>(12)</td></tr>
<tr><td rowspan="3">THEN</td><td>$j = j + 1$</td><td>(13)</td></tr>
<tr><td>$i = 1$</td><td>(14)</td></tr>
<tr><td>$f = 0$</td><td>(15)</td></tr>
<tr><td rowspan="2">ELSE</td><td>$i = 1$</td><td>(16)</td></tr>
<tr><td>$f = 0$</td><td>(17)</td></tr>
<tr><td>ELSE</td><td colspan="2">$i = i + 1$</td><td>(18)</td></tr>
<tr><td rowspan="3">ELSE</td><td colspan="3">$j = j + 1$</td><td>(19)</td></tr>
<tr><td colspan="3">$i = 1$</td><td>(20)</td></tr>
<tr><td colspan="3">$f = 0$</td><td>(21)</td></tr>
<tr><td colspan="5">UNTIL $j = n + 1$</td><td>(22)</td></tr>
</table>

Erklärung des PR-Algorithmus Der PR-Algorithmus erhält als Eingabe eine *zulässige Matrix*, also eine reguläre ganzzahlige Matrix, deren Spalten nach euklidischen Längen sortiert sind. In Anweisung (5) wird ein Sortieralgorithmus aufgerufen, der eine Permutation $\tau \in S(n)$ berechnet, so daß $(\mathbf{a}_{\tau(1)}, \ldots, \mathbf{a}_{\tau(n)})$ nach euklischen Längen geordnet ist. Der Sortieralgorithmus ist so geartet, daß nur im Falle einer echten "größer/kleiner-" Relation getauscht wird. Der Grund dafür besteht darin "deadlocks" auszuschließen. Der Algorithmus 7.6 arbeitet wie folgt: Vor Anweisung (4) sind die Paare

$$(\mathbf{a}_{i'}, \mathbf{a}_{j'}) \quad (1 \leq i' < j' \leq j-1)$$

Gauß-reduziert. Also zu Beginn ($j = 2$) ist im allgemeinen noch kein einziges Paar Gauß-reduziert. Für $j = 3$ ist das Paar $(\mathbf{b}_1, \mathbf{b}_2)$ Gauß-reduziert etc.

Für $j = n + 1$ wird die Schleife verlassen und es sind alle Paare Gauß-reduziert.

Bemerkungen Für die gerade beschriebene Paar-Reduktionsstrategie des Algorithmus 7.6 läßt sich die Rekursion

$$T(n,k) \geq T(n-1,k) + T(n,k/3) \text{ und } T(n,l) = 0 \text{ falls } l < 1,$$

für die Anzahl der Gaußschritte (Anweisung (5)) ableiten. Nach dem Verkürzungslemma aus Kapitel 5 ist sicherlich

$$T(n,k) \leq E(M(n,k)).$$

Es zeigt sich, daß $T(n,k)$ für festes n polynomiell in k und für festes k polynomiell in n ist. Als untere Schranke für $T(n,k)$ wird die Funktion:

$$Ln(2\,n\,k, (2\sqrt{k\,n}/(k+n)))$$

vermutet, wobei

$$Ln(\gamma, \delta) = exp((1-\delta)\log\gamma + \delta\,\gamma)$$

ist.

Die Wahl der Argumente der Ln-Funktion, die aus der Analyse der Komplexität von Faktorisierungsalgorithmen für ganze Zahlen bekannt ist, ist so, daß die Funktion polynomiell für k oder n fest und sonst exponentiell ist. Dies kann wie folgt kurz begründet werden. Offenbar ist

$$\begin{array}{l} Ln(\gamma, 0) = \gamma \\ \text{und} \\ Ln(\gamma, 1) = \exp(\gamma). \end{array}$$

Ferner ist

$$0 \leq \frac{2\sqrt{k\,n}}{k+n} \leq 1 \text{ für } k, n > 0. \tag{7.4}$$

Letzteres ergibt sich mit $\overline{k} = |\sqrt{k}|$ und $\overline{n} = |\sqrt{n}|$, da $(\overline{k} - \overline{n})^2 = k + n - 2\sqrt{k\,n} \geq 0$ gilt. Für n fest und $k \gg 1$ variabel (oder umgekehrt), ist der Quotient aus 7.4 sehr nahe bei Null.

Die Matrix $M(n,k)$ ist so konzipiert, daß jede unimodulare Operation die Summe der Einträge der $n-$ten Spalte konstant läßt. Jeder Paar-Reduktionsalgorithmus terminiert erst, wenn alle Differenzen zwischen Einträgen der letzten Spalte kleiner oder gleich eins sind.

Auch wenn es in $E(M(n,k))$ und n polynomielle Schranke für die Anzahl der Iteration von Algorithmus 7.6 gibt, so macht das Beispiel doch sehr drastisch deutlich, daß vom naiven Paar-Reduktionskonzept abzuraten ist. Einige "experimentelle" Daten mögen diese Aussage bekräftigen.

Der LLL-Algorithmus benötigt bei Eingabe der Matrix $M(n,k)$ nur $n\,(n-1)/2$ Gaußschritte.

Eingabe: $M(n,k)$		
n	k	$T(n,k)$
100	2	5148
	4	5643
	8	14454
	16	131472
200	2	20298
	4	21293
	8	43979
	16	753414
300	2	45448
	4	46943
	8	81029
	16	1987154
400	2	80598
	4	82593
	8	128079
	16	3796884
500	2	125748
	4	128243
	8	185129
	16	6082810
600	2	180898
	4	183893
	8	252179
	16	8709460
700	2	246048
	4	249543
	8	329229
	16	11579634
800	2	321198
	4	325193
	8	416279
	16	14544197

Tabelle 7.1: Anzahl der Iterationen eines “naiven” Paar-Reduktionsalgorithmus

Literaturverzeichnis

[1] L. Babai
On Lovász lattice reduction and the nearest lattice point problem, Combinatorica **5**, 1985.

[2] J. Buchmann, O. van Sprang
On short representations of orders and algebraic number fields, in Revision.

[3] W. Blum
Richtig aber häßlich: Beweise vom Computer, Die Zeit vom 12. Februar 1993.

[4] J. W. S Cassels
Rational Quadratic Forms, Academic Press, 1978.

[5] Henri Cohen
A course in computational algebraic number theory, Springer Verlag, 1994.

[6] M. J. Coster, A. Joux, B. A. LaMachia, A. M. Odlyzko, C. P. Schnorr und J. Stern
Improved low-density subset sum algorithms, computational complexity **2**, 111-128, 1992.

[7] J. D. Dixon
The number of steps in the Euclidean algorithm, Journal of Number Theory **2**, 414-422, 1970.

[8] A. Dupré
Sur les nombre des divisions à effectuer pour obtenir le plus grand commun diviseur entre deux nombres entiers, Journal de Mathémathiques **11**, 41-71, 1846.

[9] C.F. Gauss
Disquisitiones Arithmeticae, Leipzig 1801, German translation: Untersuchungen über die höhere Arithmetik, Springer Verlag Berlin 1889, (reprint: Chelsea, New York, 1981).

[10] M. Grötschel, L. Lovász, A. Schrijver
Geometric Algorithms and Combinatorial Optimization, Springer Verlag 1988.

[11] B. Helfrich
Algorithms to construct minkowski reduced and hermite reduced lattice bases, Theoretical Computer Science **41**, 125-139, 1985.

[12] C. Hermite
Extraits des lettres de M. Ch. Hermite à M. Jacobi sur différents objets de la théorie des nombres, Deuxième lettre, Crelles Journal für reine und angewandte Mathematik **40**, 279 -290, 1850.

[13] D. E. Knuth
The art of computer progamming, Vol. II, Addison-Wesley, 1981.

[14] D. E. Knuth
The art of computer programming (Sorting and Searching), Vol. III, Addison-Wesley, 1971.

[15] A. Korkin und G. Zolotarev
Sur les formes quadratiques, Mathematische Annalen **6**, 366-389, 1873.

[16] A.K. Lenstra, H.W. Lenstra, Jr. und L. Lovász
Factoring polynomials with rational coefficients, Mathematische Annalen 261, 515-534, 1982.

[17] H. W. Lenstra
persönliche Anregung per e-mail, im Februar 1993.

[18] H. W. Lenstra
Integer Programming with a fixed number of variables, Mathematics of Operations Research **9**, 538 - 548, 1983.

[19] J. C. Lagarias, H. W. Lenstra, Jr. und C. P. Schnorr
Korkin-Zolotarev Bases and Successive Minima of a Lattice and its Reciprocal Lattice, Combinatorica **10** (4), 333-348, 1990.

[20] J. C. Lagarias
Worst-Case complexity bounds for algorithms in the theory of integral quadratic forms, Journal of Algorithms **1**, 142-186, 1980.

[21] J. C. Lagarias und A. Odlyzko
Solving low-density subset sum problems, 24th IEEE Symposium FOCS, 1982).

[22] J. L. Lagrange
Recherche d'arithmétique, Nouv. Mém. Acad. Berlin, Œuvres, vol. VIII, 265-312 und 693-753, 1773.

[23] H. Minkowski
Über die positiven quadratischen Formen und über kettenbruchähnliche Algorithmen, Crelles Journal für reine und angewandte Mathematik **107**, 278-297, 1891.

[24] I. Nassi, B. Shneiderman
Flowchart techniques for structured programming, SIGPLAN Notices **8**, No. 8, 12-26, 1973.

[25] A. M. Odlyzko und H. I. J. te Riele, "Disproof of the Mertens conjecture", Journal für die reine und angewandte Mathematik **357**, 138-160, 1985.

[26] M. Pohst
On the computation of lattice vectors of minimal length, successive minima and reduced bases with applications, ACM Sigsam Bulletin, vol. 15, 37-44, 1981.

[27] M. Pohst, W. Plesken
Constructing Integral Lattices With Prescribed Minimum, Mathematics of Computation, Vol. **45**, Number 171, 209-221, 1985.

[28] R. Kannan
Minkowski's Convex Body Theorem and Integer Programming, Mathematics of Operations Research, Vol. **12**, No. 5, 1987.

[29] C. P. Schnorr
A hierarchy of polynomial time lattice basis reduction algorithms, Theoretical Computer Science 53, 201 - 224, 1987.

[30] A. Schönhage
Fast reduction and composition of binary quadratic forms, preprint 1992.

[31] A. Schönhage, V. Strassen
Schnelle Multiplikation großer Zahlen, Computing **7**, 281-292, 1971.

[32] O. van Sprang
Lower bounds for the LLL lattice reduction algorithm, preprint 1994.

[33] O. van Sprang
A fast algorithm for computing Minkowski-reduced rank 3 lattice bases, in Vorbereitung für das Journal of algorithms.

[34] A. Shamir
A polynomial time algorithm for breaking the Merkle - Hellman cryptosystem, 23rd IEEE Symposium FOCS, 1982.

[35] B. Vallée
Algorithmique en géometrie des nombres. Applications á la cryptographie et á la factorisation des entieres, Dossier en vu de l'obtention de l'habilitation á diriger des recherches, soutenue le 18 dècembre 1989.

[36] B. Vallée
Un approche géometrique de la réduction des réseaux en petite dimension, Thèse de doctorat de l'Université de Caen, 1986.

[37] B. Vallée
Gauß' algorithm revisited, Journal of Algorithms, 123 - 131, 1991.

[38] B. Vallée
An affine point of view on minima finding in integer lattices of lower dimensions, Département de Mathématiques, Université de CAEN, France.

[39] B. L. van der Waerden, H. Gross
Studien zur Theorie der quadratischen Formen, Birkhäuser, Basel, 1968.

[40] P. van Emde Boas
Another NP-complete problem and the complexity of computing short vectors in a lattice, Technical Report, University of Amsterdam, No. 81-04, 1981.

[41] C. K. Yap
Fast unimodular reduction: Planar integer lattices, 33rd Annual Symposium on Foundations of Computer Science (FOCS), 437 - 446, 1992.

Index