Algorithmische Konstruktionen von Gittern

Boris Hemkemeier

Dissertation zur Erlangung des Grades eines
Doktors der Naturwissenschaften

Dem Fachbereich Mathematik der Universität Dortmund
vorgelegt im September 2003

I have a great suspicion that for example Euler today would spend much more of his time on writing software because he spent so much of his time e.g., in efforts of calculating tables of moon positions. And I believe that Gauß as well would spend much more time sitting in front of the screen.

Yuri I. Manin [AS98]

Inhaltsverzeichnis

1 Einleitung **1**

2 Klassifikation ganzzahliger Gitter **5**
2.1 Einleitung 5
2.2 Die Nachbarmethode 5
2.3 Die Konstruktion des Nachbarschaftsgraphen 7
2.4 Konstruktion einer Basis eines Nachbarn 10
2.4.1 Ungerade Gitter und der duale Nachbarschaftsgraph 11
2.5 Das Programm `tn` 12
2.5.1 Die Berechnung der Automorphismengruppe eines Gitters . . 12
2.5.2 Rechnen in $L/2L$ 13
2.5.3 Die diskrete Fourier–Transformierte der Längenfunktion . . . 17
2.6 2-Nachbarn bei gerader Determinante 20
2.7 Ergebnisse 21
2.7.1 ℓ-modulare Gitter 21
2.7.2 Das Geschlecht des Barnes-Wall-Gitters Λ_{16} 23
2.7.3 Neuere Ergebnisse anderer Autoren 24
2.8 Verifikation der Ergebnisse 26

3 Über die Zerlegung von Gittern **28**
3.1 Überblick 28
3.1.1 Das Zerlegungsproblem 28
3.1.2 Minimale Erzeugendensysteme 29
3.1.3 Der MLLL bei großen Erzeugendensystemen 30
3.2 Unzerlegbare Vektoren 30
3.3 Knesers Methode 33
3.4 Der inkrementelle Algorithmus 34
3.4.1 Datenstrukturen und Laufzeitanalyse 36
3.5 Gitterbasen aus großen Erzeugendensystemen 41
3.5.1 Der BP-LLL 41
3.5.2 Der MLLL 42

3.5.3 Der serielle BP-LLL 42
3.5.4 Der serielle BP-LLL mit Aussieben 42
3.6 Beispiele und Beobachtungen 43
3.7 Implementierungshinweise 45
3.7.1 Das Zerlegungsproblem 45
3.7.2 Die Basisberechnung aus einem großen Erzeugendensystem . 46

4 Der Quantizer der Voronoïzelle eines Gitters 47
4.1 Die Voronoïzelle eines Gitters 48
4.2 Die Voronoïzelle als Polytop 49
4.3 Voronoïrelevante Vektoren 50
4.4 Der Quantizer eines Gitters 52
4.5 Gitter mit kleinem Quantizer 53

1 Einleitung

Motivation dieser Arbeit ist ein Klassifikationsprojekt aus dem Gebiet der ganzzahligen Gitter. Martin Kneser beschrieb 1957 in dem Artikel *Klassenzahlen definiter quadratischer Formen* [Kne57] ein Exhaustionsverfahren für die Klassifikation vollständiger Geschlechter von Gittern. Diese Methode ist nicht unmittelbar praktikabel für ein automatisiertes Verfahren; sie läßt sich aber unter algorithmischen Aspekten modifizieren, um auch Geschlechter mit großer Klassenzahl klassifizieren zu können.

Im Rahmen dieser Arbeit wurde daraus das Computerprogramm `tn` entwickelt, um die Geschlechter vieler Gitter zu bestimmen, die bis auf wenige Ausnahmen zuvor nicht vollständig bekannt waren. Bei der Konstruktion neuer Gitter mit Knesers Methode behält man die Kontrolle über wichtige Gitterinvarianten. Damit unterscheidet sich `tn` von lokalen Optimierungs- oder Trainingsalgorithmen und hat eine gezielte Suche nach Gittern mit herausragenden Eigenschaften wie Extremalität, Wurzelsystem oder hoher Packungsdichte erlaubt.

Die Algorithmik der Konstruktion ganzzahliger Gitter motivierte neue Problemstellungen und erwies sich auch aus theoretischer Sicht als ein ergiebiges Gebiet. Das scheinbar banale Problem der Zerlegung eines Gitters in orthogonale Untergitter führte zur Entwicklung effizienter Algorithmen zum einen für das Zerlegungsproblem selbst, zum anderen für die Erzeugung eines Gitters aus einem großen Erzeugendensystem. Minkowskis zweiter Fundamentalsatz [Min96] aus der Geometrie der Zahlen führt zu Abschätzungen der Anzahl der benötigten arithmetischen Operationen und zu einer allgemeinen, quantitativen Aussage über die Größe minimaler Erzeugendensysteme von Gittern.

Der erste Teil der Arbeit leistet einen Beitrag zur konkreten Klassifikation ganzzahliger Gitter. Die Fundierung und ein Teil der Ergebnisse wurden in der gemeinsamen Arbeit *Classification of integral lattices with large class number* [SH98] mit Rudolf Scharlau publiziert. In diesem Kapitel wird auf die originären algorithmischen Aspekte der 2-Nachbarmethode fokussiert. Mit Hilfe von Knesers Nachbarverfahren gelang es H. Niemeier bereits 1973, das Geschlecht der geraden, unimodularen Gitter in Dimension 24 zu bestimmen [Nie73]. Er konnte dabei die starken Struktureigenschaften der Gitter in diesem Geschlecht ausnutzen, um die Nachbargitter zu identifizieren und zu unterscheiden. In allgemeineren Situationen sind separierende Invari-

anten aber nur mit großem algorithmischen Aufwand zu berechnen. In diesem Sinne äußern sich Conway und Sloane verhalten über den praktischen Nutzen von Knesers Methode in nichttrivialen Fällen.

> „However there is a geometric method used by Witt and Kneser which (after the work of Niemeier) is effective roughly until the sum of the dimension and the (determinant)$^{1/2}$ exceeds 24, beyond which point it seems that the forms are inherently unclassifiable.“ [CS92], S. 352 f.

Diese Einschätzung erwies sich als zu pessimistisch, denn die Ergebnisse in [SH98] und in dieser Arbeit zeigen, daß Klassifikationen deutlich jenseits dieser Schranke durchführbar sind.

Zwei verschiedene Gitter L und L' heißen benachbart, wenn sie ein gemeinsames Untergitter jeweils vom Index 2 enthalten. Bei geraden Gittern mit ungerader Determinante läßt sich mit Knesers Nachbarmethode ein Nachbargitter $L(v)$ aus L und einem Nachbarvektor $v \in L \setminus 2L$ mit Hilfe eines sogenannten Nachbarschrittes konstruieren. Da wir nur an den Isomorphieklassen von Nachbargittern interessiert sind, läßt sich die Konstruktion aller Nachbargitter bis auf Isomorphie auf eine endliche Auswahl von Nachbarvektoren v zurückführen. Sie werden als geeignete Repräsentanten aus der endlichen Menge von Orbits der induzierten Automorphismengruppe von L auf $L/2L$ ausgewählt.

Das Computerprogramm `tn` basiert auf einer Implementation dieser Methode. Daneben enthält es zahlreiche Analysefunktionen zur Bestimmung von Invarianten der konstruierten Gitter. Wegen der Komplexität der Berechnungen wurde das Design und die Realisierung von `tn` sehr stark an Performanceaspekten ausgerichtet und deshalb nicht in einem existierenden Computeralgebrasystem vorgenommen.[1]

Bei den enumerativen Klassifikationen konzentrierte sich das Interesse auf die ℓ-modularen Gitter mit $\ell = 3, 5, 7, 11$ in Dimensionen bis zu 14 und die Bestimmung ihrer modularen und extremalen Gitter. Ein bemerkenswertes Ergebnis aus theoretischer Sicht ist ein Satz über das Geschlecht 7^6. Die Klassifikation mit `tn` lieferte den Beweis für die Nichtexistenz eines vermuteten extremalen Gitters.

Das Programm `tn` kann zur Systematisierung der Ergebnisse eine Vielzahl von Invarianten eines Gitters berechnen. Eine bisher wenig beachtete Invariante ist die Fourier-Transformation der Längenfunktion eines Gitters auf $L/2L$. Erste Vermutungen, daß diese Fourier-Transformierte nur einen kleinen Träger besitzt, bewahrheiteten sich nicht. Es scheinen jedoch im allgemeinen nur relativ wenig verschiedene Fourierkoeffizienten in der transformierten Funktion eine Rolle zu spielen. Mit `tn` lassen sich das Spektrum und die Fourier-Transformierte der Längenfunktion eines Gitters für eine feste Basis bestimmen.

[1] Seit 1999 existiert auch eine Implementierung der Basisfunktionalität der 2-Nachbarmethode in dem Computeralgebrasystem MAGMA.

Die Nachbarschaftsmethode erweist sich auch in Fällen, in denen die Theorie keinen Erfolg garantiert, als erstaunlich robust. Bei Gittern mit gerader Determinante ist der oben beschriebene Nachbarschritt nicht immer möglich. Dennoch läßt sich beispielsweise das (bereits bekannte) Geschlecht des Barnes–Wall-Gitters [SV94] in der Dimension 16 berechnen. Man kann beobachten, daß in höheren Dimensionen der Nachbarschaftsgraph, der gebildet wird aus den Isomorphieklassen von Gittern als Knoten und den Äquivalenzklassen von Nachbarvektoren als Kanten, für jedes Gitter so viele Kanten enthält, daß das Kollabieren einzelner Nachbarbildungen in der Regel nicht ergebnisrelevant ist.

Das zweite Kapitel beginnt mit einer Diskussion des Zerlegbarkeitsproblems. Ein Gitter heißt zerlegbar, wenn es sich als direkte Summe nichttrivialer, orthogonaler Untergitter schreiben läßt. Die Zerlegung eines Gitters in unzerlegbare Gitter ist eindeutig bis auf die Reihenfolge der Summanden. Der konstruktive Beweis dieses Satzes stammt von Martin Kneser [Kne54]. Seine Konstruktion, eine Analogie zum Sieb von Erathostenes, dient primär dem Beweis der Behauptung, aber nicht der konkreten Durchführung. So erweist sie sich unter dem Blickwinkel der Berechenbarkeit als nicht ausreichend effizient. Wir zeigen in Satz 3.17, wie ein inkrementelles Konstruktionsverfahren statt der Benutzung des Siebverfahrens zu praktikablen Laufzeiten führt, und geben eine Schranke für die Anzahl der algorithmischen Operationen des Gitterzerlegungsproblems an.

Das Fehlen eines Basisergänzungssatzes und eines Steinitz'schen Basisaustauschsatzes in der Modulsituation verkompliziert die algorithmische Behandlung der Basisberechnung eines Gitters außerordentlich. Das Hauptergebnis in diesem Kapitel ist ein Überdeckungssatz für Erzeugendensysteme von Gittern, der aus dem zweiten Fundamentalsatz von Minkowski folgt. Es wird gezeigt, daß in typischen Situationen der praktischen Berechnung von Gittern ein mininales Erzeugendensystem nur aus wenig mehr Vektoren als dem Gitterrang besteht. Die quantitative Aussage liefert Satz 3.14: ist $S \subseteq L$ ein minimales Erzeugendensystem eines Gitters L auf dem n-dimensionalen euklidischen Raum E mit Vektoren, die in der euklidischen Norm nicht länger als $B \in \mathbb{R}$ sind, dann ist $|S| \leqslant n + \log_2(n!(\frac{B}{\min L})^n)$. Mit Hilfe dieses Satzes wird gezeigt, daß das algorithmische Problem der Konstruktion von Gitterbasen aus großen Erzeugendensystemen durch das angegebene inkrementelle Verfahren effizient lösbar ist im Vergleich zu etablierten Methoden wie den Variationen des LLL-Algorithmus und der Berechnung der Hermite-Normalform. Tests mit ganzzahligen Zufallsgittern unterstreichen die praktische Relevanz (siehe die Laufzeitvergleiche auf Seite 44).

Das dritte Kapitel behandelt die Berechnung von Gitterquantizern. Dabei handelt es sich um Approximationen, die aus der Informationstheorie motiviert werden. Ein Quantizer eines Gitters ordnet jedem Punkt im euklidischen Raum den nächstliegenden Gitterpunkt zu. Er ist also im wesentlichen durch die Geometrie der einem Gitter-

punkt nächstliegenden Punkte, der sogenannten Voronoïzelle bestimmt. Die Qualität eines Gitterquantizers aus ingenieurwissenschaftlicher Sicht wird durch das dimensionslose zweite normalisierte Trägheitsmoment der Voronoïzelle (Definition 4.15) bewertet. Dies ist ein skaliertes Integral über die quadratische Vektorlängenfunktion in der Voronoïzelle des Ursprungs. Die Abstandsfunktion wird algorithmisch durch das Verfahren von Fincke und Pohst auf Nebenklassen [FP85] ausgewertet.[2] Wir beschreiben die bekannten, grundlegenden Eigenschaften von Voronoïzellen und die approximative Berechnung ihrer Gitterquantizer mit Monte-Carlo-Integration. Ihre Berechnung ist ebenfalls in `tn` implementiert. Alternativ lassen sich Gitterquantizer und Quantizer zu Kugelpackungskonstruktionen aus nichtlinearen Codes (Typ A Konstruktion nach [CS92]) mit einem Programm für das Computeralgebrasystem MAGMA berechnen. Gitterquantizer sind bis in die jüngste Vergangenheit nicht intensiv untersucht worden. In einer neueren Arbeit erzielten Agrell und Eriksson [AE98] eine Reihe von zum Teil kontraintuitiven Ergebnissen über Gitterquantizer.

Für das Zustandekommen dieser Arbeit schulde ich vielen Personen Dank. An erster Stelle ist hier Herr Prof. Dr. Rudolf Scharlau zu nennen. Seine Betreuung gab dieser Arbeit den Anfang und die Richtung. Er war mir in all den Jahren ein aufmerksamer und geduldiger Gesprächspartner. Die Diskussionen mit meinem Kollegen Frank Vallentin haben mir viele Anknüpfungspunkte an die Informatik eröffnet. Bernd Souvignier hat seine Implementationen der Berechnung von Erzeugern der Automorphismengruppe eines Gitters und des Isometrietests zweier Gitter freundlicherweise diesem Projekt zur Verfügung gestellt. Ohne die wunderbaren Entwicklungstools emacs, gcc und gbd aus dem GNU-Projekt von Richard Stallmans Free Software Foundation wäre aus diesem Projekt wahrscheinlich nicht ein so flexibles und leistungsfähiges Programm wie `tn` entstanden. Die Hochschulrechenzentren der Universitäten Bielefeld, Dortmund und Duisburg haben mir Kapazitäten auf ihren Supercomputern zur Verfügung gestellt. Der Besuch des AT&T Shannon Labs in New Jersey auf Einladung von Neil Sloane lenkte mein Interesse auf Gitterquantizer. Die Universität Dortmund hat mir für die Anfertigung dieser Dissertation eine Assistentenstelle zur Verfügung gestellt.

Ganz besonders danke ich meiner Frau Dr. Eva-Maria Kaffanke dafür, daß diese Arbeit nach langer Zeit doch noch ihre Vollendung gefunden hat.

[2] In der Praxis von Telekommunikationsanwendungen wie bei Analog-Digital-Wandlern spielt neben der Güte des Trägheitsmomentes auch die Effizienz der Implementation der Approximation eine Rolle.

2 Klassifikation ganzzahliger Gitter

2.1 Einleitung

In diesem Kapitel wird das Nachbarverfahren zur Klassifikation ganzzahliger Gitter mit großer Klassenzahl beschrieben. Die zugrundeliegende Methode wurde von M. Kneser entwickelt [Kne57]. H.-V. Niemeier klassifizierte mit ihrer Hilfe 1973 die geraden unimodularen Gitter in Dimension 24 [Nie73].

Obwohl dieser Meilenstein in der Klassifikation der ganzzahligen quadratischen Formen die Bedeutung von Knesers Methode beweist, wurde der praktische Nutzen in „generischeren“ Situationen als eher gering eingeschätzt (siehe [CS92], S. 352). In der gemeinsamen Arbeit mit R. Scharlau *Classification of integral lattices with large class number* [SH98] wurde gezeigt, daß eine algorithmisch optimierte Version des Nachbarverfahrens eine Vielzahl von Geschlechtern zugänglich macht, die mit bisherigen Methoden nicht klassifizierbar waren. Besonders für die Suche nach Gittern mit bestimmten Eigenschaften kann ein Exhaustionsverfahren wie die Nachbarmethode äußerst nützlich sein. Zu dieser Arbeit ist das Computerprogramm `tn` entstanden, dessen Funktionsweise im folgenden erläutert wird. In Abschnitt 2.2 werden die theoretischen Grundlagen der Konstruktion von Gittern durch die 2-Nachbarmethode zusammengefaßt. In Abschnitt 2.5 wird die Funktion von `tn` genauer beschrieben und mit einigen markanten Resultaten in Abschnitt 2.7 illustriert.

Knesers Verfahren garantiert unter gewissen Nebenbedingungen eine vollständige Klassifikation von (Spinor-)Geschlechtern. Die 2-Nachbarmethode läßt sich aber auch mit Erfolg bei Gittern mit gerader Determinante anwenden (Abschnitt 2.6), welche die Voraussetzungen des Satzes von Kneser nicht erfüllen. Ein interessantes Beispiel ist die Klassifikation des Geschlechtes des Barnes-Wall-Gitters (Abschnitt 2.7.2).

2.2 Die Nachbarmethode

Im folgenden seien alle Gitter ganzzahlig mit ungerader Determinante auf dem euklidischen n-dimensionalen Raum $(V, (-,-))$, sofern nicht etwas anderes vermerkt ist.

Definition 2.1. *Zwei Gitter L und L' werden Nachbarn genannt, falls*

$$L/(L \cap L') \cong \mathbb{Z}/2\mathbb{Z} \cong L'/(L \cap L').$$

Aus dieser Definition folgt unmittelbar das

Lemma 2.2. *Zwei benachbarte Gitter L und L' besitzen dieselbe Determinante.*

Beweis. Nach der Determinanten-Index-Formel gilt

$$\det L = [L : L \cap L']^2 \cdot \det L \cap L' = [L' : L \cap L']^2 \cdot \det L \cap L' = \det L'. \qquad \square$$

Die effektive Konstruierbarkeit von Nachbarn klärt das

Lemma 2.3.

i) Für ein gerades Gitter L und einen Vektor $v \in L \setminus 2L$ ist das Gitter

$$L(v) := L_v + \frac{1}{2}\mathbb{Z}v \quad \textit{mit} \quad L_v := \{x \in L \mid (x, v) \in 2\mathbb{Z}\}$$

ein Nachbar von L. Man nennt $L(v)$ den Nachbarn von L bezüglich des Nachbarvektors v.

ii) Wenn die Gitter L und L' Nachbarn sind, dann gilt $L(2w) = L'$ für jedes $w \in L' \setminus L$.

Beweis. Zu i) Wir zeigen zunächst $|L/L_v| = 2$. Die auf $\overline{L} := L/2L$ induzierte Bilinearform $(\overline{v}, \overline{w}) := \overline{(v, w)}$ besitzt Diskriminante 1 und ist insbesondere regulär, weil die Diskriminante von $(-,-)$ ungerade ist. Wir nehmen an, daß $L = L_v$. Dann gilt $(v, w) \in 2\mathbb{Z}$ für alle $w \in L$. Somit ist $(\overline{v}, \overline{w}) = 0$ für alle $\overline{w} \in \overline{L}$, also ist $\overline{v} = 0$ und damit $v \in 2L$, was im Widerspruch zur Voraussetzung steht.

Aus der Exaktheit der Sequenz

$$0 \longrightarrow L_v \longrightarrow L \xrightarrow{x \mapsto (\overline{v}, \overline{x})} \mathbb{Z}/2\mathbb{Z} \longrightarrow 0$$

folgt, daß $|L/L_v| = 2$.

Dann gilt offenbar auch $L/(L \cap L(v)) = 2$. Da L gerade ist, folgt, daß $v \in L_v$, aber $\frac{1}{2}v \notin L_v$, also ist $L(v)/(L \cap L(v)) \cong \mathbb{Z}/2\mathbb{Z}$ und damit sind L und $L(v)$ Nachbarn.

Zu ii). Es sei $w \in L' \setminus L$ für die Nachbarn L und L'. Dann folgt aus $|L'/(L \cap L')| = 2$, daß $2w \in L \setminus 2L$. Wie oben ergibt sich aus

$$0 \longrightarrow L_{2w} \longrightarrow L \xrightarrow{x \mapsto (\overline{2w}, \overline{x})} \mathbb{Z}/2\mathbb{Z} \longrightarrow 0,$$

daß $|L/L_{2w}| = 2$. Weil L' ganzzahlig ist, gilt für jedes $x \in L'$, daß $(x, 2w) = 2(x, w) \in 2\mathbb{Z}$, also insbesondere $L \cap L' \subseteq L_{2w}$. Da $L/L \cap L' \cong \mathbb{Z}/2\mathbb{Z} \cong L/L_{2w}$, muß $|L_{2w}/L \cap L'| = 1$ sein, also ist $L_{2w} = L \cap L'$. Aus $w \in L' \setminus L$ und $|L'/L_{2w}| = 2$ erhalten wir schließlich $L' = L_{2w} + \frac{1}{2}\mathbb{Z}(2w) = L(2w)$. □

Bemerkung 2.4. $L(v)$ ist nicht notwendigerweise ganzzahlig. Dieser Sachverhalt wird im folgenden Lemma genauer untersucht.

2.3 Die Konstruktion des Nachbarschaftsgraphen

Lemma 2.5. *Sei L ein gerades Gitter und $v \in L \setminus 2L$. Dann ist*

i) $L(v)$ ein ganzzahliges Gitter genau dann, wenn $(v, v) \in 4\mathbb{Z}$,

ii) $L(v)$ ist ein gerades Gitter genau dann, wenn $(v, v) \in 8\mathbb{Z}$.

Beweis.

i) Sei $(v, v) \in 4\mathbb{Z}$. Mit L ist auch L_v ganzzahlig. Aus $(L_v, \frac{1}{2}v) = \frac{1}{2}(L_v, v) \subseteq \mathbb{Z}$ und $(\frac{1}{2}v, \frac{1}{2}v) = \frac{1}{4}(v, v) \subseteq \frac{1}{4}4\,\mathbb{Z} = \mathbb{Z}$, folgt, daß auch $L(v)$ ganzzahlig ist. Die Umkehrung folgt unmittelbar aus der Ganzzahligkeit von L und $(v, v) \in \mathbb{Z} \Leftrightarrow (\frac{1}{2}v, \frac{1}{2}v) \in 4\mathbb{Z}$.

ii) Sei $(v, v) \in 8\mathbb{Z}$. Mit L ist auch L_v gerade. Dann ist wegen $(\frac{1}{2}v, \frac{1}{2}v) \in \frac{1}{4}\,8\mathbb{Z} = 2\mathbb{Z}$ auch $L(v)$ gerade. Die Umkehrung folgt unmittelbar aus der Ganzzahligkeit von L und $(v, v) \in 2\mathbb{Z} \Leftrightarrow (\frac{1}{2}v, \frac{1}{2}v) \in 8\mathbb{Z}$.

□

Die beiden folgenden Lemmata erlauben es, die Konstruktion aller Nachbarn eines Gitters bis auf Isomorphie auf ein endliches Problem zurückzuführen.

Lemma 2.6. *Es seien $w, w' \in L \setminus 2L$. Dann gilt $L(w) = L(w')$, falls $w - w' \in 2L_w$.*

Beweis. Es sei $z := \frac{w-w'}{2} \in L_w$. Dann folgt aus $\overline{(x, w)} = \overline{(x, w)} - \overline{(x, 2z)} = \overline{(x, w - 2z)} = \overline{(x, w')}$ für alle $x \in L$, daß $L_w = L_{w'}$. Da $2L \subseteq L_w = L_{w'}$, folgt

$$L(w) = L_w + \frac{1}{2}\mathbb{Z}w = L_w + \frac{1}{2}\mathbb{Z}(w' + 2z) = L_{w'} + \frac{1}{2}\mathbb{Z}w' = L(w').$$

□

Lemma 2.7. *Sei L ein ganzzahliges, gerades Gitter und $v \in L \setminus 2L$. Dann existiert ein $y \in L$ mit $(v, y) \notin 2\mathbb{Z}$ und für alle $w \in \overline{v} \in L/2L$ gilt entweder $L(w) = L(v)$ oder $L(w) = L(v + 2y)$. Weiter ist eines dieser Gitter gerade, das andere ungerade.*

Beweis. Da $\overline{v} \neq 0$ erfüllt sogar für jede gegebene Basis von L einer der Basisvektoren die von y geforderte Eigenschaft. Ansonsten wäre die induzierte Form $\overline{(-,-)}$ identisch 0 und besäße damit nicht die Determinante 1.

Falls $(v,v) \equiv 0 \pmod 8$, so ist $L(v)$ ein gerades Gitter und $L(v+2y)$ wegen

$$(v+2y, v+2y) = (v,v) + 4(v,y) + 4(y,y) \equiv 4 \pmod 8$$

ein ungerades Gitter. Insbesondere sind $L(v)$ und $L(v+2y)$ verschieden. Also gilt nach dem vorangehenden Lemma für alle $w \in \overline{v}$ wegen $[2L : 2L_v] = 2$ entweder $L(w) = L(v)$ oder $L(w) = L(v+2y)$. □

Damit braucht man lediglich $2^n - 1$ Repräsentanten der Klassen von $L/2L \cong \mathbb{F}_2^n$ als Nachbarvektoren zu berücksichtigen, um alle Nachbargitter von L zu konstruieren. Dies reduziert die Komplexität der Auswahl der Nachbarvektoren auf ein endliches Problem.

Dieses Argument beweist das

Korollar 2.8. *Ein Gitter besitzt bis auf Isometrie höchstens $2^n - 1$ Nachbargitter.*

Das nächste Lemma zeigt, daß Nachbarvektoren, die in demselben Orbit unter der Automorphismengruppe des Gitters liegen, isometrische Nachbarn erzeugen.

Lemma 2.9. *Ist $\sigma \in O(L)$ und $w \in L \setminus 2L$, dann ist $L(w\sigma) = L(w)\,\sigma$.*

Beweis. Für $x \in L$ gilt

$$x \in L_{w\sigma} \Leftrightarrow (x, w\sigma) \in 2\mathbb{Z} \Leftrightarrow (x\,\sigma^{-1}, w) \in 2\mathbb{Z} \Leftrightarrow x\,\sigma^{-1} \in L_w \Leftrightarrow x \in L_w\,\sigma.$$

□

Bemerkung 2.10. Aufgrund des vorhergehenden Lemmas können wir uns bei der Auswahl von Nachbarvektoren auf diejenigen Gittervektoren beschränken, die einerseits in verschiedenen Orbits unter der Automorphismengruppe $O(L)$ liegen und die andererseits in verschiedenen Klassen modulo $2L$ liegen. Das kommutative Diagramm in Abbildung 2.1 veranschaulicht diesen Sachverhalt. Alle enthaltenen Abbildungen sind surjektiv. Wegen der Kommutativität des Diagramms kann man die Berechnung aller Nachbargitter (bis auf Isometrie) von L vereinfachen, indem man zunächst zu einem Repräsentantensystem von $L/2L \cong \mathbb{F}_2^n$ übergeht und darauf die induzierte Automorphismengruppe $\overline{O(L)}$ in kanonischer Weise operieren läßt. Hierbei bildet man Vektoren aus L nach F_2^n ab, vermöge $\sum_{i=1}^n \alpha_i b_i \mapsto (\overline{\alpha_1}, \cdots, \overline{\alpha_n})$ für eine beliebige, aber feste Basis $\{b_1, \cdots, b_n\}$ von L.

Die Konstruktion aller Isomorphieklassen von Nachbargittern des Gitters L ist folgendermaßen möglich:

$$\begin{array}{ccc} L & \xrightarrow{O(L)} & \{v\,O(L) \mid v \in L\} \\ \downarrow & & \downarrow \\ {\mathbb{F}_2}^n & \xrightarrow{\overline{O(L)}} & \{\overline{v}\,\overline{O(L)} \mid \overline{v} \in \mathbb{F}_2^n\} \end{array}$$

Abbildung 2.1: Die Orbits unter $\overline{O(L)}$

i) Wähle eine beliebige, von nun an fixierte Basis von L.

ii) Identifiziere $L/2L$ mit $\mathbb{F}_2^n$ und berechne alle Orbits von $\overline{O(L)}$ auf $\mathbb{F}_2^n$ bezüglich der gewählten Basis.

iii) Wähle aus jedem Orbit in $\mathbb{F}_2^n$ unter der Operation von $\overline{O(L)}$ einen Vektor $\overline{v} \in \mathbb{F}_2^n$ und suche ein beliebiges Urbild $v \in L$ auf.

iv) Bilde den geraden Nachbarn von L bezüglich v, sofern er überhaupt existiert.

Die Isomorphieklasse des Nachbargitters hängt nicht vom gewählten Repräsentanten ab, deshalb sprechen wir auch gelegentlich von einer Nachbarbildung „bezüglich der Klasse $\overline{v}$“ anstatt „bezüglich des Vektors v“.

Mit der iterierten Bildung von Nachbarn eines gewählten Gitters werden weitere Gitter konstruiert. Bei geraden Gittern besteht diese Menge aus einer Vereinigung von echten Spinorgeschlechtern, bei allen Beispielen und Resultaten in diesem Kapitel sogar aus einem ganzen Geschlecht (siehe [SH98], S. 742; [O'M71], §102 und [Kne56], Satz 2 bis Satz 5). Es bietet sich an, die Nachbarschaft von Gittern als Inzidenzrelation in einem Graphen zu beschreiben. Dieser Graph ist ungerichtet, weil nach Lemma 2.3 ii) Nachbarschaft eine symmetrische Relation ist.

Da wir im folgenden an einer Klassifikation dieser Geschlechter interessiert sind, definieren wir den Nachbarschaftsgraphen nicht auf der Menge aller Gitter, sondern nur innerhalb eines Geschlechts und nur bis auf Isomorphie der Gitter.

Definition 2.11. *Für ein Gitter L sei $[L]$ die Klasse aller zu L isometrischen Gitter. Dann nennen wir den ungerichteten Graphen (V, E) mit $V = \{[L] \mid L \in \mathcal{G}\}$ und*

$$E = \{\, \{[L_1], [L_2]\} \mid L_1, L_2 \in \mathcal{G}, \quad L_1 \text{ und } L_2 \text{ sind Nachbargitter}\}$$

den Nachbarschaftsgraphen *von $\mathcal{G}$.*

2.4 Konstruktion einer Basis eines Nachbarn

Sei $\{b_1, \cdots, b_n\}$ eine Basis des geraden Gitters L und $v \in L \setminus 2L$. In Algorithmus 1 wird eine Basis des geraden Nachbarn bezüglich $\overline{v}$ konstruiert. Im Anschluß zeigen wir die Korrektheit des Verfahrens.

Input: Eine Basis $\{b_1, \cdots, b_n\}$ von L und $v = (v_1, \cdots, v_n) \in L \setminus 2L$ mit $(v, v) \in 4\mathbb{Z}$.
Output: Eine Basis $\{b'_1, \cdots, b'_n\}$ des geraden Nachbarn von L bezüglich v, falls dieser existiert, ansonsten eine Fehlermeldung
if $(v, v) \equiv 2 \pmod 8$ oder $(v, v) \equiv 6 \pmod 8$ **then**
  return FAILURE
end if
// 1. Initialisierung.
if $(v, v) \equiv 4 \pmod 8$ **then**
  wähle i mit $(v + b_i, v + b_i) \equiv 0 \pmod 8$
  $v \leftarrow v + b_i$
end if
//2. Normalisiere v.
$k \leftarrow \min\{i \mid v_i \notin 2\mathbb{Z}\}$
wähle x mit $x \cdot v_k \equiv 1 \pmod 8$
$v \leftarrow x \cdot v$
// 3. Konstruktion der Basisvektoren.
wähle $m \neq k$ mit $(v, b_m) \notin 2\mathbb{Z}$
$b'_k \leftarrow \frac{1}{2}v, \quad b'_m \leftarrow 2b_m$
for $i = 1, \cdots, n, \quad k \neq i \neq m$ **do**
  if $(b_i, v) \in 2\mathbb{Z}$ **then**
    $b'_i \leftarrow b_i$
  else
    $b'_i \leftarrow b_i + b_m$
  end if{ // also $(b'_i, v) \in 2\mathbb{Z}$ }
end for
return b'

Algorithmus 1: Eine Basis des geraden Nachbarn von L bzgl. v

In der Initialisierung im ersten Schritt wird v, wenn möglich, auf eine durch 8 teilbare Norm gebracht. Ist $(v, v) \equiv 0 \pmod 8$, so bilden wir den Nachbarn $L(v)$. Ist $(v, v) \equiv 4 \pmod 8$, so wählen wir nach Lemma 2.7 ein b_i mit $(v, b_i) \notin 2\mathbb{Z}$ und bilden $L(v + 2b_i)$. Ist hingegen $(v, v) \equiv 2$ oder $6 \pmod 8$, so existiert kein ganzzahliger Nachbar bezüglich $\overline{v}$.

Wir suchen den ersten ungeraden Koeffizienten v_k in v auf und multiplizieren v mit einem geeigneten Skalar x, so daß $x \cdot v_k \equiv 1 \pmod 8$. In der Implementation wird x durch einen einfachen Table–Lookup bestimmt.

$$x = \begin{cases} 3, & \text{falls } v_k = 3 \\ 5, & \text{falls } v_k = 5 \\ 7, & \text{falls } v_k = 7 \end{cases} .$$

Da alle skalaren Faktoren ungerade sind, gilt immer noch $L(v) = L(xv)$. Anschließend wird ein $m \neq k$ bestimmt mit $(v, b_m) \notin 2\mathbb{Z}$. Angenommen, ein solches m existiere nicht. Sei $M = \langle \{b_i\}_{i \neq k} \rangle_{\mathbb{Z}}$. Dann liegt nach der Annahme $\overline{v} \in \overline{M}^{\perp}$, also ergibt sich beim Übergang zum orthogonalen Komplement $\overline{M} \subseteq \overline{v}^{\perp}$. Aus Dimensionsgründen gilt $\overline{M} = \overline{v}^{\perp}$. Da $(v, v) \in 2\mathbb{Z}$, folgt $v \in \overline{v}^{\perp} = \overline{M}$. Wegen $\overline{v_k} = \overline{1}$ folgt $v \notin M$, was ein Widerspruch ist.

Aus der Konstruktion folgt, daß $b'_i \in L_v$, für $i \neq k$ und daß $b'_k \in L(v)$, also gilt $\langle b'_1, \cdots, b'_n \rangle_{\mathbb{Z}} \subseteq L(v)$. Die Übergangsmatrix von $\{b_1, \cdots, b_n\}$ zu $\{b'_1, \cdots, b'_n\}$ besitzt folgende Gestalt (hierbei nehmen wir ohne Beschränkung der Allgemeinheit an, daß $k = 1$ und $m = n$ ist):

$$\begin{pmatrix} \frac{1}{2} & 0 & 0 & \cdot & \cdot & \cdot & 0 \\ * & 1 & 0 & \cdot & \cdot & \cdot & 0 \\ * & 0 & 1 & 0 & \cdot & \cdot & 0 \\ \cdot & \cdot & 0 & 1 & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ * & 0 & 0 & \cdot & 0 & 1 & 0 \\ * & * & * & \cdot & * & * & 2 \end{pmatrix}$$

Die Determinante dieser Übergangsmatrix ist gleich 1; damit folgt aus dem Vergleich der Determinanten $\det\langle b'_1, \cdots, b'_n \rangle_{\mathbb{Z}} = \det L = \det L(v)$, daß $\langle b'_1, \cdots, b'_n \rangle_{\mathbb{Z}} = L(v)$.

2.4.1 Ungerade Gitter und der duale Nachbarschaftsgraph

Nach Lemma 2.7 gehört zu jedem geraden Nachbarn L' eines geraden Gitters L genau ein ungerader Nachbar L''. Dieser Sachverhalt läßt sich auch von einem elementaren algebraischen Standpunkt aus betrachten. Sei $v \in L \setminus 2L$ mit $L(v) = L'$ und $y \in L$ mit $(v, y) \notin 2\mathbb{Z}$ und $(v, v) \equiv 0 \pmod 4$, d.h., $L(v + 2y) = L''$. Wir setzen $M = L + \frac{1}{2}\mathbb{Z}v$ und erhalten mit M/L_v die elementarabelsche Gruppe der Ordnung 4 mit den drei nichttrivialen Elementen $L/L_v, L'/L_v, L''/L_v$. Mit L und v ist die Gruppe M/L_v und damit das Paar aus dem ungeraden und aus dem geraden Nachbarn von L eindeutig bestimmt. Dies liefert uns eine (wohlbekannte) Interpretation des

Nachbarschaftsgraphen. Eine Kante im Graphen kennzeichnet eine Nachbarschaft von geraden Gittern. Dieser Kante ordnen wir das entsprechende ungerade Gitter zu. Der duale Graph ist also eine Teilmenge des Nachbarschaftsgraphen für ungerade Gitter. Bei ungeraden Gittern gibt es im allgemeinen keine Korrespondenz von geraden und ungeraden Nachbarn, wie man leicht an dem Fall sieht, daß v in dem orthogonalen Komplement eines ungeraden Teilgitters $N \subset L$ liegt. Dann ist auch $N \subseteq L_v$, also sind beide Nachbarn von L bezüglich v ungerade.

In Spezialfällen kann der Übergang zum dualen Graphen äußerst nützlich sein. R.E. Borcherds benutzt in [Bor92] den Nachbarschaftsgraphen der Niemeier-Gitter, um die ungeraden unimodularen Gitter in Dimension 24 zu klassifizieren.

2.5 Das Programm `tn`

Die konkrete Berechnung des Nachbarschaftsgraphen erweist sich als eine komplexe Aufgabe. Dies liegt an der Vielzahl der involvierten Teilprobleme und verwendeten Zahlsysteme. Im Programm selbst werden drei verschiedene Arithmetiken eingesetzt. Der aufwendigste Teil in den Berechnungen von `tn` besteht in der Auswahl der geeigneten Vektoren zur Nachbarbildung. Das Implementationsziel von `tn` ist, alle Nachbarbildungen berechnen zu können, solange es überhaupt möglich ist, die Automorphismengruppen der beteiligten Gitter zu berechnen und die Gitter auf Isometrie zu prüfen.

Das Programm `tn` behandelt nur ganzzahlige Gitter, so daß die meisten Berechnungen in hardwarenaher 32-Bit Integerarithmetik erfolgen können.

Einige Routinen verlangen die Berechnung der „kurzen Vektoren“ eines Gitters; damit ist eine Auflistung oder Abzählung aller Gittervektoren bis zu einer bestimmten Länge, meist der Länge des größten Basisvektors, gemeint. Dazu benutzen wir den Algorithmus von U. Fincke und M. Pohst [FP85] mit den Ergänzungen von H. Cohen (Algorithm 2.7.7 in [Coh96], S. 105) in einer Implementation mit Doublearithmetik.

Die Berechnungen zur Auswahl der Vektoren zur Nachbarbildung werden, wie in Bemerkung 2.10 beschrieben, hauptsächlich im Vektorraum $L/2L \cong \mathbb{F}_2^n$ vorgenommen. Für diesen Zahlbereich wurde eine eigene Implementation in Binärarithmetik implementiert, die in Abschnitt 2.5.2 vorgestellt wird.

2.5.1 Die Berechnung der Automorphismengruppe eines Gitters

Für die Berechnung von Automorphismengruppen und Isometrieen von Gitteren werden in `tn` die Programme `autom` und `isom` verwendet, die uns Bernd Souvignier

freundlicherweise zur Verfügung stellte.[1] Sie sind Implementation der von B. Souvignier und W. Plesken in [PS97] beschriebenen Verbesserungen der generischen Sims-Stabilisatorkettenmethode in Gittern.

Die Routinen `autom` und `isom` sind vom Rechenzeitbedarf her die aufwendigsten Programmteile von `tn`.

2.5.2 Rechnen in $L/2L$

Da $\overline{L} = L/2L \cong \mathbb{F}_2^n$, liegt es nahe, die Rechnungen in $\overline{L}$ in hardwarenaher Binärarithmetik zu implementieren. Die Berechnung aller Orbits unter $\overline{O(L)}$ erfolgt durch Anwendung aller von `autom` berechneten und anschließend induzierten Erzeuger. Aus diesem Grund ist die Effizienz dieser Routinen im Hinblick auf Geschwindigkeit und Speicherplatzbedarf in der Praxis ein kritischer Erfolgsfaktor des Programms `tn`. Die Implementation erfolgte in der Programmiersprache „C“ [KR88].

Vektoren aus $\mathbb{F}_2^n$ werden mit dem Datentyp `unsigned int` identifiziert, der in den meisten Prozessorarchitekturen in einem 32-Bit Wort implementiert ist. In Dimensionen jenseits von 32 sind aus mehreren Gründen keine kompletten Klassifikationen zu erwarten. Auf der einen Seite dauern Automorphismengruppenberechnungen und Isometrietests bereits sehr lange, auf der anderen Seite ist die Klassenzahl bei Geschlechtern in diesen Dimensionen sehr groß. Beispielsweise weiß man durch Anwendung der Maßformel (s.u.), daß es mehr als achtzig Millionen unimodulare gerade Gitter in Dimension 32 gibt.

Bei der Berechnung der Orbits von $\overline{O(L)}$ auf $\overline{L}$ verwenden wir zwei verschiedene Kodierungen für Vektoren in $\mathbb{F}^n$, je nach Optimierungsziel, Rechenzeitbedarf und Speicherplatzbedarf. Bei der Matrix-Vektor-Multiplikation wird ein Vektor von $\overline{L}$ als Bitfolge in einem Objekt vom Typ `unsigned int` kodiert. Eine Matrix besteht aus einem Array von n solcher Objekte. Für die Markierung der Orbits von $\overline{O(L)}$ verwenden wir im Vektorraum $\overline{L}$ ein Kompressionsverfahren, bei dem jeder Vektor in $\overline{L}$ nur durch ein einziges Bit repräsentieren wird, und die Kodierung und Dekodierung sehr effizient erfolgt. Die folgenden beiden Beispiele illustrieren die benutzten Techniken bei der Matrix-Vektor Multiplikation in $\overline{L}$ (2.5.2.1) und der Bestimmung eines kanonischen Repräsentanten eines Orbits von $\overline{O(L)}$. Der Quelltext von `tn` ist hier für den Zweck der Darstellung aufbereitet worden. Fehlerbehandlung und der Umgang mit Spezialfällen wie $\dim L \leqslant 3$ führen zu etwas komplizierterem Code.

[1] Bernd Souvignier implementierte später neben `isom` und `autom` die 2-Nachbarmethode in dem Computeralgebrasystem MAGMA.

2.5.2.1 Matrix-Vektor-Multiplikation in $\overline{L}$

Als Beispiel für die Effizienz der Arithmetik in $\overline{L}$ geben wir hier die verwendete Routine für eine Matrix-Vektor-Multiplikation an.

```
 1  /* Matrix-Vektor Multiplikation in L/2L */
 2
 3  typedef unsigned int vector_2;
 4
 5  /* A definition for matrices over GF(2). */
 6  typedef
 7  struct struct_matrix_2
 8  {
 9    int dim;        /* dimension */
10    vector_2 *v;    /* rows of the matrix */
11  }
12  *matrix_2;
13
14  void
15  matrix_vector_2 (vector_2 *imv, const matrix_2 m,
16                   const vector_2 v)
17       /* matrix_vector_2 computes the image of */
18       /* the column vector v under the */
19       /* operation of the matrix m and puts the */
20       /* result in imv. */
21  {
22    int i, j;
23    vector_2 tmp;
24
25    for (i = 0, *imv = 0; i < m->dim; i++){
26       /* Compute (m->v[i],v) and set result
27          in i-th bit of im */
28       tmp = m->v[i] & v;
29       for (j = __WORDSIZE / 2; j >= 1; j /= 2)
30          tmp ^= tmp >> j;
31       *imv |= (tmp & 1) << ((m->dim - 1) - i);
32      }
33  }
```

Listing 2.1: Effiziente Matrix-Vektor Multiplikation in $\overline{L}$

Nachdem der Bildvektor `imv` mit dem Nullvektor initialisiert ist, wird in der `for`-Schleife jeweils die i-te Komponente von `imv` ermittelt. Hierzu berechnet man das Skalarprodukt der i-ten Zeile `m->v[i]` der Matrix `m` mit dem Vektor `v` und weist es dem i-ten Bit in `im` zu. Man setzt j auf die halbe Anzahl der Bits in einem `vector_2`.

Die Zeile 30 wird in der Schleife durchlaufen, um durch mehrfache Faltung mit einer XOR-Operation das Ergebnis im 0-ten Bit von `tmp` zu erhalten. Die oberen Bits spielen für das Ergebnis dann keine Rolle mehr. Das 0-te Bit wird in Zeile 31 an die $(n-i)$-te Stelle in `im` kopiert. Um bereits gesetzte Bits nicht zu löschen, erfolgt dies mit einer logischen OR-Operation auf `im`.

2.5.2.2 Orbits unter Gruppenoperation

Die allgemeine Standardmethode, einen Orbit unter einer Gruppenoperation zu bestimmen, ist der elegante `union-find` Algorithmus mit Pfadkompression [TvL84]. In unserer Situation können wir zusätzlich ausnutzen, daß die Orbits aller Vektoren in $\overline{L}$ bestimmt werden müssen. Dies erlaubt ein effizienteres Vorgehen sowohl in Bezug auf Speicherplatz als auch auf Rechenzeit. Dabei identifizieren wir $\overline{L}$ mit den Koordinatenvektoren, die durch eine feste Basis von L gegeben sind und kodieren

jeden Koordinatenvektor auf ein bestimmtes Bit. Diese Bits bilden eine Folge in einem zusammenhängenden Speicherbereich. Dadurch wird auf $\overline{L}$ eine totale Ordnung induziert, die bei unserer Kodierung mit der üblichen lexikographischen Ordnung zusammenfällt. Dies erlaubt eine sehr schnelle Kodierung und Dekodierung von Vektoren in $\overline{L}$. In `tn` sind für eine Liste von Vektoren in $\overline{L}$ die folgenden Methoden für Listenoperationen implementiert.

`insert_vector (v)` Füge den Vektor `v` in die Liste ein.

`delete_vector (v)` Lösche den Vektor `v` aus der Liste.

`lookup_vector (v)` Prüfe, ob der Vektor sich bereits in der Liste befindet und liefere dementsprechend `true` oder `false` zurück.

`find_min` Liefere den bezüglich der lexikographischen Ordnung kleinsten Vektor in der Liste zurück.

Diese Operationen sind aus Effizienzgründen in `tn` zum Teil inline implementiert, d.h. nicht durch explizite Funktionsaufrufe. Der Klarheit halber formulieren wir beispielhaft zwei dieser Operationen als Funktionsaufrufe.

Die Vektoren werden in einem Array von `unsigned char`, einem 8-Bit Datentyp kodiert. Dabei geben die führenden $n-3$ Bits die Position des Bytes im Array an, in welchem der Vektor kodiert wird, und die letzten drei Bits werden durch die Position in dem entsprechenden Byte kodiert.

Als erstes Beispiel beschreiben wir die Operation `insert_vector(v)`. Um das den Vektor `v` beschreibende Bit aufzufinden, berechnen wir zunächst den Offset im Array. Die führenden $n-3$ Bit von `v`, interpretiert als Binärzahl, geben die Position des entsprechenden Bytes im Array an (im Source Zeile 14 `l[v»3]`). In diesem Byte setzen wir das Bit, dessen Position durch das Bitmuster letzten 3 Bit von `v` in der üblichen Binararithmetik beschrieben wird. Hierzu maskieren wir alle Bits von `v` bis auf die letzten 3 aus (`(v & 7)`) und verwenden das Ergebnis, um das erste Bit um die entsprechende Anzahl zu verschieben (im Source: `1 << (v & 7)`).

```
1  /* insert vector v in list l */
2
3  typedef unsigned int vector_2;
4  typedef *unsigned char vlist;
5
6  /* Insert v in list l. Idea: Lookup the byte
7  in vlist l whose offset is binary coded by
8  the leading n-3 bits in v.
9  Now mark in this byte the bit whose number
10 is binary coded by the last 3 bits. */
11
12 void
13 insert_vector (vlist l, vector_2 v){
14    l[v>>3] |= (1 << (v & 7))\label{insert_vector_fast_decoding}
15 }
```

Listing 2.2: Operationen in $\overline{L}$: `insert_vector (v)`

Bei der folgenden Routine `find_min(L)`, die einen Vektor aus der Liste `L` zurückliefert, müssen wir die Fehlersituation der leeren Liste abfangen. Der Nullvektor spielt in $\overline{L}$ keine Rolle, weil er nicht zur Nachbarbildung verwendet werden kann. Wir benutzen deshalb sein Kodifikat „0“ als Flag, um der aufrufenden Funktion mitzuteilen, daß die Liste leer ist.

```
1  int dim; /* the dimension of \overline{L} */
2  typedef unsigned int vector_2;
3  typedef *unsigned char vlist;
4  /* Lookup the "smallest" vector in l
5     and return it. */
6
7  vector_2
8  find_min (vlist l){
9    int i, t;
10 /* Lookup the first non-empty byte
11    in the array L */
12   for (i = 0; i < (1 << (dim - 3)); i++)
13     if (l[i] != 0)
14       break;
15
16   for (t = 0; t < 8; t++)
17     if (((l[i] >> t) & 1) == 1)
18       break;
19   if (t == 8)
20     return 0; /* There is no umarked vector. */
21   else
22     return (8 * i + t);
23     }
```

Listing 2.3: Operationen in $\overline{L}$: `find_min`

Um in der Liste den kleinsten Vektor aufzufinden, d.h., in dem Array das erste gesetzte Bit zu finden, suchen wir die Liste linear durch. Diese Implementation ist für unsere Zwecke in der Laufzeit hinreichend effizient, denn wir können jeweils 8 Bits zugleich prüfen, in dem wir jeweils ein ganzes Byte betrachten und testen, ob es gleich 0 ist.[2] Beispielsweise muß in der Dimension 20 lediglich ein zusammenhängender Speicherbereich von höchstens 128 kByte durchlaufen werden. Der damit verbundene Aufwand ist irrelevant gegenüber den anderen Operationen in `tn` wie beispielsweise der Basisreduktion. Dieses Durchlaufen des Arrays geschieht in der folgenden Schleife.

```
for (i = 0; i < (1 << (dim - 3)); i++)
 if (l[i] != 0)
   break;
```

Nun enthält die Variable `i` den richtigen Offset und wir suchen in dem `i`-ten Byte des Arrays das erste gesetzte Bit.

```
for (t = 0; t < 8; t++)
  if ((l[i] >> t) & 1)
    break;
```

[2] Die naheliegende Verbesserung, gleich 32 oder mehr Bits durch einen Cast auf einen größeren Datentyp wie Integer zu überprüfen, führt nicht zu wesentlichen Geschwindigkeitssteigerungen und macht die Portierung auf andere Betriebssysteme komplizierter.

Als Ergebnis geben wir den Vektor zurück, dessen Bitmuster der Binärzahl $8i + t$ entspricht.

2.5.3 Die diskrete Fourier–Transformierte der Längenfunktion

2.5.3.1 Die diskrete Fourier–Transformation auf endlichen abelschen Gruppen

Das Programm `tn` enthält eine Vielzahl von Analysefunktionen, um Invarianten eines Gitters zu bestimmen. Darunter befindet sich die bisher wenig untersuchte Funktion $\mathcal{F}_{L/2L}(\ell)$, die Fourier-Transformierte der Längenfunktion ℓ. Diese Zerlegung der Längenfunktion, welche einer Nebenklasse in $L/2L$ die Norm eines kürzesten in ihr enthaltenen Vektors zuordnet, in eine Linearkombination aus irreduziblen Charakteren von $L/2L$ erlaubt bei vielen Gittern eine recht einfache Beschreibung dieser Funktion. Die Zerlegung selbst ist abhängig von der Wahl der zugrundeliegenden Basis in L beziehungsweise in $L/2L$; die Vielfachheiten der Koeffizienten in der Linearkombination, das sogenannte Spektrum, sind hingegen unabhängig von der Wahl einer Basis. Wir beschreiben zunächst die Fourier-Transformation und ergänzen diese mit einigen Beispielen.

Sei G eine endliche abelsche Gruppe, $G^\vee$ die Menge der irreduziblen Charaktere von G und seien f, f' komplexwertige Funktionen auf G. Die irreduziblen Charaktere $G^\vee$ bilden eine Orthonormalbasis bezüglich des Skalarproduktes $\langle f, f' \rangle = \frac{1}{|G|} \sum_{g \in G} f(g)\overline{f'(g)}$ im Raum der Klassenfunktionen G^* auf G. Deshalb kann man f folgendermaßen als Linearkombination der irreduziblen Charaktere schreiben, die eine Orthonormalbasis im Raum der Klassenfunktionen bilden:

$$f = \sum_{\chi \in G^\vee} \langle \chi, f \rangle \, \chi$$

Man nennt $\hat{f}(\chi) := \langle \chi, f \rangle = \frac{1}{|G|} \sum_{g \in G} \chi(g)\overline{f(g)}$ den Fourierkoeffizienten von χ bezüglich f.

Dadurch wird der Operator $\mathcal{F}$ der Fourier–Transformation auf der Gruppe G definiert

$$\begin{array}{rccc} \mathcal{F}_{\mathcal{G}}: & G^* & \to & (G^*)^* \\ & f & \mapsto & \hat{f} : \chi \mapsto < \chi, f > \end{array}$$

Wenn man die Charaktertafel von G als Matrix $(\chi(g))_{\chi \in G^\vee, g \in G}$ und f als Spaltenvektor $(f(g))_{g \in G}$ schreibt, dann berechnet sich $(\hat{f}(\chi))_\chi$ bis auf einen Faktor als einfache Matrizenmultiplikation durch

$$(\hat{f}(\chi))_\chi = \frac{1}{|G|} \cdot (\chi(g))_{\chi, g} \, \overline{(f(g))_g} \, .$$

2.5.3.2 Die diskrete Fourier-Transformierte der Längenfunktion eines Gitters

Für ein Gitter sei $\ell : L/2L \to \mathbb{R}$ mit $\ell(\overline{v}) = \min_{w \in \overline{v}}(w, w)$ als die Längenfunktion von L definiert.

2.5.3.2.1 Der Operator $\mathcal{F}_{L/2L}$ Die Gruppe $L/2L$ ist eine elementarabelsche Gruppe der Ordnung 2^n mit $n = \operatorname{rank} L$. Im ersten Schritt verifiziert man, daß sich ihre Charaktertafel in Form einer Hadamard-Matrix schreiben läßt. Diese Matrix ist bis auf einen skalaren Faktor gleich der Operatormatrix von $\mathcal{F}_{L/2L}$. Zunächst benötigt man eine Notation für die irreduziblen Charaktere von $L/2L \cong \mathbb{F}_2^n$. Die Charaktere dieser Gruppe lassen sich als mehrfache Tensorprodukte der Charaktere von $\mathbb{F}_2$ schreiben.

Die Charaktertafel von $\mathbb{F}_2$ enthält genau zwei Charaktere, den Hauptcharakter χ^+ und den Signumcharakter χ^-.

	0	1
χ^+	1	1
χ^-	1	-1

Tabelle 2.1: Die Charaktertafel von $\mathbb{F}_2$

Die irreduziblen Charaktere von $\mathbb{F}_2^n$ ergeben sich als n-fache Tensorprodukte der Charaktere χ^+ und χ^-. Zur Vereinfachung der Notation wird ein irreduzibler Charakter von $\mathbb{F}^n$ nur mit einer Folge von $+$ und $-$ indiziert und nicht als voll ausgeschriebenes Tensorprodukt, beispielsweise $\chi^{+-++} := \chi^+ \otimes \chi^- \otimes \chi^+ \otimes \chi^+$. Damit ergeben sich für $\mathbb{F}_2^1 n$ und $\mathbb{F}_2^2$ und $\mathbb{F}_2^3$ die Charaktertafeln wie sie in den Tabellen 2.1-2.3 beschrieben sind.

	00	01	10	11
χ^{++}	1	1	1	1
χ^{+-}	1	-1	1	-1
χ^{-+}	1	1	-1	-1
χ^{--}	1	-1	-1	1

Tabelle 2.2: Die Charaktertafel von $\mathbb{F}_2^2$

Das Programm `tn` berechnet die Fourier-Transformierte der Längenfunktion ℓ. Bei der Berechnung einer Vielzahl von Gittern stellt sich heraus, daß sich die Koeffizienten in der Zerlegung in irreduzible Charaktere auf nur wenige Werte bei großen Vielfachheiten konzentrieren.

	000	010	100	110	001	011	101	111
χ^{+++}	1	1	1	1	1	1	1	1
χ^{++-}	1	−1	1	−1	1	−1	1	−1
χ^{+-+}	1	1	−1	−1	1	1	−1	−1
χ^{+--}	1	−1	−1	1	1	−1	−1	1
χ^{-++}	1	1	1	1	−1	−1	−1	−1
χ^{-+-}	1	−1	1	−1	−1	1	−1	1
χ^{--+}	1	1	−1	−1	−1	−1	1	1
χ^{---}	1	−1	−1	1	−1	1	1	−1

Tabelle 2.3: Die Charaktertafel von $\mathbb{F}_2^3$

Die folgenden Beispiele sollen diese Beobachtung illustrieren und zu neuen Fragestellungen Anlaß geben.

2.5.3.3 Beispiele

i) Sei $L = (\mathbb{Z}^2, q(.,.))$ mit $G(q) = \begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix}$ Dann ist

$$(\hat{\ell}(g))_g = \frac{1}{|L/2L|} \cdot (\chi(g))_{\chi,g} \, (\ell(g))_g =$$
$$\frac{1}{4} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix} \cdot \begin{pmatrix} 0 \\ 2 \\ 1 \\ 3 \end{pmatrix} = \frac{1}{4} \begin{pmatrix} 6 \\ -4 \\ -2 \\ 0 \end{pmatrix}.$$

Also läßt sich ℓ in die folgende Linearkombination irreduzibler Charaktere zerlegen

$$\ell = \frac{1}{4}(6\chi^{++} - 4\chi^{+-} - 2\chi^{-+} + 0\chi^{--}).$$

ii) Sei $A_2 = (\mathbb{Z}^2, q(.,.))$ mit $G(q) = \begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix}$. Dann zerfällt ℓ analog des vorhergehenden Beispiels in die Linearkombination

$$\ell = \frac{1}{4}(6\chi^{++} - 2\chi^{-+} - 2\chi^{+-} - 2\chi^{--}).$$

iii) Für größere n zeigt sich bei der Zerlegung von ℓ für die Gitter A_n eine Verteilung der Koeffizienten auf nur wenige Werte.

a) Koeffizientenverteilung für A_{12}

Koeffizient	Vielfachheit
26624	1
-2048	13
0	4082

b) Koeffizientenverteilung für A_{13}

Koeffizient	Vielfachheit
57344	1
-4096	14
0	8177

c) Koeffizientenverteilung für A_{14}

Koeffizient	Vielfachheit
122880	1
-8192	15
0	16358

2.6 2-Nachbarn bei gerader Determinante

Bisher haben wir die 2-Nachbarbildung nur bei geraden Gittern mit ungerader Determinante untersucht. Den Grund für diese Einschränkung finden wir im Beweis von Lemma 2.3. L_v muß bei der Nachbarbildung ein echtes Untergitter von L sein; dies ist im allgemeinen aber nicht gegeben, wie wir im Beispiel $L = 2M$ für ein ganzzahliges Gitter M sehen. Dann gilt für jeden Vektor $v \in L$, daß $L = L_v$ ist. Für viele Vektoren v ist $L \neq L_v$ aber auch bei gerader Determinante erfüllt und damit ist die Konstruktion eines Nachbarn ebenfalls möglich.

Für die Konstruktion des 2-Nachbarschaftsgraphen eines Gitters L mit gerader Determinante, das aus den Vektoren $b_1, \cdots, b_n$ erzeugt wird, geht man im wesentlichen genau so vor wie im ungeraden Fall. Man berechnet die Menge aller Orbits von $\overline{O(L)}$ auf $\overline{L}$ und wählt aus jedem Orbit einen Repräsentanten $\overline{v}$ aus. Dann sucht man einen Basisvektor b_i von L mit $(\overline{v}, \overline{b_i}) = 1$ und $\overline{v} \neq \overline{b_i}$. Diese Eigenschaften sind unabhängig von der Auswahl des Repräsentanten v. Wie in Lemma 2.3 i) gezeigt, ist bei ungerader Determinante die Existenz eines solchen b_i gesichert, im geraden Fall jedoch nicht. Findet man keinen Basisvektor b_i mit diesen Eigenschaften, dann verwirft man diesen Orbit und geht zum nächsten über.

Dieses Vorgehen garantiert natürlich nicht eine erfolgreiche Klassifikation eines ganzen Geschlechtes, führt in der Regel aber in größeren Dimensionen (etwa ab Dimension 5 bis 8) zum Ziel. In fast allen geraden Gittern L höherer Dimension finden

sich genügend Orbits von Vektoren $v \in L$ mit $L_v \neq L$, welche die Existenz von 2-Nachbarn oft sicherstellen. Wir verfolgen hier mit `tn` einen pragmatischen Ansatz. `tn` berechnet den Nachbarschaftsgraphen, und eine erfolgreiche Verifikation mit der Maßformel sichert die Vollständigkeit des Ergebnisses (s. Abschnitt 2.8). Wie oben bereits beschrieben, sind jedoch Beispiele von Gittern ohne 2-Nachbarn auch in hohen Dimensionen konstruierbar.

Als prominentes Beispiel für eine erfolgreiche Klassifikation in einem Geschlecht mit gerader Determinante führen wir in Abschnitt 2.7.2 die Klassifikation des Geschlechtes des Barnes-Wall-Gitters an.

2.7 Ergebnisse

Die wichtigsten Ergebnisse, die mit Hilfe von `tn` erzielt wurden, sind in dem Artikel [SH98] dokumentiert. Wir geben hier eine kurze Zusammenfassung und einige neuere Resultate an.

2.7.1 ℓ-modulare Gitter

Ein n-dimensionales ganzzahliges Gitter L heißt *modular vom Level* ℓ, wenn es ein $\ell \in \mathbb{N}$ gibt mit $L \cong \sqrt{\ell} L^{\#}$ [Que95]. Es folgt unmittelbar, daß $\det L = \ell^{n/2}$; insbesondere ist n gerade, wenn ℓ quadratfrei ist. Viele der aus der Literatur bekannten Gitter sind modular, so z.B. das 12-dimensionale Coxeter-Todd-Gitter, das 16-dimensionale Barnes-Wall-Gitter und die geraden unimodularen Gitter E_8, das Gosset-Gitter, und Λ_{24}, das Leech-Gitter [CS92]. Aus der Theorie der Modulformen ergibt sich für modulare Gitter eine obere Schranke für das Minimum. Gitter, die dieses theoretische Maximum annehmen, heißen *extremal*. Für eine Einordnung und Diskussion dieses Themas verweisen wir auf den Übersichtsartikel [SSP99]. Die Existenz und die Eindeutigkeit modularer Gitter ist eine zentrale Frage in dieser Theorie. Einige der extremalen Gitter realisieren die besten bekannten Packungsdichten oder würden sie im Falle der Existenz übertreffen.

Für eine Berechnung des Nachbarschaftsgraphen mit `tn` benötigen wir ein Startgitter aus dem zu bestimmenden Geschlecht. Außer bei $\ell = 5$ haben wir bei den unten aufgeführten Fällen (hier gilt jeweils $\ell \equiv 3 \pmod 4$) mit dem direkten Produkt aus $n/2$ Gittern in der Dimension 2 mit Grammatrix $\begin{pmatrix} \ell/2 & 1 \\ 1 & 2 \end{pmatrix}$ begonnen.

Wir zitieren hier eine Liste mit den aus der Theorie der Modulformen berechneten möglichen maximalen Minima.

	n	4	6	8	10	12	14	16	18
$\ell = 3:$	min	2	2	2	2	4	4	4	4
$\ell = 5:$	min	2		4		4		6	
$\ell = 7:$	min	2	4	4	4	6	6	6	8
$\ell = 11:$	min	4	4	6	6	8	8		

Tabelle 2.4: Maximal mögliche Minima von modularen Gittern (aus [SH98])

2.7.1.1 Der Level $\ell = 3$

Die Klassenzahlen für die Dimensionen $2, 4, 6, 8, 10, 12$ sind jeweils $1, 1, 1, 2, 3, 10$; alle diese Gitter sind modular. Bis auf das Coxeter-Todd-Gitter in Dimension 12 mit Minimum 4 sind alle anderen Gitter reflektiv, d.h., sie enthalten ein Wurzelsystem von vollem Rang ([SB96]). Eine Methode, das Geschlecht des Coxeter-Todd-Gitters über die Konstruktion der möglichen Wurzelsysteme zu bestimmen, und die daraus folgende Klassifikation wurden bereits in [SV95, SV00] ohne Detailrechnungen angegeben.

Mit dem Programm `tn` läßt sich aufgrund des Konstruktionsverfahrens nicht nur eine Auflistung aller Gitter eines Geschlechts berechnen, sondern auch der vollständige Nachbarschaftsgraph beschreiben. Die folgende Tabelle enthält die Inzidenzmatrix des Nachbarschaftsgraphen des Geschlechtes des Coxeter-Todd-Gitters, bei welchem die Kanten mit der Anzahl der verschiedenen Orbits gewichtet wurden, die unter der jeweiligen Operation von $\overline{O(L)}$ auf $L/2L$ zu einer Nachbarschaftsbildung führen.

	K_1	K_2	K_3	K_4	K_5	K_6	K_7	K_8	K_9	K_{10}
K_1	1	1								
K_2	1	5	1	1	1	1				
K_3		1	4	1			1			
K_4		1	1	7	1		1	1	1	
K_5		1		1	5	1	1		1	
K_6	1			1	1	1				
K_7			1	1	1		6	1		1
K_8				1			1	4	1	
K_9				1	1			1	4	1
K_{10}							1		1	2

Tabelle 2.5: Der Nachbarschaftsgraph des Geschlechtes des Coxeter-Todd-Gitters mit den Bezeichnungen aus [SV95]

In der Dimension 14 gibt es 29 Gitter in diesem Geschlecht, darunter ein extre-

males mit Minimum 4. Seine Automorphismengruppe von der Ordnung $2^7 \cdot 3^6 \cdot 7 \cdot 13$ ist (die modulo der Spiegelung durch den Ursprung einfache Gruppe) $G_2(3) \cdot 2$ ([SH98], Proposition 3.2). Eine explizite Konstruktion dieses Gitters wird im $\mathbb{ATLAS}$ [CCN$^+$85], S. 60 beschrieben.

2.7.1.2 Der Level $\ell = 5$

Solche Gitter existieren nur in durch 4 teilbaren Dimensionen. In der Dimension 12 existiert ein interessantes Geschlecht mit Determinante 5^6, welches aus 48 Gittern besteht, darunter 40 modularen und 4 extremalen. Die Klassifikation dieses Geschlechtes wurde unabhängig von G. Nebe (unveröffentlicht) erzielt.

2.7.1.3 Der Level $\ell = 7$

Das Hauptergebnis des Klassifikationsprojektes in [SH98] ist der Beweis der Nichtexistenz eines vermuteten Gitters. In der Dimension 12 existiert kein extremales Gitter mit der Determinante 7^6. Dies ist das kleinste bekannte Beispiel für die Nichtexistenz eines möglichen extremalen Gitters.

2.7.1.4 Der Level $\ell = 11$

In den Dimensionen $4, 6, 8$ existieren jeweils $3, 5, 31$ Gitter, alle sind modular und in jedem Geschlecht gibt es ein eindeutiges extremales Gitter.

In der Dimension 10 gibt es neben dem Craig-Gitter ein weiteres extremales (Beobachtung von H.-G. Quebbemann, [Que95]) alle weiteren der insgesamt 297 Gitter besitzen ein kleineres Minimum. Siehe Abschnitt 2.7.3.1 für den 12-dimensionalen Fall.

2.7.2 Das Geschlecht des Barnes-Wall-Gitters Λ_{16}

Das Gitter Λ_{16} wurde 1959 von Barnes und Wall konstruiert [BW59] und ist mit Minimum 4 und Determinante 256 das dichteste bekannte Gitter in Dimension 16. Die Bezeichnung Λ_{16} ist durch seine Konstruktion als geschichtetes Gitter motiviert (siehe [CS92], Chapt. 6). Das Geschlecht des Barnes-Wall-Gitters wurde von R. Scharlau und B.B. Venkov [SV94] klassifiziert. Analog zur Klassifikation der geraden, unimodularen Gitter in Dimension 24 von B.B. Venkov in [Ven92] werden die Gitter im Geschlecht des Barnes-Wall-Gitters über ihr Wurzelsystem bestimmt.

Obwohl es sich hier um ein Geschlecht von gerader Determinante handelt, setzen wir `tn` mit den in Abschnitt 2.6 beschriebenen Modifikationen ein. Ausgehend von einer Erzeugermatrix für Λ_{16} untersuchen wir zunächst beispielhaft die Existenz

eines Nachbarvektors und präsentieren anschließend die Klassifikation und den 2-Nachbarschaftsgraphen des kompletten Geschlechtes des Barnes-Wall-Gitters.

Eine Erzeugermatrix A für Λ_{16} entnehmen wir [CS92], S. 130.

$$A = \frac{1}{\sqrt{2}} \begin{pmatrix}
4 & & & & & & & & & & & & & & & \\
2 & 2 & & & & & & & & & & & & & & \\
2 & 0 & 2 & & & & & & & & & & & & & \\
2 & 0 & 0 & 2 & & & & & & & & & & & & \\
2 & 0 & 0 & 0 & 2 & & & & & & & & & & & \\
2 & 0 & 0 & 0 & 0 & 2 & & & & & & & & & & \\
2 & 0 & 0 & 0 & 0 & 0 & 2 & & & & & & & & & \\
2 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & & & & & & & & \\
2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & & & & & & & \\
2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & & & & & & \\
2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & & & & & \\
1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & & & & \\
0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & & & \\
0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & & \\
0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1
\end{pmatrix}$$

Wir betrachten den elften und zwölften Zeilenvektor aus der Matrix A und setzen $v = \frac{1}{\sqrt{2}}(1,1,1,1,0,1,0,1,1,0,0,1,0,0,0,0)$. v ist ein Minimalvektor, also primitiv und insbesondere nicht in $2\Lambda_{16}$ enthalten. Das Skalarprodukt von v mit dem elften Basisvektor $w = \frac{1}{\sqrt{2}}(2,0,0,0,0,0,0,0,0,0,2,0,0,0,0,0)$ ist gleich 1. Also ist $w \notin (\Lambda_{16})_v$, und damit ist gezeigt, daß $(\Lambda_{16})_v \neq \Lambda_{16}$. Da $(v+2w, v+2w) = 4+4+16 = 24 \equiv 0 \pmod 8$, erhalten wir nach Lemma 2.5 mit $\Lambda_{16}(v+2w)$ einen geraden Nachbarn von Λ_{16}.

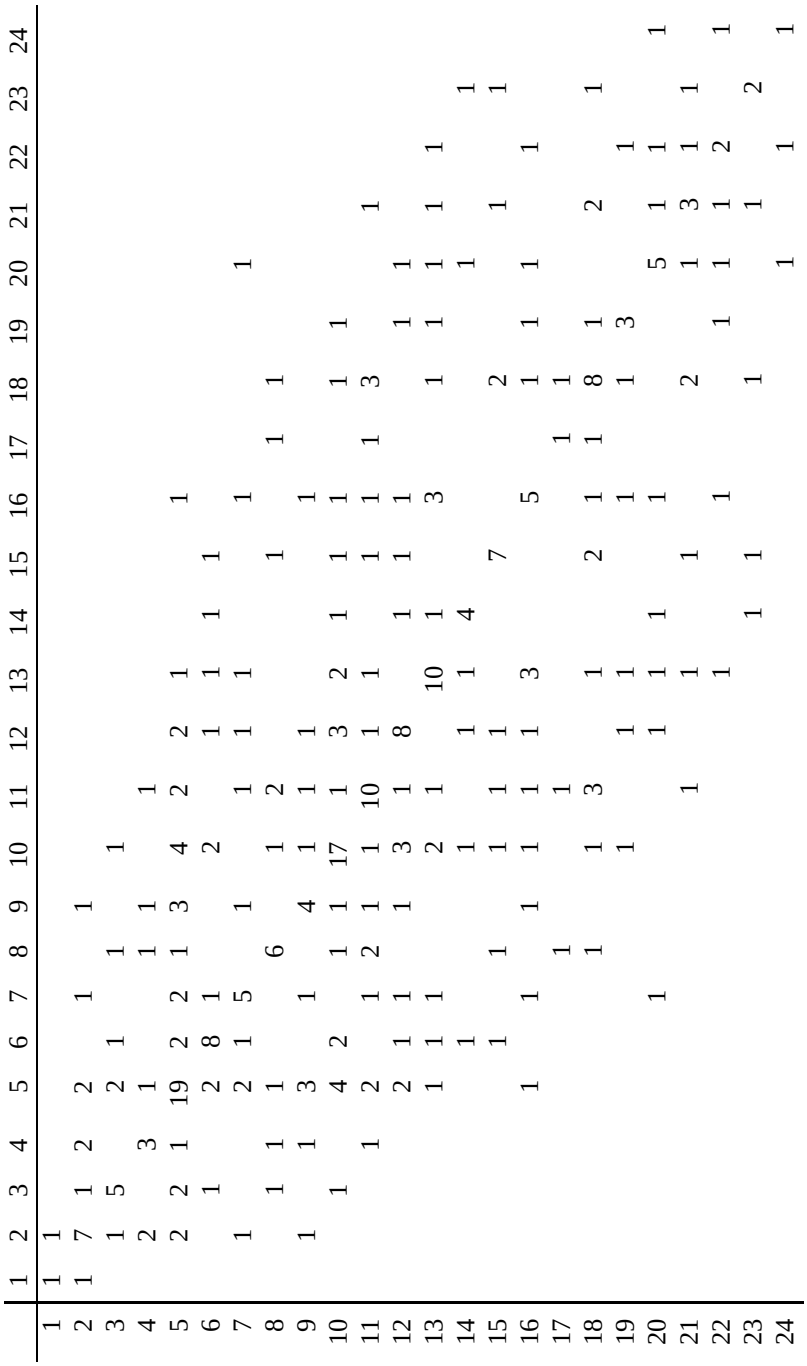
2.7.2.1 Der Nachbarschaftsgraph

Wir geben in Tabelle 2.6 den Nachbarschaftgraphen des Geschlechtes des Barnes-Wall-Gitters an. Die Bezeichnung der Gitter folgt der Klassifikation in [SV94]. Die Vielfachheiten in der Inzidenzmatrix geben die Anzahl der Orbits an, die zu einer Nachbarbildung führen.

2.7.3 Neuere Ergebnisse anderer Autoren

2.7.3.1 11-modulare Gitter in Dimension 12

Ein extremales 11-modulares Gitter mit Minimum 8 in Dimension 12 wäre geringfügig dichter als das dichteste bekannte Gitter in dieser Dimension, das Coxeter-Todd-Gitter. Das Maß dieses Geschlechtes liegt bei ungefähr 15096.99 ([SH98], S. 748),



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
1	1	1																						
2	1	7	1	2	2		1		1															
3		1	5		2	1		1		1														
4		2		3	1			1	1		1													
5		2	2	1	19	2	2	1	3	4	2	2	1			1								
6			1		2	8	1			2		1	1		1									
7		1			2	1	5		1		1	1	1			1				1				
8			1	1	1			6		1	2						1							
9		1		1	3		1		4	1	1	1				1								
10			1		4	2		1	1	17	1	3	2	1										
11				1	2		1	2	1	1	10	1	1			1		1						
12					2	1	1		1	3	1	8				1								
13					1	1	1			2	1		10	1										
14										1		1	1	4										
15						1		1		1	1	1	3		7			2						
16					1		1		1	1	1	1				5		1	1					
17								1		1	1						1	1						
18								1		1	3		1		2		1	8	1		2			
19										1		1	1			1		1	3			1		
20							1					1	1	1		1				5	1	1		1
21											1		1		1			2		1	3	1	1	
22													1		1					1	1	2		1
23																		1		1			2	
24																				1		1		1

Tabelle 2.6: Der Nachbarschaftsgraph des Geschlechtes des Barnes-Wall-Gitters

aber die meisten Gitter besitzen relativ große Automorphismengruppen, so daß man aufgrund der Maßformel (siehe Gleichung 2.1 auf Seite 26)) von vielen hunderttausend oder Millionen Gittern in diesem Geschlecht ausgehen kann. Eine vollständige Klassifikation mit `tn` erscheint weder sinnvoll noch aussichtsreich.

In [NV96] zeigen G. Nebe und B.B. Venkov, daß kein solches extremales Gitter existiert. In dem Beweis, der sich im wesentlichen auf analytische Methoden und Siegel-Modulformen vom Grad 2 stützt, wird `tn` als Hilfsmittel benutzt, um ein Erzeugendensystem des Vektorraums der vorkommenden Thetareihen zu konstruieren.

2.7.3.2 Die Nachbarmethode im hermiteschen Fall

In [Sch98] stellt A. Schiemann das Programm `hn` vor, daß die Nachbarmethode im hermiteschen Fall anwendet. Die erfolgreichen Klassifikationen sind in [Sch98] beschrieben.

2.7.3.3 Extremale Gitter

In dem Übersichtsartikel über extremale Gitter von R. Scharlau und R. Schulze-Pillot [SSP99] wurden die Ergebnisse mit `tn` und dem oben erwähnten `hn` berechnet.

2.7.3.4 Diverses

In der Doktorarbeit [Mis96] werden von M. Mischler einige Geschlechter mit `tn` bestimmt.

2.8 Verifikation der Ergebnisse

Bei allen Klassifikationsproblemen stellt sich die Frage nach der Korrektheit und der Vollständigkeit des Ergebnisses.

Das sogenannte Maß eines Geschlechts $m(\mathcal{G})$ kann einerseits mit analytischen Methoden hergeleitet werden (s. [CS88]), genügt andererseits nach Siegels Maßformel der Identität

$$m(\mathcal{G}) = \sum_{[L]\in\mathcal{G}} \frac{1}{|O(L)|}\,, \tag{2.1}$$

wobei die Summe über die Menge aller mit `tn` in $\mathcal{G}$ gefundenen Isomorphieklassen $[L]$ gebildet wird.

Diese Testfunktion erlaubt analog einer Quersumme in den oben zitierten Beispielen eine sehr verläßliche Überprüfung der berechneten Klassifikationen. Die Maßformel ist sensitiv gegenüber fehlenden Isomorphieklassen, einer fehlerhaften Berechnung der (Ordnung der) Automorphismengruppen und inkorrekten Isometrietests. Ei-

ne fehlerhafte Berechnung eines Geschlechtes, die von der Maßformel nicht aufgedeckt wird, ist nur möglich bei gleichzeitigem Auftreten von zwei Fehlern, die sich bei der Berechnung der Maßformel gegenseitig aufheben. Die an diesem Test beteiligten Programmteile sind die Generierung der Nachbarn, die Berechnung ihrer Automorphismengruppen und Isometrietests gegen die bisher gefundenen Gitter. Arbeiten zwei dieser drei Teile korrekt, so wird ein Fehler im dritten durch die Maßformel erkannt. Die Algorithmik des Isometrietests und der Berechnung der Automorphismengruppe sind sich in großen Programmteilen ähnlich, weil der Isometrietest mathematisch vergleichbar mit dem Auffinden eines nichttrivialen Automorphismus ist. In diesen Programmteilen befinden sich aber eine ganze Anzahl von Plausibilitätstests, die Inkonsistenzen entdecken können.

Die wenigen bereits bekannten Klassifikationen (s.o.) konnten sowohl für ungerade als auch für gerade Determinanten erfolgreich reproduziert werden. Die prominente Klassifikation der Niemeier-Gitter [Nie73] konnte mit `tn` nicht durchgeführt werden. Der Grund liegt in der algorithmischen Komplexität der Reduktion von Gitterbasen und in dem Umstand, daß die Automorphismengruppen in `autom` als Permutationsgruppen auf der Menge der kurzen Vektoren realisiert sind. Die Basis eines Nachbargitters, das mit dem oben beschriebenen Algorithmus konstruiert wird, enthält zumeist einige längere Vektoren. In der Routine `autom` müssen alle Vektoren des Gitters bis zu der Länge des längsten Basisvektors berechnet werden. In der Dimension 24 ist eine vollständige Aufzählung der kurzen Vektoren praktisch nur für Vektoren der Länge 2 und 4 möglich. Der LLL-Algorithmus reduziert aber nicht alle erzeugten Gitterbasen auf eine Basis, die nur Vektoren der Länge kleiner oder gleich 4 enthalten. So bleibt die Klassifikation von Niemeier ein Meilenstein in der Theorie der ganzzahligen quadratischen Formen, für die es auch heute kein algorithmisches Äquivalent gibt.

3 Über die Zerlegung von Gittern

3.1 Überblick

Im Mittelpunkt dieses Kapitels steht ein Algorithmus nach einer geometrischen Idee M. Knesers zur Zerlegung eines Gitters in eine orthogonale Summe von unzerlegbaren Untergittern. Eine vereinfachte Variante dieser Methode ermöglicht eine effiziente Konstruktion einer Gitterbasis aus einem großen Erzeugendensystem. Eine frühere Version dieses Kapitels ist in einen gemeinsamen Artikel [HV98] mit Frank Vallentin eingeflossen.

In diesem Kapitel ist es zweckmäßig, ein Gitter L als Untermodul im n-dimensionalen euklidischen Raum E versehen mit dem Standardskalarprodukt $(.,.)$ zu betrachten und den Aspekt der quadratischen Form als beschreibende Invariante eines Gitters in den Hintergrund treten zu lassen. Das Gitter L soll immer von vollem Rang sein, d.h., $\dim\langle L\rangle_{\mathbb{R}} = n$. Im folgenden bezeichnen wir hier mit der Norm eines Vektors $x \in E$ seine euklidische Norm $||x|| = \sqrt{(x,x)}$.

3.1.1 Das Zerlegungsproblem

Bei vielen algorithmischen Konstruktionen von Gittern hat man nicht genügend Kontrolle über die Eigenschaften und Invarianten des zu konstruierenden Gitters. Insbesondere ist es für Klassifikationszwecke interessant, ob ein Gitter die orthogonale Summe zweier niederdimensionalerer Gitter ist. In dieser Situation benötigen wir einen effizienten Algorithmus, der ein Gitter in eine orthogonale Summe von Teilgittern zerlegt.

Definition 3.1. *Ein nichttriviales Gitter L heißt* zerlegbar, *falls es (echte) Untergitter $L', L'' \subset L$ gibt mit $L = L' + L''$ und $(L_1, L_2) = 0$, andernfalls* unzerlegbar.

Bemerkung 3.2. Wir schreiben $L = L' \oplus L''$, falls L in L' und L'' zerlegbar ist. In diesem Kapitel werden wir das Symbol $\oplus$ unter Strapazierung der Notation **ausschliesslich** für eine innere direkte, orthogonale Summe verwenden.

Die Existenz einer Zerlegung in unzerlegbare Untergitter folgt unmittelbar durch Induktion über den Rang von L. In einem nicht konstruktiven Beweis zeigte Eichler

1952 in [Eic52] die Existenz einer bis auf die Reihenfolge der Summanden eindeutigen Zerlegung in unzerlegbare Untergitter. M. Kneser zeigte kurz darauf in einem elegantem Beweis [Kne54], daß diese Zerlegung effektiv berechenbar ist. Mittels eines Erathostenessiebes werden Gittervektoren eliminiert, die nicht in einem unzerlegbaren Untergitter liegen. Damit erhält man ein Erzeugendensystem für die unzerlegbaren Untergitter. In Abschnitt 3.3 geben wir eine Version von Knesers Beweis für Erzeugendensysteme an, welche die benutzten geometrischen Methoden illustriert.

Bei der Umsetzung von Knesers Beweis in einen Algorithmus stößt man auf zwei Probleme. Das Aussieben der zerlegbaren Vektoren ist ein aufwendiger Vorgang, dessen Laufzeit im wesentlichen quadratisch von der Größe des Erzeugendensystems abhängt. Des weiteren bildet die unzerlegbaren Vektoren im allgemeinen selbst ein sehr großes Erzeugendensystem für die unzerlegbaren Untergitter. Die Berechnung einer Gitterbasis aus einem Erzeugendensystem ist zwar ein algorithmisch gut verstandenes Problem, in der Praxis jedoch für große Erzeugendensysteme aufwendig. Die bekannten Algorithmen wie der MLLL und die Bestimmung der Hermite-Normalform sind für Erzeugendensysteme konstruiert, die nicht wesentlich größer als der Rang des erzeugten Gitters sind. In der oben beschriebenen Situation führt dies in der Regel zu einem relativ schlechten Laufzeitverhalten.

In Abschnitt 3.4 wird ein schnellerer Algorithmus für das Zerlegungsproblem als die Implementation von Knesers Methode angegeben; insbesondere erfolgt das Aussieben der erzeugenden Vektoren in linearer Laufzeit.

3.1.2 Minimale Erzeugendensysteme

Das zentrale Ergebnis in diesem Kapitel ist ein Überdeckungssatz für Erzeugendensysteme, der eine obere Schranke für die Größe eines minimalen Erzeugendensystems aus Vektoren mit einer beschränkten Länge angibt.

In Gittern mit nicht zu kleinem Minimum M (betrachtet im Verhältnis zur Determinante) besteht jedes minimale Erzeugendensystem nur aus wenig mehr als n Vektoren. Der Überdeckungssatz 3.14 quantifiziert dieses Ergebnis genauer: Ist $S \subset L$ ein minimales Erzeugendensystem von L mit Vektoren, die nicht länger als $B \in \mathbb{R}$ sind, dann ist $|S| \leqslant n + \log_2(n!(\frac{B}{M})^n)$.

Die algorithmische Hürde dieser Aufgabe rührt vom Fehlen eines Basisergänzungssatzes und eines Steinitz'schen Basisaustauschsatzes für die Modulsituation her. Man kann sich dies leicht an einem Erzeugendensystem veranschaulichen, in dem kein Vektor im Erzeugnis der anderen liegt, siehe etwa das Beispiel 3.21 auf Seite 43.

3.1.3 Der MLLL bei großen Erzeugendensystemen

Die in Kapitel 3.4 entwickelten Ideen für eine effiziente Lösung des Zerlegungsproblems werden in Kapitel 3.5 eingesetzt, um einen effizienten Algorithmus zu konstruieren, der aus einem großen Erzeugendensystem eine Gitterbasis berechnet. Die Methode besteht aus einem Meta-Algorithmus, der als Kern bekannte Routinen wie den MLLL oder die Berechnung der Hermite-Normalform benutzt. Diese Kernalgorithmen werden wiederholt aufgerufen, jedoch ausschließlich für minimale, also sehr kleine Erzeugendensysteme. Für den Meta-Algorithmus können bei großen Erzeugendensystemen schärfere asymptotische Schranken gezeigt werden als für die zugrundeliegenden Kernalgorithmen. Die praktische Relevanz erkannt man an den Laufzeitgewinnen, die in den graphischen Auswertungen zweier Experimente (s. S. 44, Abb. 3.3 und Abb. 3.4) veranschaulicht werden.

3.2 Unzerlegbare Vektoren

Bevor wir uns der konkreten Berechnung einer Zerlegung von L in unzerlegbare Untergitter zuwenden, betrachten wir qualitative Kriterien für die Zerlegbarkeit bzw. Unzerlegbarkeit von Vektoren. Für ein Gitter mit der Zerlegung $L = L_1 \oplus \ldots \oplus L_r$ sollen unzerlegbare Vektoren zwei Eigenschaften genügen:

i) Wenn $x \in L_i$, $y \in L_j, i \neq j$ unzerlegbare Vektoren sind, dann sei $x + y \in L_i \oplus L_j$ zerlegbar.

ii) Die Menge der unzerlegbaren Vektoren in $L_i, 1 \leqslant i \leqslant r$ erzeugt L_i.

Der Begriff der Unzerlegbarkeit von Vektoren wurde zuerst von M. Kneser definiert [Kne54] und später von O'Meara [O'M80] verfeinert.[1]

Definition 3.3. *Sei* $v \in L$.

i) v *heißt* orthogonal zerlegbar, *falls Vektoren* $x, y \in L \setminus 0$ *existieren mit* $v = x + y$ *und* $(x, y) = 0$, *andernfalls* orthogonal unzerlegbar.

ii) v *heißt* spitz zerlegbar, *falls Vektoren* $x, y \in L \setminus 0$ *existieren mit* $v = x + y$ *und* $(x, y) \geqslant 0$, *andernfalls* spitz unzerlegbar.

iii) v *heißt* linear zerlegbar *oder einfach* zerlegbar, *falls Vektoren* $x, y \in L$ *existieren mit* $v = x + y$ *und* $||v|| > ||x|| \geqslant ||y||$, *andernfalls* (linear) unzerlegbar.

Lemma 3.4 ([O'M80]). *Sei* $v \in L$, *dann gilt:*
v *unzerlegbar* $\Rightarrow$ v *spitz unzerlegbar* $\Rightarrow$ v *orthogonal unzerlegbar.*

[1] Wir danken M. Kneser für seinen freundlichen Hinweis auf die Arbeit von O'Meara.

Beweis. Sei v orthogonal zerlegbar, dann ist v offenbar auch spitz zerlegbar. Sei v nun spitz zerlegbar, d.h., es gibt $x, y \in L \setminus 0$ mit $v = x + y$ und $(x, y) \geqslant 0$. Dann folgt aus $||v|| = \sqrt{x^2 + 2(x, y) + y^2} > ||x||$, daß v zerlegbar ist. □

O'Meara gibt eine Charakterisierung orthogonal und spitz zerlegbarer Vektoren an.

Satz 3.5 ([O'M80] 3.6, 3.7). *Sei $v \in L$.*

i) v ist genau dann orthogonal zerlegbar, wenn ein $w \in L$ existiert mit $w \neq \pm v$, $v - w \in 2L$ und $||v|| = ||w||$.

ii) v ist genau dann spitz zerlegbar, wenn ein $w \in L$ existiert mit $w \neq \pm v$, $v - w \in 2L$ und $||v|| \geqslant ||w||$.

Beweis. Wir folgen O'Mearas Beweis und zeigen zunächst Aussage ii). Sei $v \in L$.

$$
\begin{aligned}
& v \text{ spitz zerlegbar} \\
\iff \quad & \exists y \in L \setminus \{0, v\} \quad (y, v - y) \geqslant 0 \\
& (\text{Nun sei } w := v - 2y \text{ bzw. } y := \frac{1}{2}(v - w)). \\
\iff \quad & \exists w \in L \setminus \{\pm v\} \quad (\frac{1}{2}(v - w), \frac{1}{2}(v + w)) \geqslant 0 \text{ und } v - w \in 2L \\
\iff \quad & \exists w \in L \setminus \{\pm v\} \quad (v, v) - (w, w) \geqslant 0 \text{ und } v - w \in 2L \\
\iff \quad & \exists w \in L \setminus \{\pm v\} \quad ||v|| \geqslant ||w|| \text{ und } v - w \in 2L
\end{aligned}
$$

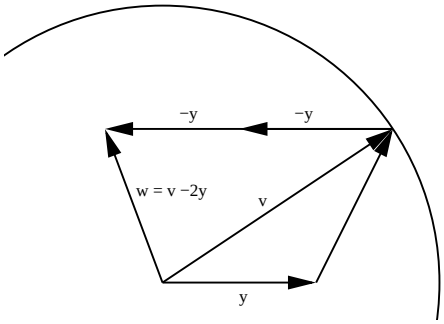
Aussage i) erhalten wir, wenn wir im vorangehenden Beweis „spitz“ durch „orthogonal“ und alle „$\geqslant$“ durch „$=$“ ersetzen. □

Damit hat O'Meara eine bemerkenswerte Eigenschaft von voronoïrelevanten Vektoren bewiesen, allerdings ohne den Zusammenhang zur Voronoïtheorie zu erwähnen.[2] Unseres Wissens ist dieser simple Zusammenhang bis heute nicht formuliert worden.

Korollar 3.6. *Ein Vektor $0 \neq x \in L$ ist genau dann spitz unzerlegbar, wenn er voronoïrelevant ist.*

Beweis. Nach Satz 3.5 ist x genau dann spitz unzerlegbar, wenn $\pm x$ die einzigen kürzesten Vektoren in der Nebenklasse $x + 2L$ sind. Die Behauptung folgt nun unmittelbar aus einem Kriterium von Voronoï, hier Satz 4.12, S. 51. □

[2] Siehe die Definition 4.10 auf Seite 50.



Für $v, y \in L$ mit $(y, v-y) \geqslant 0$ ex. ein $w \in L$ mit $||w|| \leqslant ||v||$ und $v - w \in 2L$.

Abbildung 3.1: Illustration zum Beweis von Satz 3.5 ii) „⇒“

Aus Satz 3.5 folgt ebenfalls, daß es höchstens $2 \cdot (2^n - 1)$ spitz unzerlegbare (und damit linear zerlegbare) Vektoren gibt ([O'M80] 3.17, 3.18). Eine einfache geometrische Überlegung zeigt bereits, daß lange Vektoren immer zerlegbar sind.

Lemma 3.7. *Sei R der Überdeckungsradius*[3] *von L und $v \in L$ mit $||v|| > 2R$. Dann ist v zerlegbar.*

Beweis I:. Sei $w \in L$ mit $||w - \frac{1}{2}v|| \leqslant R$. Dann gilt

$$||w|| \leqslant ||w - \frac{1}{2}v|| + ||\frac{1}{2}v|| \leqslant R + \frac{1}{2}||v|| < ||v||$$

und

$$||w - v|| \leqslant ||w - \frac{1}{2}v|| + || - \frac{1}{2}v|| \leqslant R + \frac{1}{2}||v|| < ||v||.$$

Also ist v zerlegbar, weil v die Summe der beiden echt kürzeren Vektoren w und $v - w$ ist. □

Mit einem Argument aus der Voronoïtheorie folgt die Behauptung unmittelbar.

Beweis II:. (F. Vallentin) Ist $||v|| > 2R$, so folgt aus $\frac{1}{2}v \notin V_0(L)$, daß v nicht voronoïrelevant ist und mit Satz 3.5 zerlegbar sein muß. □

Bemerkung 3.8. Der Begriff der orthogonalen Zerlegbarkeit von Vektoren wurde von M. Kneser analog zum Begriff der Zerlegbarkeit von Gittern definiert [Kne54]. In generischen Gittern sind alle Vektoren orthogonal unzerlegbar. Ein einfaches Beispiel

[3] Der Überdeckungsradius ist der maximal mögliche Abstand eines Punktes in E von einem Gitterpunkt in L, siehe Definition 4.3, S. 48.

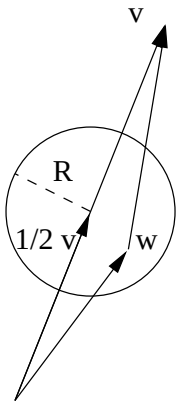


Abbildung 3.2: Lemma 3.7: Lange Vektoren sind zerlegbar

hierfür ist das von den Vektoren $(1,0),(\theta,1)$ mit $\theta^2 \notin \mathbb{Q}$ erzeugte Gitter. Insbesondere sind in einem eindimensionalen Gitter alle nichttrivialen Vektoren orthogonal unzerlegbar. O'Mearas Definition der linearen Zerlegbarkeit von Vektoren erscheint uns intuitiver.

3.3 Knesers Methode

Sei $B \in \mathbb{R}$, so daß $S := \{v_1, \cdots, v_s\} = \{v \in L \setminus 0 \mid ||v|| \leqslant B\}$ ein Erzeugendensystem von L ist. Auf der Menge der unzerlegbaren Vektoren I in S definieren wir den Orthogonalitätsgraph $\Gamma = (I, E)$ mit $E = \{\{v,w\} \in I \times I \mid (v,w) \neq 0\}$. Wir zerlegen Γ in seine Zusammenhangskomponenten $\Gamma_i = (I_i, E_i), 1 \leqslant i \leqslant r$ und setzen $L_i = \langle I_i \rangle_{\mathbb{Z}}$.

Satz 3.9 ([Kne54]). *Die Gitter $L_i, 1 \leqslant i \leqslant r$ sind unzerlegbar und die orthogonale Zerlegung*

$$L = L_1 \oplus \ldots \oplus L_r \tag{3.1}$$

ist eindeutig bis auf die Reihenfolge der Summanden.

Beweis. Sei $v \in S \setminus \langle I \rangle_{\mathbb{Z}}$ und $||v||$ minimal mit dieser Eigenschaft. Dann ist v wegen $v \notin I$ zerlegbar und es gibt $x, y \in S$ von echt kürzerer Norm als v mit $v = x + y$. Mindestens einer dieser beiden Vektoren ist nicht in $\langle I \rangle_{\mathbb{Z}}$, sonst wäre auch $v \in \langle I \rangle_{\mathbb{Z}}$. Dies widerspricht jedoch der Minimalität von $||v||$, somit ist $S \subseteq \langle I \rangle_{\mathbb{Z}}$ gezeigt. Daraus folgt die erste Behauptung $L = \sum_{i=1}^{r} L_i$.

$(I_i, I_j) = 0$ für $i \neq j$ impliziert $(L_i, L_j) = 0$. Daraus folgt mit der Definitheit von $(.,.)$ und der vorhergehenden Aussage, daß

$$L = L_1 \oplus \ldots \oplus L_r.$$

Seien $L', L'' \subseteq L_i$ für ein beliebiges, aber festes i mit $L_i = L' \oplus L''$. Für $v \in I_i$ gilt entweder $v \in L'$ oder $v \in L''$, ansonsten wäre v zerlegbar. Ohne Beschränkung der Allgemeinheit sei $v \in L'$. Wegen $(L', L'') = 0$ sind $L' \cap \Gamma$ und $L'' \cap \Gamma$ zwei nicht zusammenhängende Teilmengen von Γ. Also ist $L'' = 0$ und $L' = L_i$. Damit ist gezeigt, daß L_i unzerlegbar ist.

Da jedes L_i bereits durch einen in ihm enthaltenen unzerlegbaren Vektor eindeutig bestimmt ist, folgt die Eindeutigkeit der Zerlegung bis auf Reihenfolge der Summanden. □

Wir beschreiben nun Knesers Methode zur Konstruktion von Γ :

Initialisierung Sei $\{b_1, \cdots, b_n\}$ eine Basis von L. Setze $B = \max_i ||b_i||$ und berechne $S = \{v \in L \setminus 0 \mid ||v|| \leqslant B\}$.

Konstruiere I Für jedes Paar $u, v \in S$ mit $u + v \in S$ markiere $u + v$ als zerlegbar. Sei I die Menge aller nicht als zerlegbar markierten Vektoren.

Konstruiere E Füge $\{u, v\}$ für alle $u, v \in I$ in E ein, falls $(u, v) \neq 0$.

Bei diesem Verfahren müssen wir jedes Paar $\{u, v\}$ für alle $u, v \in S$ untersuchen. Diese $\binom{s}{2} \in O(s^2)$ Operationen sind für große s sehr aufwendig.

3.4 Der inkrementelle Algorithmus

Um für ein L_i in Gleichung 3.1 eine Basis effizient zu bestimmen, ist es nicht notwendig, alle unzerlegbaren Vektoren in L_i zu berechnen. Es genügt, ein Erzeugendensystem in den L_i aufzufinden, welches einerseits aus relativ wenigen Vektoren besteht und andererseits erlaubt, schnell zu verifizieren, daß sich alle unzerlegbaren Vektoren in den daraus erzeugten Gittern befinden. Dies motiviert den folgenden Algorithmus 2.

Die Korrektheit dieses Algorithmus zeigt

Satz 3.10. *Sei L ein n-dimensionales Gitter auf E, $B \in \mathbb{R}$, so daß die Menge $S = \{v_1, \cdots, v_s\} := \{v \in L \mid ||v|| \leqslant B\}$ ein Erzeugendensystem von L enthält. Dann berechnet Algorithmus 2 die Zerlegung von L in unzerlegbare Untergitter L_i, d. h., $L = L_1 \oplus \ldots \oplus L_r$.*

Für ein gegebenes System von paarweise orthogonalen Untergittern $L_1 \oplus \cdots \oplus L_k$ sei π_i die orthogonale Projektion auf den $\mathbb{R}$-Vektorraum, der von L_i aufgespannt wird. I und Γ seien wie oben definiert.

1: *Input:* $B \in \mathbb{R}$ mit einem Erzeugendensystem $S = \{v \in L \setminus 0 \mid ||v|| \leqslant B\}$ von L.
2: *Output:* Unzerlegbare Untergitter L_i mit $L = L_1 \oplus \ldots \oplus L_r$.
3: *// 1. Initialisierung.*
4: Wähle $v \in S$ mit $||v||$ minimal.
5: $S \leftarrow S \backslash \{v\}$, $k \leftarrow 1$, $L_k \leftarrow \langle v \rangle_{\mathbb{Z}}$
6: *// 2. Füge sukzessiv Vektoren aus S in die L_i ein.*
7: **while** $S \neq \emptyset$ **do**
8: Wähle $v \in S$ mit minimalem $||v||$.
9: $S \leftarrow S \backslash \{v\}$.
10: **if** $v \notin L_1 \oplus \ldots \oplus L_k$ **then**
11: *// v is „ergänzend", insbesondere unzerlegbar.*
12: $J \leftarrow \{i \in \{1, \cdots, k\} \mid \pi_i(v) \neq 0\}$
13: $\tilde{L} \leftarrow \mathbb{Z}v + \bigoplus_{i \in J} L_i$
14: *// Ordne die Liste der L_i neu.*
15: $\{L_1, \cdots, L_{k-|J|}\} \leftarrow \{L_j \mid j \notin J\}, \quad L_{k-|J|+1} \leftarrow \tilde{L}, \quad k \leftarrow k - |J| + 1$
16: **end if**
17: **end while**

Algorithmus 2: Zerlegung des Gitters L

Beweis. Sei $L_1 \oplus \ldots \oplus L_k$ mit den folgenden Eigenschaften bereits berechnet:

i) $L = \langle S \rangle_{\mathbb{Z}} + \sum_{i=1}^{k} L_i$,

ii) L_i ist unzerlegbar für $1 \leqslant i \leqslant k$,

iii) $(L_i, L_j) = 0$ für $i \neq j$.

Diese Eigenschaften sind nach der Initialisierung in Zeile 5 von Algorithmus 2 erfüllt.

Wir zeigen nun die Schleifeninvarianz (Zeile 7–16) dieser Eigenschaften. Sei $v \in S$ ein Vektor mit minimaler Norm in S wie in Zeile 8 gewählt. Ist $v \in L_1 \oplus \ldots \oplus L_k$, so können wir v verwerfen und die Eigenschaften i) – iii) bleiben erhalten. Sei nun $v \notin L_1 \oplus \ldots \oplus L_k$. Einen solchen Vektor nennen wir „ergänzend"[4].

Zu i): v wird hinzugefügt und es gilt

$$L = \langle S \setminus \{v\} \rangle_{\mathbb{Z}} + (\mathbb{Z}v + \sum_{i=1}^{k} L_i).$$

[4] Die Begriffsbildung ist lediglich lokal, d.h. nur im jeweiligen Schleifendurchlauf sinnvoll, weil diese Eigenschaft von der Menge der bereits abgearbeiteten Vektoren in S abhängt und nicht eine Invariante des Vektors v ist.

Insbesondere gilt $L = L_1 \oplus \ldots \oplus L_r$ nachdem alle Vektoren in S abgearbeitet sind.

Zu ii): v ist unzerlegbar, sonst gäbe es $x, y \in S$ von echt kürzerer Norm als v mit $v = x + y$. Nach Voraussetzung wären x, y jedoch schon in einer vorherigen Schleife abgearbeitet worden, woraus $x+y = v \in L_1 \oplus \ldots \oplus L_k$ folgen würde. (Insbesondere folgt aus der orthogonalen Summe $v = (v - \pi_i(v)) + \pi_i(v), 1 \leqslant i \leqslant k$, daß es keine nichttriviale Projektion von v auf $\langle L_i \rangle_{\mathbb{Q}}$ gibt, welche in L_i liegt.) Wir setzen $J = \{j \in \{1, \cdots, k\} \mid \pi_j(v) \neq 0\}$ und wählen Vektoren $v_j \in L_j, j \in J$ mit $(v_j, v) \neq 0$. Dann ist der Teilgraph $\{(v_j, v)\}_{j \in J} \subseteq \Gamma$ ein Stern und damit zusammenhängend, also ist $\bigoplus_{j \in J} L_j + \mathbb{Z}v$ unzerlegbar.

Zu iii): $\bigoplus_{i \in I \setminus J} L_i \oplus (\bigoplus_{j \in J} L_j + \mathbb{Z}v)$ ist eine orthogonale Zerlegung, weil $\pi_i(v) = 0$ für $i \notin J$.

Damit ist die Korrektheit von Algorithmus 2 gezeigt. □

3.4.1 Datenstrukturen und Laufzeitanalyse

In diesem Abschnitt beschreiben wir für die Realisierung von Algorithmus 2 geeignete Datenstrukturen. Wir werden in den Sätzen 3.17 und 3.19 Laufzeitanalysen für die hier entwickelten Algorithmen aufgrund dieser Datenstrukturen vornehmen. Im folgenden fassen wir E als Vektorraum aus Spaltenvektoren auf.

Eine zentrale Rolle in diesem Algorithmus spielen die orthogonalen Zerlegungen $L_1 \oplus \ldots \oplus L_k$, die in jedem Schleifendurchlauf konstruiert werden. Jedes Teilgitter L_i wird durch eine Gitterbasis $v_{i1}, \cdots, v_{ir_i} \subset E$ repräsentiert. Des weiteren benötigen wir eine (möglicherweise leere) Vektorraumbasis $w_1, \cdots, w_{n-l}$ des orthogonalen Komplements $\langle L_1, \cdots, L_k \rangle_{\mathbb{R}}^{\perp}$, $l = n - \sum_{i=1}^{k} r_i$. Nun setzen wir

$$A = \begin{pmatrix} v_{11} & \cdots & v_{kr_k} & w_1 & \cdots & w_{n-l} \end{pmatrix}^{-1} \in \mathfrak{M}_{n \times n}(\mathbb{R}).$$

Für jedes $x \in E$ gilt $v \in L_1 \oplus \ldots \oplus L_k$ genau dann, wenn die ersten $n - l$ Koeffizienten von Av in $\mathbb{Z}$ liegen und die restlichen l gleich 0 sind. Wir können somit in Zeile 10 des Algorithmus durch eine Matrix-Vektor-Multiplikation testen, ob der Vektor x in dem bereits konstruierten Untergitter liegt. Ist das der Fall, so überspringen wir ihn und gehen zum nächsten Schleifendurchlauf. Andernfalls berechnen wir wie folgt das neue Gitter $\tilde{L}$ mit einer seiner Basen, ordnen die Liste unserer unzerlegbaren Teilgitter L_i neu und berechnen die Indexmenge I und dann die Matrix A in der folgenden Weise neu: A ist eine Diagonalmatrix mit den Blöcken $\begin{pmatrix} v_{i1} & \cdots & v_{ir_i} \end{pmatrix}^{-1}$ auf der Hauptdiagonalen. Für jeden der ersten l Koeffizienten von Av, welcher nicht in $\mathbb{Z}$ liegt, suchen wir den entsprechenden Block auf (etwa j) und fügen j in J ein.

Anschließend konstruieren wir eine Basis von $\tilde{L} = \sum_{j \in J} L_j$ mit einer Variante[5] des bekannten LLL-Algorithmus [LLL82]. Die Berechnung einer Basis von $\tilde{L}$ kann effizient erfolgen, weil $\tilde{L}$ ein Erzeugendensystem $\{v_{ji} \mid j \in J, 1 \leqslant i \leqslant r_j\} \cup \{v\}$ aus höchstens $\sum_{j \in J} r_j \leqslant n + 1$ Vektoren besitzt. Nun ordnen wir die Liste der Gitter $\tilde{L}, L_i, i \notin J$ neu und indizieren sie von 1 bis $k - |J| + 1$. Schließlich berechnen wir A mittels Gaußtransformationen neu.

Die Effizienz dieses Algorithmus beruht auf der Tatsache, daß sich die meisten $v \in S$ in einer bereits konstruierten Zerlegung $L_1 \oplus \ldots \oplus L_k$ befinden und die aufwendige Berechnung des Gitters M und seiner Basis im Verhältnis zur Mächtigkeit von S nur selten notwendig ist. Im folgenden werden wir diese Behauptung genauer quantifizieren. Dazu benutzen wir hauptsächlich Minkowskis zweiten Fundamentalsatz.

3.4.1.1 Ketten von Untergittern

Satz 3.11 ([Min96]). *Sei L ein Gitter auf dem n-dimensionalen euklidischen Raum E mit den sukzessiven Minima $\lambda_1(L), \cdots, \lambda_n(L)$ und $B_n = \{x \in E \mid ||x|| \leqslant 1\}$ der n-dimensionale Ball mit Radius 1. Dann gilt*

$$\frac{2^n}{n!}\sqrt{\det L} \leqslant \lambda_1(L) \cdot \ldots \cdot \lambda_n(L) \cdot \operatorname{vol} B_n \leqslant 2^n \sqrt{\det L}. \tag{3.2}$$

Im Zentrum dieses Abschnittes steht der folgende

Satz 3.12. *Sei L ein Gitter mit Minimum M auf dem n-dimensionalen euklidischen Raum V. Das Untergitter $L' \subseteq L$ sei ebenso wie L von Vektoren erzeugt, die alle nicht länger als $B \in \mathbb{R}$ sind. Weiter gebe es eine endliche Kette von Untergittern $L_i \subseteq L$ mit*

$$L' = L_1 \subset \cdots \subset L_t = L \tag{3.3}$$

Dann gilt für die Länge t der Kette:

i) Ist $\operatorname{rank} L' = \operatorname{rank} L$, dann ist

$$t \leqslant \log_2(n! \left(\frac{B}{M}\right)^n).$$

ii) Wenn alle L_i aus Vektoren nicht länger als B erzeugt werden, dann ist

$$t \leqslant n + \log_2(n! \left(\frac{B}{M}\right)^n).$$

[5] Beispielsweise mit einer der in [Poh87] oder [BP89] beschriebenen Modifikationen. Siehe Abschnitt 3.5 für eine ausführliche Diskussion der Alternativen.

Beweis. Zu i): Sei $s_i = [L : L_i]$ der Index von L_i in $L, 1 \leqslant i \leqslant t$. Dann gilt nach der Determinanten-Index-Formel $s_i^2 = [L : L_i]^2 = \frac{\det L_i}{\det L}$, daß

$$\sqrt{\det L' / \det L} = [L : L'] = s_1 > \cdots > s_t = [L : L] = 1.$$

s_{i+1} ist wegen $s_{i+1} \cdot [L_{i+1} : L_i] = s_i$ ein Teiler von s_i. Also ist die maximale Anzahl von Gittern in der Kette (3.3) durch die längstmögliche Teilerkette von $\sqrt{\det L' / \det L}$ nach oben beschränkt. Der kleinstmögliche Teiler ist jeweils 2, also erhalten wir für t eine obere Schranke von $\log_2 \sqrt{\det L' / \det L}$.

Nach Satz 3.11 gilt für L'

$$\sqrt{\det L'} \leqslant \frac{n!}{2^n} \operatorname{vol} B_n \cdot \lambda_1(L') \cdot \ldots \cdot \lambda_n(L') \leqslant \frac{n!}{2^n} \operatorname{vol} B_n B^n$$

und für L

$$\sqrt{\det L} \geqslant \frac{1}{2^n} \operatorname{vol} B_n \cdot \lambda_1(L) \cdot \ldots \cdot \lambda_n(L) \geqslant \frac{1}{2^n} \operatorname{vol} B_n M^n.$$

Daraus folgt

$$s_1 = [L : L'] = \sqrt{\frac{\det L'}{\det L}} \leqslant n! \left(\frac{B}{M}\right)^n.$$

Zu ii) Wir nehmen ohne Beschränkung der Allgemeinheit an, daß $\operatorname{rank} L_{i+1} - \operatorname{rank} L_i \leqslant 1$ für alle $1 \leqslant i < n$ ist. Ansonsten verfeinern wir die Kette und fügen neue Gitter hinzu, bis sich zwei in der Kette benachbarte Gitter jeweils im Rang um höchstens 1 unterscheiden. Dann zeigen wir die Behauptung für die verfeinerte, längere Kette.

Nun werden die Gitter umnumeriert, um sie mit ihrem Rang zu indizieren. Die Kette bestehe nun aus Gittern $L_{ij} \subseteq L, 1 \leqslant j \leqslant m_i$ vom Rank i, erzeugt von Vektoren nicht länger als $B \in \mathbb{R}$, so daß

$$L_{01} \subset L_{11} \subset L_{12} \subset \cdots \subset L_{n,m_n-1} \subset L_{nm_n} = L. \tag{3.4}$$

Für $1 \leqslant i \leqslant n$ sei $z_i \in L_{i1} \setminus \langle L_{i-1,m_{i-1}} \rangle_{\mathbb{R}}$ mit $||z_i|| \leqslant B$. Ein solches z_i existiert, weil alle Gitter L_{ij} aus Vektoren nicht länger als B erzeugt werden. Wir setzen $F = \langle z_1, \cdots, z_n \rangle_{\mathbb{Z}}$. Die Menge $\{z_1, \cdots, z_n\}$ ist nach Konstruktion $\mathbb{R}-$linear unabhängig, also ist $\operatorname{rank} F = n$. Nun addieren wir F zu jedem Gitter in der Kette (3.4) hinzu. Damit erhalten wir die folgende Kette von Gittern, die alle den Rang n besitzen.

$$(F =) L_{01} + F \subseteq \ldots \subseteq L_{ij} + F \subseteq \ldots \subseteq L_{nm_n} + F (= L). \tag{3.5}$$

Man verifiziert leicht, daß $L_{ij} + F \neq L_{i,j+1} + F$, $j < r_i$ für alle $1 \leqslant i \leqslant n$ gilt. Dann kann in der Kette (3.5) Gleichheit nur an den Positionen $L_{i-1,r_{i-1}} + F \subseteq L_{i1} + F, 1 \leq i \leq n$, also höchstens n-mal gelten. Wir wenden die in Teil i) gezeigte Aussage auf die Kette der verschiedenen Gitter in (3.5) an und haben damit gezeigt, daß die Kette (3.4) und damit auch die Kette in Behauptung ii) nicht mehr als $n + \log_2(n!(\frac{B}{M})^n)$ Gitter enthält. □

3.4.1.2 Ein Überdeckungssatz für Erzeugendensysteme

Wir fassen diese Ergebnisse in einem „Überdeckungssatz“ zusammen, der die Existenz von kleinen Erzeugendensystemen garantiert.

Definition 3.13. *Ein Erzeugendensystem S eines Gitters heißt* minimal, *wenn für alle S' gilt*

$$S' \subset S \Rightarrow \langle S' \rangle_{\mathbb{Z}} \subset \langle S \rangle_{\mathbb{Z}}.$$

Satz 3.14. *Sei L ein Gitter mit Minimum M auf dem n-dimensionalen euklidischen Raum E. Sei S ein minimales Erzeugendensystem von L mit Vektoren, die nicht länger als $B \in \mathbb{R}$ sind. Dann ist*

$$|S| \leqslant n + \log_2(n!(\frac{B}{M})^n).$$

Beweis. Sei $S = \{s_1, \cdots, s_t\}$. Wegen der Minimalität von S gilt $\langle s_1, \cdots, s_i \rangle_{\mathbb{Z}} \neq \langle s_1, \cdots, s_{i+1} \rangle_{\mathbb{Z}}$ für alle $1 \leqslant i < t$. Die Behauptung folgt aus der Anwendung von Satz 3.12 ii) auf die Kette

$$\langle s_1 \rangle_{\mathbb{Z}} \subset \langle s_1, \cdots, s_i \rangle_{\mathbb{Z}} \subset \langle s_1, \cdots, s_t \rangle_{\mathbb{Z}} = \langle S \rangle_{\mathbb{Z}}.$$

□

3.4.1.3 Laufzeitanalyse von Algorithmus 2

Eine asymptotische Schranke für die Anzahl der „teuren“ Updateoperationen in Algorithmus 2, bei denen die Basis von $\tilde{L}$ und die Matrix A, wie auf Seite 36 beschrieben, neu berechnet werden müssen, erhalten wir in

Korollar 3.15. *Die Anzahl der Updateoperationen in Algorithmus 2 für die Zerlegung eines Gitters L ist in $O(n \log \frac{nB}{M})$.*

Beweis. In Algorithmus 2 werden sukzessive Zerlegungen von Teilgittern $L_1 \oplus \ldots \oplus L_k \subseteq L$ konstruiert. Diese Teilgitter erfüllen die Voraussetzungen von Satz 3.12 ii). Damit ist die Anzahl der ergänzenden Vektoren, die eine Updateoperation notwendig machen, nicht größer als $n + \log_2(n!(\frac{B}{M})^n)$. Auf diese Schranke wenden wir die Stirling'sche Formel $n! \approx \sqrt{2\pi n}(\frac{n}{e})^n$ an und erhalten eine obere Abschätzung, die in $O(n \log \frac{nB}{M})$ liegt. □

Ein wichtiger Schritt ist die Konstruktion des Gitters $\tilde{L}$ im Algorithmus 2, Zeile 13. Wir benötigen hierfür ein Verfahren, das aus einem Erzeugendensystem von höchstens $n + 1$ Vektoren eine Basis von $\tilde{L}$ effizient berechnet. Weiterhin sollten diese Basisvektoren in irgendeinem Sinn reduziert, d.h. heißt möglichst kurz sein. Ohne

diese Reduktion kann es bei der Iteration von Basisberechnungen zu einer Koeffizientenexplosion kommen, welche die Laufzeit der späteren Basisberechnungen außerordentlich verschlechtert. Dieses Verhalten ist beispielsweise bei der Berechnung der Hermite-Normalform problematisch (siehe das Beispiel von Hafner und McCurley in [HM89]). Wir ziehen daher die Verwendung einer Variante des LLL-Algorithmus vor, der bereits während der Basisberechnung reduziert. Es sind exakte Schranken für die Länge von LLL-reduzierten Basisvektoren in Beziehung zu den sukzessiven Minima des Gitters bekannt. Für die maximale Laufzeit des LLL-Algorithmus ist bei ganzzahligen Gittern eine Abschätzung bekannt.

Lemma 3.16. *Sei L ein ganzzahliges Gitter auf dem n-dimensionalen euklidischen Raum E, welches von $n+1$ Vektoren, alle kürzer als $B \in \mathbb{R}$, erzeugt wird. Dann kann eine (LLL-reduzierte) Basis $\{b_1, \cdots, b_n\}$ von L in $O(n^4 \log B)$ arithmetischen Operationen berechnet werden und es gilt für $1 \leqslant j \leqslant n$, daß*

$$||b_j|| \leqslant 2^{(n-1)/2} \cdot \lambda_j(L) \leqslant 2^{(n-1)/2} \cdot \lambda_n(L). \tag{3.6}$$

Beweis. Nach [BP89], Theorem 3.1. läßt sich eine LLL-reduzierte Basis in $O(2n+2)^4 \log B \subseteq O(n^4 \log B)$ arithmetischen Operationen berechnen. Die Länge der Basisvektoren ist nach [LLL82], Proposition 1.12 wie behauptet beschränkt. □

Nach diesen Vorbereitungen können wir die Laufzeit von Algorithmus 2 nach oben abschätzen.

Satz 3.17. *Sei L ein ganzzahliges Gitter auf dem n-dimensionalen euklidischen Raum $E, B \in \mathbb{R}$, so daß $S := \{v_1, \cdots, v_s\} = \{v \in V \setminus 0 \mid ||v|| \leqslant B\}$ das Gitter L erzeugt. Dann können wir eine orthogonale Zerlegung $L = L_1 \oplus \ldots \oplus L_r$ von L in unzerlegbare Teilgitter $L_i, 1 \leqslant i \leqslant r$ in höchstens $O(n^6 \log^2(nB) + sn^2)$ arithmetischen Operationen berechnen.*

Beweis. Wir analysieren die Laufzeit von Algorithmus 2. Dazu benutzen wir die in Abschnitt 3.4.1 beschriebenen Datenstrukturen.

Mittels Radix-Sort ([Knu74]) sortieren wir alle Vektoren in S ihrer Norm nach und fügen sie in eine Liste ein. Die Norm berechnen wir in $O(n)$ Operationen, so daß der gesamte Sortiervorgang in $O(sn)$ Operationen erfolgen kann.

Seien nun in Zeile 7 bei Eintritt in die Schleife $v_{11}, \cdots, v_{kr_k}, w_1, \cdots, w_{n-l}$ und A bereits per Induktion berechnet. In Zeile 8 wird ein $v \in S$ mit minimaler Norm entnommen und geprüft, ob $v \in L_1 \oplus \ldots \oplus L_k$, d.h., ob die ersten $n-l$ Koeffizienten von Av in $\mathbb{Z}$ liegen und die restlichen l Koeffizienten gleich 0 sind. Dies kostet für alle $v \in S$ zusammengenommen $O(sn^2)$ Operationen. Falls diese Bedingung erfüllt ist, dann liegt v in dem bereits erzeugten Untergitter und es wird kein Update der Datenstrukturen benötigt. Nach Satz 3.12 ii) werden für ganz S in keinem Fall mehr

als $\frac{1}{2}\log(n!(\frac{B}{M})^n) + n$ Updatevorgänge benötigt. Bei jedem Update wird eine LLL-reduzierte Gitterbasis aus einem Erzeugendensystem mit höchstens $n+1$ Vektoren berechnet. Diese Vektoren sind entweder aus S oder Output eines früheren LLLs, also LLL-reduziert. Damit ist ihre Norm durch $2^{(n-1)/2}\lambda_n(L) \leqslant 2^{(n-1)/2}B$ nach oben beschränkt. Nach Lemma 3.16 können wir eine Basis des erzeugten Gitters in höchstens $O(n^4 \log(2^{(n-1)/2}B)) = O(n^5 + n^4 \log B)$ Operationen berechnen. Die Matrix A können wir beispielsweise mit dem Gaußverfahren (siehe [GVL96]) in $O(n^3)$ Operationen updaten.

In der Summe erhalten wir damit für die Gesamtlaufzeit eine obere Schranke von $O(n^6 \log(nB) + n^5 \log(nB) \log B + sn^2)$ arithmetischen Operationen. Daraus folgt die leicht schlechtere, aber in der Schreibweise übersichtlichere Schranke von $O(n^6 \log^2(nB) + sn^2)$ arithmetischen Operationen. □

Bemerkung 3.18. Conway und Sloane konstruierten 1995 ein Gitter, das aus seinen Minimalvektoren erzeugt wird, aber keine Basis aus Minimalvektoren besitzt [CS95]. Damit ist gezeigt, daß es nicht möglich ist, das Anwachsen der Normen der Basisvektoren in der neuen Basis gegenüber dem ursprünglichen Erzeugendensystem zu verhindern. Dennoch ist die Abschätzung über die Länge von LLL-reduzierten Vektoren in Gleichung (3.6) äußerst pessimistisch. In der Praxis findet der LLL-Algorithmus deutlich kürzere Basisvektoren, insbesondere wenn das Erzeugendensystem bereits aus relativ kurzen Vektoren besteht.

3.5 Gitterbasen aus großen Erzeugendensystemen

Ein in der Praxis wichtigeres und viel häufigeres Problem als die Zerlegung von Gittern ist die Berechnung einer Gitterbasis aus einem großen Erzeugendensystem. Hierzu stellen wir die bekanntesten Algorithmen samt einiger Modifikationen zusammen. Im folgenden sei L ein ganzzahliges Gitter mit Minimum M im n-dimensionalen euklidischen Raum E, das von s Vektoren erzeugt wird, die alle nicht länger als $B \in \mathbb{R}$ sind. Weiter nehmen wir an, daß $s > n$ gilt:

3.5.1 Der BP-LLL

Die schon in Lemma 3.16 angesprochene Modifikation des LLL-Algorithmus von Buchmann und Pohst ([BP89]), im folgenden BP-LLL genannt, wird in der Praxis selten eingesetzt. Für theoretische Überlegungen ist dieses Verfahren wertvoll, weil dafür eine obere Abschätzung der Laufzeit von $O((s+n)^4 \log B) \subseteq O(s^4 \log B)$ arithmetischen Operationen existiert.

3.5.2 Der MLLL

Die von Pohst in [Poh87] beschriebene Modifikation des LLL ist unseres Wissens der meistverwendete Algorithmus zur Basisbestimmung. Leider ist die theoretische Laufzeit des MLLL gegenwärtig nicht bekannt. In der Praxis ist er jedoch in seinem Laufzeitverhalten für kleine Erzeugendensysteme (d. h. $s \approx n$) dem LLL ähnlich.

3.5.3 Der serielle BP-LLL

Anstatt den BP-LLL auf das gesamte Erzeugendensystem mit s Vektoren anzuwenden, benutzen wir ihn sukzessiv für Erzeugendensysteme bestehend aus $n+1$ Vektoren. Die Länge der Vektoren in den dabei konstruierten Basen lassen sich mit Lemma 3.16 durch $2^{(n-1)/2}\lambda_n(L) \leqslant 2^{(n-1)/2}B$ nach oben abschätzen. Damit ist die Laufzeit des seriellen BP-LLL in $O(s(2n+2)^4 \log(2^{(n-1)/2}B)) \subseteq O(sn^4(n+\log B))$.

3.5.4 Der serielle BP-LLL mit Aussieben

Wir können den Algorithmus 2 vereinfachen, um lediglich eine Gitterbasis statt einer Zerlegung des Gitters zu berechnen. Der daraus entststehende Algorithmus 3 (s. S. 42) benutzt eine Unterfunktion `construct_basis`, die eine Basis aus höchstens $n+1$ Erzeugern berechnet. Um die Laufzeit mit Satz 3.19 abzuschätzen, benutzen wir hierfür den BP-LLL; man beachte aber auch hier die Bemerkung in 3.5.2.

Input: Ein Erzeugendensystem $S = \{v_1, \cdots, v_s\}$ von L mit $||v_i|| \leqslant B$.
Output: Eine Basis $\{b_1, \cdots, b_n\}$ von L.
1. //Initialisierung.
Wähle $v \in S$, $S \leftarrow S\backslash\{v\}$, $b_1 \leftarrow v$
2. //Füge Vektoren v aus S sukzessive hinzu.
while $S \neq \emptyset$ **do**
  Wähle $v \in S$ und setze $S \leftarrow S\backslash\{v\}$.
  if $v \notin \langle b_1, \cdots, b_k\rangle_{\mathbb{Z}}$ **then**
    $\{b_1, \cdots, b_k\} \leftarrow$ `construct_basis`$(b_1, \cdots, b_k, v)$
  end if
end while

Algorithmus 3: Berechnung der Basis von L aus einem Erzeugendensystem S

Satz 3.19. *Sei L ein ganzzahliges Gitter auf dem n-dimensionalen euklidischen Raum E, das von s Vektoren, alle nicht länger als $B \in \mathbb{R}$, erzeugt wird. Dann können wir eine Basis von L in höchstens $O(n^6 \log^2(nB) + sn^2)$ arithmetischen Operationen berechnen.*

Beweis. Wir benutzen für die Formulierung von Algorithmus 3 dieselben Datenstrukturen wie sie in Abschnitt 3.4.1 beschrieben sind und verwenden den BP-LLL für die Unterfunktion `construct_basis`. Dann berechnet der folgende Algorithmus 3 eine (LLL-reduzierte) Basis von L. Die Abschätzung der Laufzeit erfolgt vollständig analog zum Beweis von Satz 3.17. □

Bemerkung 3.20. Im Vergleich zum seriellen BP-LLL werden in Algorithmus 3 für große s die meisten LLLs durch eine einzige Matrix-Vektor-Multiplikation ersetzt. Dafür werden die im Vergleich zu s relativ wenige Updates mit der Berechnung der Inversen einer $n \times n$-Matrix verteuert.

3.6 Beispiele und Beobachtungen

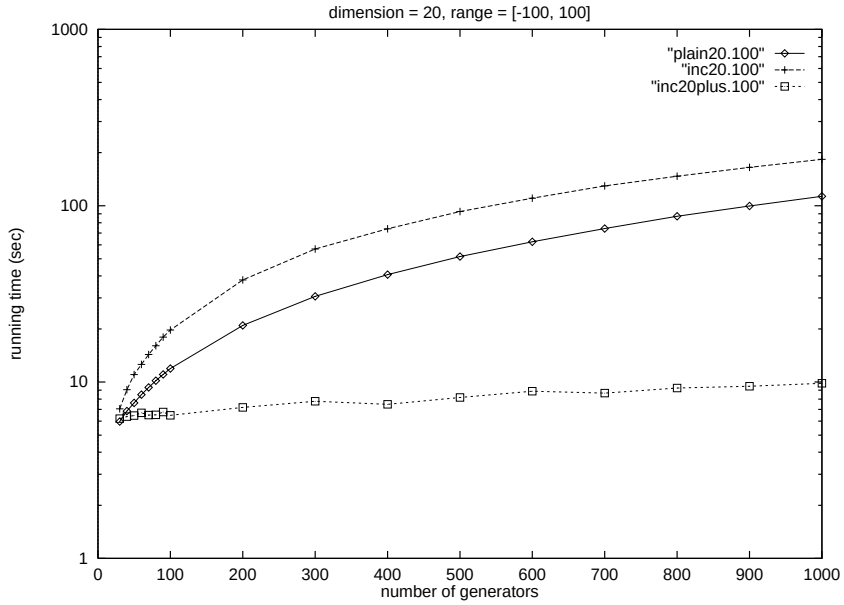
Beispiel 3.21. Sei $E = \mathbb{R}^1$ und $\{p_1, \cdots, p_k\}$ eine Menge von paarweise verschiedenen Primzahlen, deren Produkt gleich t ist. Dann folgt mit dem chinesischen Restesatz $\langle \frac{1}{p_1}, \cdots, \frac{1}{p_k} \rangle_{\mathbb{Z}} = \frac{1}{t}\mathbb{Z}$. In dem Erzeugendensystem liegt kein Vektor im Erzeugnis der anderen. Das Erzeugendensystem ist also „groß“ und minimal, aber das daraus erzeugte Gitter besitzt das im Vergleich zu den p_i relativ kleine Minimum $\frac{1}{t}$.

Bei diesem Beispiel handelt es sich um einen atypischen Extremfall. Viele Gitter, die in der Theorie der quadratischen Formen behandelt werden, etwa ganzzahlige Gitter, Gitter mit hoher Packungsdichte oder durch algebraische Codes konstruierte Gitter, besitzen eher große Minima. In disem Fall wird der Überdeckungssatz 3.14 praktisch relevant und besagt, daß minimale Erzeugendensysteme solcher Gitter relativ klein sind.

Im folgenden untersuchen wir die praktische Anwendbarkeit von Satz 3.14 und Algorithmus 3. Als Kern von Algorithmus 3 benutzen wir in unseren Versuchen den MLLL von Pohst für ganzzahlige Gitter. Bei nicht ganzzahligen Gittern verkompliziert sich die Situation lediglich bei der Vorbereitung des Inputs für unseren Algorithmus. Ist das Gitter zwar nicht ganzzahlig, aber rational, so können wir es durch geeignetes Skalieren auf ein ganzzahliges Gitter transformieren. Handelt es sich um ein echt reelles Gitter, so werden wir zunächst eine Approximation auf ein rationales Gitter vornehmen. Dieses Verfahren wird beispielsweise in [BP89] benutzt und seine Auswirkungen auf den LLL analysiert. Um die Problematik der mit einer Approximation verbundenen Rundungsfehler hier zu vermeiden, betrachten wir nur ganzzahlige Gitter.

Lenstra, Lenstra und Lovász beweisen in [LLL82] für die Laufzeit des LLLs eine asymptotische obere Schranke von $O(n^4 \log B)$ arithmetischen Operationen. Experimente zeigen, daß diese Schranke pessimistisch ist; dies gilt ebenfalls für die oben erwähnten Modifikationen des LLL wie den MLLL. Bei sehr vielen Inputvektoren wird

jedoch auch der MLLL merkbar langsam. Die naive Idee, sukzessive den MLLL auf $n+1$ Vektoren anzuwenden, führt zu keiner Verbesserung. Wir nennen diese Variante den *inkrementellen MLLL*. Die Abbildungen 3.3 und 3.4 zeigen die Laufzeiten vom MLLL, dem inkrementellen MLLL und Algorithmus 3 im Vergleich. Wir wenden die Algorithmen auf Erzeugendensysteme mit Vektoren an, deren Koeffizienten ganzzahlig, pseudo-zufällig und vom Betrag kleiner als eine Konstante K sind. Insbesondere ist die Norm der Vektoren durch $B = \sqrt{n}K$ beschränkt.

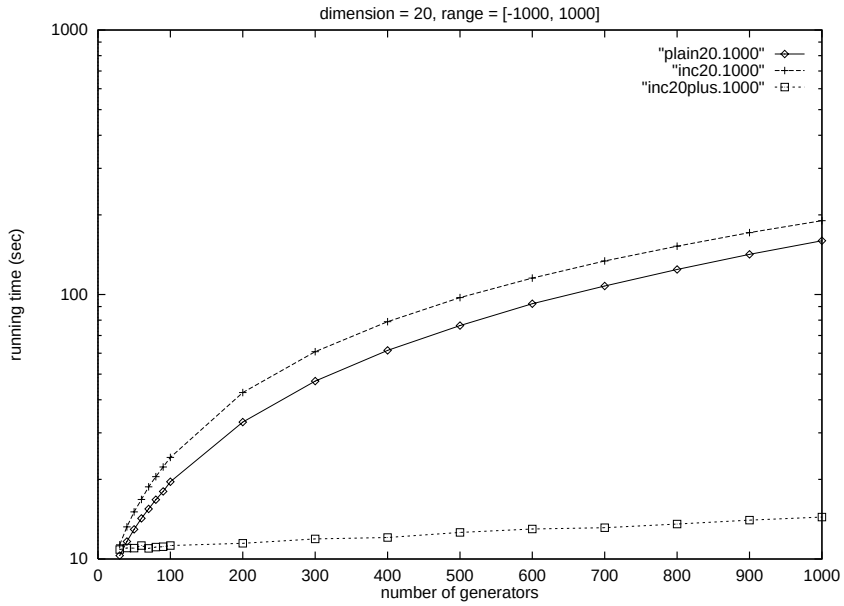


◊ MLLL, + incremental MLLL, □ Algorithm 3

Abbildung 3.3: Laufzeiten in Dimension 20 mit $K = 100$

Die Auswertung der Messungen in den Abbildungen 3.3 und 3.4 zeigt, daß die Anzahl der Updateoperationen in Algorithmus 3 sehr klein in Beziehung zur Größe des Erzeugendensystems ist. Typische Determinanten der Anfangsgitter von vollem Rang sind in der Größenordnung von 2^{100}, die Determinanten der erzeugten Gitter sind sehr klein. Die Determinanten der Anfangsgitter enthalten oft sehr große Primteiler, so daß die Anzahl der Updateoperationen sehr viel kleiner ist als die in Korollar 3.15 angegebene Schranke, bei der die vorkommenden Primteiler nach unten auf 2 abgeschätzt werden.

Die Abbildungen 3.3 und 3.4 stammen von Frank Vallentin, der auch das Testpro-



◊ MLLL, + incremental MLLL, □ Algorithm 3

Abbildung 3.4: Laufzeiten in Dimension 20 mit $K = 1000$

gramm entwickelt hat. Dabei wurde als Kern Victor Shoups NTL[6] Bibliothek benutzt, aus welcher die rationale Arithmetik und der MLLL stammen.

3.7 Implementierungshinweise

Die oben beschriebenen Verfahren führen, wie an den Beispielen gezeigt, zu in der Praxis effizienten Implementierungen mit beweisbaren Komplexitätsabschätzungen. Dennoch sind mit einfachen Heuristiken und Methoden in vielen Fällen weitere erhebliche Laufzeitgewinne zu erzielen. Die Liste der folgenden Hinweise sollte nur als Wegweiser zu einer Implementation von schnellen und robusten Bibliotheksfunktionen dienen.

3.7.1 Das Zerlegungsproblem

Beim Zerlegungsproblem ist eine Basis oder Grammatrix des Gitters bereits bekannt. Daraus kann man unmittelbar die Invarianten Rang und Determinante des Gitters sehr

[6] URL: http://www.shoup.net/ntl/

leicht berechnen. Diese können als Abbruchkriterium im Schleifendurchlauf von Algorithmus 2 verwendet werden.

i) Wenn im Anschluß an Zeile 15 in Algorithmus 2 das Gitter L_1 denselben Rang wie das Gitter L besitzt, dann kann es nur ein triviales orthogonales Komplement besitzen, also ist bereits auch das Gitter L unzerlegbar und der Algorithmus kann beendet werden.

ii) Wenn im Anschluß an Zeile 15 in Algorithmus 2 die Determinante des Gitters $L_1 \oplus \ldots \oplus L_K$ gleich der Determinate von L ist, dann gilt $L_1 \oplus \ldots \oplus L_K = L$, und wir haben eine Zerlegung von L gefunden, wenn auch noch nicht alle Vektoren in S abgearbeitet sind.

Ist ein Gitter zerlegbar, so kann diese Eigenschaft gelegentlich schon an einer LLL-reduzierten Grammatrix des Gitters abgelesen werden. Wenn beispielsweise alle sukzessiven Minima nahe beieinander liegen, dann sind zerlegbare Vektoren relativ lang im Vergleich zu unzerlegbaren Vektoren. Der LLL-Algorithmus wird dann mit großer Wahrscheinlichkeit bereits eine Basis aus unzerlegbaren Vektoren finden. Die zu dieser Basis gehörige Grammatrix besitzt dann (bis auf eine Permutation der Basisvektoren) die Gestalt einer Blockmatrix.

3.7.2 Die Basisberechnung aus einem großen Erzeugendensystem

Der Algorithmus 3 ist leicht parallelisierbar. Eine Erzeugendenmenge kann in kleinere, etwa gleich große Teilmengen zerlegt werden. Die Basen der aus den Teilmengen erzeugten Gitter können jeweils mit dem Algorithmus 3 nach dem „divide and conquer“-Verfahren parallel berechnet werden. In einem letzten Schritt werden die Basisvektoren der Teilgitter wieder zu einem Erzeugendensystem S zusammengefügt, welches wiederum als Input für Algorithmus 3 dient. Diese Vorgehensweise empfiehlt sich allerdings nur für sehr große Erzeugendensysteme. In der Regel ist selbst eine häufige Ausführung der Lookup-Funktion in Zeile 8 schneller als der zusätzliche Aufwand durch wenige neue MLLLs.

4 Der Quantizer der Voronoïzelle eines Gitters

Quadratische Formen und ihre Gitter finden in einem Gebiet der Informationstheorie wichtige Anwendungen. Bei der Übertragung von Informationen über ein störungsanfälliges Medium ist es von eminenter Bedeutung, auf der Empfängerseite die gesendeten Informationen korrekt und vollständig wieder herstellen zu können. Die grundsätzliche Idee zur Fehlerkorrektur besteht in der Erhöhung der Redundanz der Informationen, um kleinere Fehlübertragungen korrigieren zu können. Eine solche Möglichkeit ist die Verwendung fehlerkorrigierender Codes. Viele dieser Codes mit guten Korrektureigenschaften können aus Gittern konstruiert werden. Diese Methoden greifen aber erst, wenn die Informationen bereits als codierte, digitale Daten vorliegen. Bei der Übertragung durch Kabel oder Funkwellen erreichen die Informationen den Empfänger zunächst analog und in der Regel gestört. Ein Analog-Digital-Wandler setzt die analogen Signale in digitale um. Schon in diesem Schritt werden kleinere Störungen korrigiert. Hierfür werden die gesendeten Daten nur aus einem kleinen, fest definierten Zeichenvorrat, dem sogenannten Code Book, gebildet. Ein empfangenes, gestörtes Datum wird mittels eines Quantizers, manchmal auch Approximation oder Compressor genannt, auf das „nächstliegende“ Zeichen im Code Book abgebildet. Bereits diese informale Beschreibung läßt die Beziehung zwischen Quantizern und Voronoïzellen erahnen. Eine Methode, Quantizer mit guten Korrektureigenschaften zu konstruieren, basiert auf Gittern im euklidischen Raum.

Definition 4.1. *Sei P eine diskrete Punktmenge in E. Eine Funktion $q : E \to P$ heißt Quantizer, wenn für jedes $v \in E$ gilt, daß $||v - q(v)|| \leqslant ||x - p||$ für alle $p \in P$.*

Ein Quantizer ordnet also jedem Punkt im euklidischen Raum den nächstliegenden Punkt oder, falls dieser nicht eindeutig ist, einen der nächstliegenden Punkte in P zu.

Ein Quantizer ist also im wesentlichen bereits durch die Angabe seiner Wertemenge charakterisiert. Lediglich für Punkte in E, die sich genau gleich weit von zwei Punkten in P befinden, ist der Funktionswert für zwei Quantizer von P nicht notwendigerweise gleich.

Man kann sich die Funktion eines Quantizers als hochdimensionalen Rundungsprozeß oder auch als eine verlustbehaftete Datenkompression vorstellen. Für die Qua-

lität eines Quantizers können je nach Anwendung verschiedene Maße dienen. Ein universelles und weit verbreitetes ist der durchschnittliche quadratische Fehler je Dimension

$$G_q = \frac{1}{n} \int_E ||v - q(v)||^2 \cdot p(v)\, dv, \tag{4.1}$$

wobei die Funktion p die Wahrscheinschlichkeitsdichte der Datenquelle beschreibt ([CS92], S. 58).

Wenn man für P die Menge der Punkte eines Gitters L wählt, so läßt sich die Konstruktion von gitterbasierten Quantizern elegant mit dem geometrischen Konzept der Voronoïzelle eines Gitters beschreiben. Im folgenden Abschnitt werden einige wichtige Definitionen und Eigenschaften der Voronoïzellen von Gittern zusammengefaßt.

Alle Gitter L in diesem Kapitel haben vollen Rang n, d.h., sie spannen E als $\mathbb{R}$-Vektorraum auf.

4.1 Die Voronoïzelle eines Gitters

Zunächst wird der zentrale Begriff des Voronoïbereichs eines Punktes eingeführt.

Definition 4.2. *Sei P eine diskrete Teilmenge von E. Für ein $v \in P$ heißt die Menge*

$$V_v(P) = \{x \in E \mid ||x - v|| \leqslant ||x - w|| \text{ für alle } w \in P\}$$

der Voronoïbereich von v in P.

Ist $v \in L$ ein Gitterpunkt, so nennt man $V_v(L)$ die Voronoïzelle von v in L. *Wenn keine Verwechselung möglich ist, dann schreibt man auch kurz V_v.*

Offensichtlich gilt $q^{-1}(v) \subseteq V_v(P)$ für ein $v \in P$. Im allgemeinen gilt jedoch keine Gleichheit, weil es Punkte $x \in E$ gibt, die sich in mehr als einer Voronoïzelle befinden.

Die Punkte in E, die sich am weitesten von Gitterpunkten entfernt befinden, kennzeichnen eine wichtige Invariante des Gitters, den Überdeckungsradius.

Definition 4.3. *Für eine Teilmenge $P \subseteq E$ heißt die Zahl*

$$R(P) = \sup_{x \in E} \inf_{v \in P} ||x - v|| \in \mathbb{R}^{\geqslant 0} \cup \{\infty\}$$

der Überdeckungsradius *von P.*

R(L) ist der größtmögliche Abstand, den ein Punkt in E von einem Gitterpunkt haben kann. Das Auffinden von Punkten mit dieser Eigenschaft, den sogenannten *tiefen Löchern* ist hilfreich zur Konstruktion von Gittern mit hoher Packungsdichte (s. [CS92] Chapt. 6).

In diesem Kapitel werden einige geometrischen Eigenschaften von Voronoïzellen vorgestellt.

4.2 Die Voronoïzelle als Polytop

Das erste Ziel ist es, die Voronoïzelle V_0 des Ursprungs 0 eines Gitters als Polytop zu beschreiben und ihre Facetten zu bestimmen.

In diesem Kapitel tritt der Aspekt der definierenden quadratischen Form in den Hintergrund. Zur Vereinfachung wird hier lediglich das euklidische Skalarprodukt (v, v) benutzt, das wegen der zahlreichen Klammern in einigen Formeln auch als $v \cdot v$ geschrieben wird. Falls $v \in E \setminus 0$, schreiben wir $H(v) = \{x \in E \,|\, x \cdot v = v \cdot v\}$ für die Hyperebene mit dem Normalenvektor v und $H_v = \{x \in E \,|\, x \cdot v \leqslant v \cdot v\}$ für den dadurch bestimmten abgeschlossenen Halbraum, der die 0 enthält.

Wenn P eine diskrete Menge mit $0 \in P$ ist, dann kann man in dieser Notation den Voronoïbereich von 0 als Durchschnitt von Halbräumen beschreiben.

Satz 4.4. *Sei P eine diskrete Menge und $0 \in P$. Dann ist*

$$V_0 = \bigcap_{v \in P \setminus \{0\}} H_{\frac{1}{2}v}.$$

Beweis. Sei $0 \neq v \in P$ und $x \in E$. Dann gilt

$$\begin{aligned} & ||x - 0|| \leqslant ||x - v|| \\ \iff \quad & x \cdot x \leqslant (x - v) \cdot (x - v) = x \cdot x - 2x \cdot v + v \cdot v \\ \iff \quad & x \cdot v \leqslant \tfrac{1}{2} v \cdot v \\ \iff \quad & x \cdot \tfrac{1}{2} v \leqslant \tfrac{1}{2} v \cdot \tfrac{1}{2} v \\ \iff \quad & x \in H_{\frac{1}{2}v}. \end{aligned}$$

Damit folgt

$$x \in V_0 \quad \iff \quad \forall_{0 \neq v \in P} ||x - 0|| \leqslant ||x - v|| \quad \iff \quad x \in \bigcap_{0 \neq v \in P} H_{\frac{1}{2}v}.$$

□

Im Fall, daß P ein Gitter ist, folgt aus der Definition 4.2 und der Eigenschaft $P = P - v$ für alle $v \in P$ unmittelbar das

Lemma 4.5. *Sei L ein Gitter auf E.*

i) Für $v \in L$ gilt $V_v = v + V_0$.

ii) Für $\sigma \in O(L)$ gilt $V_0 \sigma = V_0$, inbesondere ist $-V_0 = V_0$.

Die Voronoïzelle eines beliebigen Gitterpunktes ist ein Translat von V_0. Deshalb schreiben wir auch „die“ Voronoïzelle von L, wenn wir nicht eine ausgezeichnete Zelle meinen.

Lemma 4.6. *Es seien $v, w \in V_w$, $x \in V_w$ und $v \in L \setminus \{w\}$ mit $||x-w|| = ||x-v||$, dann ist auch $x \in V_v$.*

Beweis. Sei $z \in L \setminus \{v, w\}$. Aus $x \in V_w$ folgt, daß $||z-x|| \geqslant ||x-w|| = ||x-v||$. Also ist $x \in V_v$. □

Bemerkung 4.7. Aus diesem Lemma folgt u.a. der wichtige Satz, daß $\mathcal{T} = \{V_v \mid v \in L\}$ eine Seite-an-Seite Pflasterung des Raumes E ist.

Lemma 4.8. *Sei $C \subseteq E$ konvex und abgeschlossen. Die Menge C ist genau dann beschränkt, wenn sie keinen Strahl enthält.*

Beweis. Wenn C einen Strahl enthält, dann ist C offenbar nicht beschränkt. Sei C nun nicht beschränkt und ohne Beschränkung der Allgemeinheit sei die Sphäre mit Radius 1 um den Nullpunkt $S_0(1) = \{x \in E \,|\, ||x|| = 1\} \subseteq C$. Da die Menge C nicht beschränkt ist, enthält sie eine Folge $(x_n)_n \subseteq C$ mit $||x_n|| > n$. Wegen der Kompaktheit von $S_0(1)$ enthält die Folge $(\frac{x_n}{||x_n||})_n \subseteq S_0(1)$ nach dem Satz von Bolzano-Weierstraß eine konvergente Teilfolge $(\frac{x_{n_k}}{||x_{n_k}||})_{n_k}$, deren Limes wir mit x bezeichnen. Wir zeigen nun, daß C den Strahl R^+x enthält. Sei $\alpha \in R^+$. Für fast alle k ist $x_{n_k} > \alpha$ und wegen der Konvexität von C dann auch $\frac{\alpha}{||x_{n_k}||} x_{n_k} \in [0, x_{n_k}] \subseteq C \cap S_0(\alpha)$. Aus der Abgeschlossenheit von C folgt, daß $\lim_k \frac{\alpha}{||x_{n_k}||} x_{n_k} = \alpha x \in C$ und damit die Behauptung. □

Lemma 4.9. *$V_0(L)$ ist beschränkt.*

Beweis. Seien $w, x \in E$ mit $w + \alpha x \in V_0$ für alle $\alpha \in \mathbb{R}^{\geqslant 0}$ und $S_x = \{v \in L \mid x \cdot v \geqslant 0\}$. Für alle $\alpha \in \mathbb{R}^{\geqslant 0}$ und $v \in S_X$ gilt, daß $w + \alpha x \in V_0 \Leftrightarrow (w + \alpha x) \cdot \frac{1}{2}v \leqslant \frac{1}{2}v \cdot \frac{1}{2}v$. Damit folgt $x \cdot v = 0$. L ist ein Gitter auf E und somit ist $\dim\langle S_x\rangle_{\mathbb{R}} = \dim E$ und folglich $x = 0$. V_0 ist konvex und enthält keinen Strahl, womit ist gezeigt, daß V_0 beschränkt ist. □

4.3 Voronoïrelevante Vektoren

Definition 4.10. *Sei $v \in L \setminus 0$.*

i) Wir nennen v einen Hüllenvektor, *wenn $H(\frac{1}{2}v) \cap V_0 \neq \emptyset$, also seine Hyperebene die Voronoïzelle berührt.*

ii) v heißt voronoïrelevant *oder kurz* relevant, *wenn $H(\frac{1}{2}v)$ einen $n-1$-dimensionalen (affinen) Schnitt mit V_0 besitzt, d. h., $H(\frac{1}{2}v)$ eine Wand von V_0 enthält [CS92]. Ist v relevant, so schreiben wir F_v für die Wand $H(\frac{1}{2}v) \cap V_0$.*

Die Menge der voronoïrelevanten Vektoren genügt, um V_0 zu konstruieren.

Satz 4.11. *Die Wände F_v von V_0 sind zentralsymmetrisch mit Mittelpunkt $\frac{1}{2}v$.*

Beweis. Sei $t \in F_v$. Wie in 4.5 gezeigt, liegt $-t+v$ in V_v und wegen $(-t+v)\cdot\frac{1}{2}v = \frac{1}{2}v\cdot\frac{1}{2}v$ auch in $H(\frac{1}{2}v)$, also mit Lemma 4.6 in $V_0 \cap H(\frac{1}{2}v) = F_v$. Damit ist auch die Konvexkombination $\frac{1}{2}t + \frac{1}{2}(-t+v) = \frac{1}{2}v$ in F_v und F_v ist zentralsymmetrisch bezüglich $\frac{1}{2}v$. □

Wir wissen nun, daß V_0 als Polytop ein endlicher Durchschnitt von Halbräumen ist. Bereits 1908 hat Voronoï eine Kennzeichnung dieser Halbräume angegeben:

Satz 4.12 ([Vor08]). *Ein Vektor $0 \neq v \in L$ ist genau dann relevant, wenn $\pm v$ die einzigen kürzesten Vektoren in der Nebenklasse $v + L/2L$ sind.*

Beweis. Wegen der Wichtigkeit dieses Satzes und der Kürze des Argumentes geben wir hier den Beweis aus [CS92], S. 475[1] an.

Es seien $v, w \in L$ mit $v - w \in 2L, v \neq \pm w$ und $w\cdot w \leqslant v\cdot v$. Dann sind $t = \frac{1}{2}(v+w)$ und $u = \frac{1}{2}(v-w)$ beide in $L\setminus 0$ und man verifiziert $x\cdot v = x\cdot(t+u) \leqslant \frac{1}{2}(t\cdot t + u\cdot u) \leqslant \frac{1}{4}(v\cdot v + w\cdot w) \leqslant \frac{1}{2}v\cdot v$. Damit ist v nicht relevant, weil $x \in H_t \cap H_u \Rightarrow x \in H_v$.

Sei nun $v \in L$ nicht relevant und einer der kürzesten Vektoren in der Nebenklasse $v + 2L$. Dann gibt es ein $w \in L\setminus 0$ mit $w \neq v$ und $\frac{1}{2}w\cdot\frac{1}{2}w \leqslant \frac{1}{2}v\cdot\frac{1}{2}w$. Wir erhalten

$$(v-2w)\cdot(v-2w) = v\cdot v - 4\,v\cdot w + 4\,w\cdot w \leqslant v\cdot v$$

und haben mit $v-2w$ wegen $0 \neq v-2w \neq \pm v$ einen weiteren kürzesten Vektor in $v+2L$ gefunden. □

An dieser Stelle sei nochmals auf das Korollar 3.6 auf Seite 31 verwiesen, das genau die spitz unzerlegbaren Vektoren als voronoïrelevante identifiziert.

Korollar 4.13. *Für ein Gitter L gilt:*

i) Es gibt höchstens $2\cdot(2^n-1)$ relevante Vektoren.

ii) $V_0(L)$ ist endlicher Durchschnitt von Halbräumen, also ein Polytop, insbesondere konvex.

iii) $R(L) < \infty$.

Bemerkung 4.14. Bereits Voronoï zeigte, daß die Schranke von $2\cdot(2^n-1)$ relevanten Vektoren scharf ist [Vor08, Vor09].

[1] In der ersten Auflage fehlt wegen eines Druckfehlers die Formulierung dieses Satzes. In der zweiten Auflage ist der Satz enthalten, aber der Korrekturhinweis im Vorwort, S. xxviii, verweist auf die falsche Seite.

4.4 Der Quantizer eines Gitters

Im Falle einer Datenquelle mit uniform verteiltem Output und einem Gitterquantizer vereinfacht sich die Gleichung (4.1). Dies motiviert die folgende

Definition 4.15 ([CS92], Chapt. 2.3, Chapt. 21.). *Für ein Gitter L von vollem Rang, heißt*

$$G(L) := G_{V_0} = \frac{1}{n}(\det L)^{-\frac{n+2}{2n}} \int_{V_0} x \cdot x \, dx \quad , \tag{4.2}$$

das normalisierte dimensionslose zweite Trägheitsmoment *von L.*

Mathematisch exakt läßt sich diese Invariante von L im allgemeinen nur schwer errechnen. In [CS92], Chapt. 21 findet sich die Herleitung für einige wichtige Standardpolytope, unter ihnen auch das Simplex. Letzteres wird dort benutzt, um mit Hilfe der Operation der Weyl-Gruppe den Quantizer $G(L)$ für alle Wurzelgitter L zu berechnen.

Für Gitter mit weniger Strukturinformationen ist es praktikabel, das Trägheitsmoment in Gleichung (4.2) durch numerische Monte-Carlo-Integration zu approximieren. Es ist jedoch nicht unmittelbar möglich, eine uniforme Verteilung von Samples in V_0 zu erzeugen, weil die genaue Struktur von V_0 im allgemeinen nicht bekannt ist. Man kann sich hier die Translationseigenschaft von Voronoïzellen zunutze machen. Man definiert eine Abbildung $V_0 \to [0,1)^n$ folgendermaßen. Für eine feste Basis B von L bestimme man für jedes $y \in V_0$ den Koordinatenvektor $u = B^{-1}$ bezüglich B und bestimme komponentenweise den nicht ganzzahligen Anteil $x = (x_1, \cdots, x_n) = (u_1 - \lfloor u_1 \rfloor, \cdots, u_n - \lfloor u_n \rfloor)$. Mit Hilfe der Translationseigenschaft von Voronoïzellen verifiziert man leicht, daß diese Abbildung bijektiv ist. Somit kann man mit einer uniformen Datenquelle in $[0,1)^n$ eine uniforme Verteilung in V_0 erzeugen.[2] Die Transformation mit B muß im Integral in Gleichung (4.2) durch eine entsprechende Substitution berücksichtigt werden.

Im folgenden Algorithmus 4 bezeichnet $\overline{S}$ das arithmetische Mittel der Abstände der Samples zu Gitterpunkten, $G(L)$ das approximierte Trägheitsmoment, $s^2 = \frac{1}{t-1}\sum_k (S[k] - \overline{S})^2$ die Streuung und $\hat{\sigma} = \frac{s}{\sqrt{n}}$ den Schätzwert des Fehlers. Für den korrekten Rückgabewert müssen $\hat{\sigma}$ und $G(L)$ jeweils mit $\frac{1}{n}(\det L)^{-\frac{1}{n}}$ skaliert werden.

Algorithmus 4 kann leicht verallgemeinert werden, um den mittleren quadratischen Fehler für Quantizer von beliebigen Vereinigungen von Nebenklassen eines Gitters $L_C \bigcup_{c \in C} L + c$ für $C = \{c_1, \cdots, c_m\} \subset E$ zu berechnen. Hierzu bestimmt man nicht den Abstand von $B \cdot x_k$ zu L, sondern jeweils den zu $L + c, c \in C$ und setzt $S[k]$ auf das Minimum aller berechneten Abstände. Für die Bestimmung von $G(L_C)$

[2] Die zugrundeliegende geometrische Idee wird in [CS82b, CS82a] beschrieben.

1: *Input:* Ein Gitter $L \subseteq E$, gegeben durch eine Basis B und $t \in \mathbb{N}$, die Anzahl der zu berechnenden Samples.
2: *Output:* $G(L)$ und $\hat{\sigma}$, der Schätzwert des Fehlers von $G(L)$.
3: *// 1. Initialisierung.*
4: Initialisiere ein Array $S[0...k]$ für die zu berechnenden Samples.
5: *// 2. Berechnung der Samples.*
6: **for** $k = 0$ to t **do**
7: Wähle $x_k \in [0,1)^n$ randomisiert.
8: $y_k \leftarrow \text{closest\_vector}(L, B \cdot x_k)$
9: $S[k] \leftarrow ||y_k - B \cdot x_k||^2$
10: **end for**
11: *// 3. Ergebnisaufbereitung.*
12: $\overline{S} \leftarrow \frac{1}{n} \sum_i^t S[i]$. *// Arithmetisches Mittel über die* $S[k]$
13: $\sigma \leftarrow \sqrt{\frac{1}{t-1} \sum_i^t (\overline{S} - S[i])^2}$
14: $\hat{\sigma} \leftarrow \frac{1}{n} (\det L)^{-\frac{1}{n}} \cdot \frac{\sigma}{\sqrt{t}}$
15: $G(L) \leftarrow \frac{1}{n} (\det L)^{-\frac{1}{n}} \cdot \overline{S}$
16: return $(G(L), \hat{\sigma})$

Algorithmus 4: Numerische Approximation von $G(L)$

und des entsprechenden Schätzwertes des Fehler muß dem Umstand Rechnung getragen werden, daß sich die Dichte der Gitterpunkte um den Faktor m gegenüber der Dichte von L vervielfacht. Dies wird durch Korrekturfaktor $\frac{1}{|C|^2}$ bei der Determinante berücksichtigt. Dem liegt die Identität $(\sum_{c \in C} \text{vol}\, V_c)^2 = \det L$ zugrunde, insbesondere ist es nicht notwendig, die exakten Volumina $V_{c_1}, \cdots, V_{c_m}$ zu kennen. Somit erhält man Algorithmus 5.

4.5 Gitter mit kleinem Quantizer

Das dimensionslose zweite normalisierte Trägheitsmoment eines Quantizers $G(L)$ ist eine Invariante des Gitters L, die bis vor kurzem wenig untersucht wurde. Aus der Definition von $G(L)$ in Gleichung (4.2) folgt unmittelbar, daß für ein festes Volumen von $V_0(L)$ das Trägheitsmoment minimal ist, wenn $\int_{V_0} x \cdot x \, dx$ minimal für alle Gitter L ist. Nach dem Isoperimeterprinzip besäßen n-dimensionale Kugeln das kleinste Trägheitsmoment. Kugeln sind jedoch nicht raumfüllend wie Voronoïzellen. Zur Zeit hat man kein anschauliches Verständnis von den Parametern, die ein niedriges Trägheitsmoment bestimmen. In einer neueren Arbeit erzielten Agrell und Eriksson [AE98] eine Reihe von zum Teil kontraintuitiven Ergebnissen über Gitterquantizer.

1: *Input:* Ein Gitter $L \subseteq E$, gegeben durch eine Basis B, eine Menge $C = \{c_1, \cdots, c_m\} \subset E$ und $t \in \mathbb{N}$, die Anzahl der zu berechnenden Samples.
2: *Output:* $G(L)$ und $\hat{\sigma}$, der Schätzwert des Fehlers von $G(L)$
3: *// 1. Initialisierung.*
4: Initialisiere ein Array $S[0...k]$ für die zu berechnenden Samples.
5: *// 2. Berechnung der Samples.*
6: **for** $k = 0$ to t **do**
7: Wähle $x_k \in [0,1)^n$ randomisiert.
8: $y_k \leftarrow \text{closest\_vector}(L, B \cdot x_k)$
9: $S[k] \leftarrow ||y_k - B \cdot x_k||^2$
10: **for** $j = 1$ to m **do**
11: $y_k \leftarrow \text{closest\_vector}(L, B \cdot x_k - c_j)$
12: $S[k] \leftarrow \min(S[k], ||y_k - B \cdot x_k + c_j||^2)$
13: **end for**
14: **end for**
15: *// 3. Ergebnisaufbereitung.*
16: $\overline{S} \leftarrow \frac{1}{n} \sum_i^t S[i]$. *// Arithmetisches Mittel über die* $S[k]$
17: $\sigma \leftarrow \sqrt{\frac{1}{t-1} \sum_i^t (\overline{S} - S[i])^2}$
18: $\hat{\sigma} \leftarrow \frac{1}{n}(\det L/|C|^2)^{-\frac{1}{n}} \cdot \frac{\sigma}{\sqrt{t}}$
19: $G(L_c) \leftarrow \frac{1}{n}(\det L/|C|^2)^{-\frac{1}{n}} \cdot \overline{S}$
20: return $(G(L_c), \hat{\sigma})$

Algorithmus 5: Numerische Approximation von $G(L_C)$

Sie adaptieren dort einen Trainingsalgorithmus für Quantizer auf die Gittersituation. Die mit dieser Methode gefundenen Gitter befinden sich bereits relativ nahe an der von Conway und Sloane vermuteten theoretischen unteren Schranke für den mittleren quadratischen Fehler eines Gitterquantizers [CS85], insbesondere verbessern sie die bekannten Rekorde in den Dimensionen 9 und 10, s. [CS92], S. 61. In Dimension 10 fanden sie in D_{10}^+ ein Gitter, dessen sehr niedriges Trägheitsmoment sie mit 0.070814 ± 0.000001 angeben. Der Umstand, daß ein relativ bekanntes Gitter wie D_{10}^+ in dieser Hinsicht bisher immer übersehen wurde, legt nahe, daß Gitterquantizer bis dahin nur oberflächlich untersucht wurden, ganz im Gegensatz zu den klassischen Problemen wie dem Packungs- und dem Überdeckungsproblem. In Dimension 9 be-

rechneten sie ein nichtklassisches Gitter mit der folgenden Erzeugermatrix

$$\begin{pmatrix} -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 \\ 1/2 & 1/2 & 1/2 & 1/2 & 1/2 & 1/2 & 1/2 & 1/2 & 0.573 \end{pmatrix}$$

und einem Trägheitsmoment von 0.071626 ± 0.0000002 (nach [AE98]), welches sich schon relativ nahe am von Conway und Sloane vermuteten theoretisch möglichen Minimum $0.070902...$ befindet. Die ersten acht Zeilen erzeugen das Gitter D_8. Die ersten acht Koordinaten des neunten Vektors beschreiben ein „tiefes Loch“[3] von D_8. Fügt man das deep hole als Erzeuger zu D_8 hinzu, so erhält man das Gitter E_8. Die neunte Koordinate mit dem seltsam anmutenden, numerisch approximierten Wert 0.573 läßt sich geometrisch nicht einordnen. Das Minimum von D_8 ist gleich 2, das Gitter von Agrell und Eriksson besitzt Minimum $(2 \cdot 0.573)^2 = 1.313316$. Die Voronoïzelle dieses Gitters ist also relativ flach, während man nach der obigen Motivation eher eine kugelförmigere Voronoïzelle erwartet hätte.

Mit `tn` kann man in verschiedenen Dimensionen eine Reihe von Gittern finden mit fast so niedrigen Trägheitsmomenten wie die niedrigsten bekanntesten. Auch in höheren Dimensionen, als sie von Agrell und Eriksson veröffentlicht wurden, findet man mit `tn` Gitter mit Trägheitsmomenten in der Nähe der vermuteten theoretischen unteren Schranke, wie zum Beispiel $G(A_{11}^{+3}) \approx 0.070422 \pm 0.000012$.

[3] Siehe S. 48.

Literaturverzeichnis

[AE98] E. Agrell and T. Eriksson, *Optimization of lattices for quantizers*, IEEE Trans. Inform. Theory **44** (1998), no. 5, 1814–1828.

[AS98] M. Aigner and V.A. Schmidt, *Interview with Yuri I. Manin: Good proofs are proofs that make us wiser*, The Berlin Intelligencer (1998), 16–21, Special edition of Springer-Verlag and Mitteilungen der Deutschen Mathematiker-Vereinigung at the ICM1998.

[Bor92] R.E. Borcherds, *The* 24*-dimensional odd unimodular lattices*, in *Sphere Packings, Lattices and Groups* [CS92].

[BP89] J. Buchmann and M. Pohst, *Computing a lattice basis from a system of generating vectors*, EUROCAL '87 (Leipzig, 1987), Lecture Notes in Comput. Sci., vol. 378, Springer, Berlin, 1989, pp. 54–63.

[BW59] E.S. Barnes and G.E. Wall, *Some extrem forms defined in terms of Abelian groups*, Journal of the Australian Math. Soc. **1** (1959), 47–63.

[CCN+85] J.H. Conway, R.T. Curtis, S.P. Norton, R.A. Parker, and R.A. Wilson, *Atlas of finite groups. Maximal subgroups and ordinary characters for simple groups. With comput. assist. from J. G. Thackray.*, Oxford: Clarendon Press., 1985.

[Coh96] H. Cohen, *A course in computational algebraic number theory*, 3. ed., Graduate Texts in Mathematics, vol. 138, Springer Verlag, 1996.

[CS82a] J.H. Conway and N.J.A. Sloane, *Fast quantizing and decoding algorithms for lattice quantizers and codes*, IEEE Trans. Inform. Theory **28** (1982), no. 2, 227–232.

[CS82b] ———, *Voronoi regions of lattices, second moments of polytopes, and quantization*, IEEE Transaction Inf. Science **28** (1982), no. 2, 211–226.

[CS85] ———, *A lower bound of the average error of vector quantizers*, IEEE Trans. Inform. Theory **31** (1985), no. 1, 106–109.

[CS88] ———, *Low-dimensional lattices. IV. The mass formula*, Proc. R. Soc. London A **419** (1988), 259–286.

[CS92] ———, *Sphere packings, lattices and groups*, 2nd ed., Grundlehren der mathematischen Wissenschaften, vol. 290, Springer, 1992.

[CS95] ———, *A lattice without a basis of minimal vectors*, Mathematika **42** (1995), no. 1, 175–177.

[Eic52] M. Eichler, *Note zur Theorie der Kristallgitter*, Mathematische Annalen **125** (1952), 51–55.

[FP85] U. Fincke and M. Pohst, *Improved methods for calculating vectors of short length in a lattice, including a complexity analysis*, Math. Com. **44** (1985), 463–471.

[GVL96] G.H. Golub and C.F. Van Loan, *Matrix computations*, third ed., Johns Hopkins Studies in the Mathematical Sciences, Johns Hopkins University Press, Baltimore, MD, 1996.

[HM89] J. Hafner and K. McCurley, *Asymptotically fast triangularization of matrices over rings*, SIAM J. Comp. **20** (1989), 1068–1083.

[HV98] B. Hemkemeier and F. Vallentin, *On the decompositon of lattices*, Tech. Report TR98-52, Electronic Colloquium on Computation and Complexity, 1998, Available at `http://www.eccc.uni-trier.de/eccc/`.

[Kne54] M. Kneser, *Zur Theorie der Kristallgitter*, Mathematische Annalen **127** (1954), 105–106.

[Kne56] ———, *Klassenzahlen indefiniter quadratischer Formen*, Arch. Math. **7** (1956), 323–332.

[Kne57] ———, *Klassenzahlen definiter quadratischer Formen*, Arch. Math. **8** (1957), 241–250.

[Knu74] D.E. Knuth, *The art of computer programming.*, Addison-Wesley Series in Computer Science and Information Processing, vol. 3: Sorting and searching., Addison-Wesley Publishing Company, 1974.

[KR88] B.W. Kernighan and D. Ritchie, *C programming language*, 2nd ed., Prentice Hall PTR, 1988.

[LLL82] A.K. Lenstra, Jr. Lenstra, H.W., and L. Lovász, *Factoring polynomials with rational coefficients*, Math. Ann. **261** (1982), no. 4, 515–534.

[Min96] H. Minkowski, *Geometrie der Zahlen*, Teubner, Leipzig, 1896.

[Mis96] M. Mischler, *Un lien les $\mathbb{Z}$-réseaux unimodulaires et les formes hermitiennes les F-réseaux*, Thèse de doctorat, Faculté de Sciences de Université de Franche-Comté et de Université de Lausanne, 1996.

[Nie73] H.-V. Niemeier, *Definite quadratische Formen der Dimension 24 und Diskriminate* 1, J. Number Theory **5** (1973), 142–178.

[NV96] G. Nebe and B. B. Venkov, *Nonexistence of extremal lattices in certain genera of modular lattices*, J. Number Theory **60** (1996), no. 2, 310–317.

[O'M71] O.T. O'Meara, *Introduction into quadratic forms*, Springer, Berlin, 1971.

[O'M80] ———, *On indecomposable quadratic forms*, J. Reine Angew. Math. **317** (1980), 120–156.

[Poh87] M. Pohst, *A modification of the LLL reduction algorithm*, J. Symbolic Comput. **4** (1987), no. 1, 123–127.

[PS97] W. Plesken and B. Souvignier, *Computing isometries of lattices*, Journal Symb. Comp. **24** (1997), no. 3/4, 327–334.

[Que95] H.-G. Quebbemann, *Modular lattices in euclidian spaces*, Journal of Number Theory **54** (1995), no. 2, 190–202.

[SB96] R. Scharlau and B. Blaschke, *Reflective integral lattices*, J. Algebra **181** (1996), 934–961.

[Sch98] A. Schiemann, *Classification of hermitian forms with the eighbour method*, To appear in Journal of Symbolic Computation, March 1998.

[SH98] R. Scharlau and B. Hemkemeier, *Classification of integral lattices with large class number*, Math. Comp. **67** (1998), no. 222, 737–749.

[SSP99] Rudolf Scharlau and Rainer Schulze-Pillot, *Extremal lattices.*, Matzat, B. Heinrich (ed.) et al., Algorithmic algebra and number theory. Selected papers from a conference, Heidelberg, Germany, October 1997. Berlin: Springer. 139-170 , 1999 (English).

[SV94] R. Scharlau and B.B. Venkov, *The genus of the Barnes-Wall lattice*, Comment. Math. Helv. **69** (1994), no. 2, 322–333.

[SV95] R. Scharlau and B. Venkov, *The genus of the Coxeter-Todd-Lattice*, Preprint, 1995.

[SV00] ———, *Classifying lattices using modular forms — a preliminary report*, Codes, Lattices, Modular Forms and Vertex Operator Algebras (M. Ozeki and M. Harada E. Bannai, eds.), Yamagata University, October 2000.

[TvL84] R.E. Tarjan and J. van Leeuwen, *Worst–case analysis of set union algorithms*, Journal of the ACM **31** (1984), no. 2, 245–281.

[Ven92] B.B. Venkov, *Even unimodular* 24*-dimensional lattices*, in *Sphere Packings, Lattices and Groups* [CS92].

[Vor08] G. F. Voronoï, *Nouvelles applications des paramètres continus à la theorie des formes quadratiques*, Journal für reine und angewandte Mathematik **134** (1908), 198–287.

[Vor09] ———, *Nouvelles applications des paramètres continus à la theorie des formes quadratiques*, Journal für reine und angewandte Mathematik **136** (1909), 67–181.