Aufzählung von kurzen Gittervektoren in allgemeiner Norm

•

Dissertation
zur Erlangung des Doktorgrades
der Naturwissenschaften

•

vorgelegt beim Fachbereich Mathematik
der Johann Wolfgang Goethe–Universität

in Frankfurt am Main

von

Harald Ritter[1]

aus Aschaffenburg

Frankfurt am Main 1997
(D F 1)

[1]e-mail: ritter@mi.informatik.uni-frankfurt.de

vom Fachbereich Mathematik der
Johann Wolfgang Goethe–Universität
als Dissertation angenommen

Dekan	Prof. Dr. H.F. de Groote
Gutachter	Prof. Dr. C.P. Schnorr Prof. Dr. M. Sieveking
Datum der Disputation	17.7.1997

Zusammenfassung

Unter einem Gitter vestehen wir eine (durch eine Gitterbasis erzeugte) diskrete, additive Untergruppe des $\mathbb{R}^n$. Wir entwickeln Aufzählungsverfahren zur Bestimmung eines (bezüglich einer beliebig vorgegebenen Norm) nachweislich kürzesten, von Null verschiedenen Gittervektors, die in möglichst hohen Dimensionen vertretbare Rechenzeiten besitzen. Wir lösen Rucksackprobleme durch Reduktion auf die Suche nach einem Gittervektor mit l_∞–Norm 1. Mit unserem Aufzählungsverfahren bezüglich der l_∞–Norm gelingt es, *alle* Rucksackprobleme bis Dimension 66 effizient zu lösen. Wir brechen mit diesem Verfahren Kryptosysteme, die auf dem Rucksackproblem basieren.

Der Aufwand zur Bestimmung eines kürzesten, von Null verschiedenen Gittervektors hängt entscheidend von der Wahl der Basis des jeweiligen Gitters ab. Er wird umso geringer, je besser die Basisvektoren die kürzesten Gittervektoren approximieren. Wir untersuchen daher verschiedene Reduktionsbegriffe. Wir zeigen Schranken für die Güte der Approximation und entwickeln Algorithmen zur Bestimmung von reduzierten Gitterbasen.

Für große Dimensionen reichen selbst „gut vorreduzierte" Gitterbasen nicht mehr aus, um einen kürzesten Gittervektor effizient zu bestimmen. Wir entwickeln ein probabilistisches Aufzählungsverfahren, das für zufällige Eingaben eine fest vorgegebene Erfolgswahrscheinlichkeit beweisbar nicht unterschreitet.

Inhaltsverzeichnis

Einleitung

Unter einem *Gitter* verstehen wir allgemein eine (durch eine *Gitterbasis* erzeugte) diskrete, additive Untergruppe des $\mathbb{R}^n$. *Gitterbasenreduktion* bedeutet, eine Basis aus der (unendlichen) Menge der Gitterbasen auszuwählen, die aus möglichst kurzen Vektoren besteht.

Das Hauptziel der vorliegenden Arbeit besteht darin, Verfahren zur Bestimmung eines (bezüglich einer beliebig vorgegebenen Norm) kürzesten, von Null verschiedenen Gittervektors zu entwickeln, die in möglichst hohen Dimensionen vertretbare Rechenzeiten besitzen. Im Gegensatz zu heuristischen Verfahren, die mit möglichst geringer Rechenzeit eine möglichst hohe Trefferquote zu erreichen suchen, gilt es, einen nachweislich kürzesten, von Null verschiedenen Gittervektor zu bestimmen. Ein solcher Nachweis ist für viele Anwendungen unabdingbar. Betrachten wir als Beispiel Rucksackprobleme. Ein Rucksackproblem besteht darin, zu gegebenen natürlichen Zahlen $n, a_1, \ldots, a_n$ und s einen 0–1–Vektor $(x_1, \ldots, x_n)$ zu finden mit $\sum_{i=1}^{n} a_i x_i = s$ oder zu zeigen, daß kein solcher Vektor existiert. Jedes Rucksackproblem läßt sich in natürlicher Weise reduzieren auf die Suche nach einem l_∞–Norm kürzesten, von Null verschiedenen Vektor in einem geeigneten Gitter (siehe Kapitel 6.1). Das Rucksackproblem ist genau dann lösbar, wenn ein Gittervektor existiert mit l_∞–Norm 1. Damit löst ein Verfahren zur Bestimmung eines l_∞–Norm–kürzesten, von Null verschiedenen Gittervektors vollständig das Rucksackproblem. Eine solche vollständige Lösung gelingt nicht durch heuristische Verfahren.

Als Grundlage der Aufzählungsverfahren dient uns das Verfahren ENUM von Schnorr und Euchner [SE94] (siehe auch Kapitel 3.1), das einen kürzesten, von Null verschiedenen Gittervektor bezüglich der euklidischen Norm bestimmt. Zu einer gegebenen Gitterbasis $b_1, \ldots, b_m$ werden rekursiv für $t = m, \ldots, 1$ alle ganzzahligen Linearkombinationen der Vektoren $b_t, \ldots, b_m$ durchlaufen, deren zu $b_1, \ldots, b_{t-1}$ orthogonaler Anteil hinreichend kleine euklidische Norm besitzt. Die geometrischen Eigenschaften der euklidischen Norm erlauben eine besonders effiziente Berechnung der Längen dieser orthogonalen Anteile mit Hilfe der Gram–Schmidt–Orthogonalisierung. Bei der Übertragung des Verfahrens ENUM auf allgemeine Normen ersetzen wir die Längen der orthogonalen Anteile durch die von Lovász und Scarf [LS92] eingeführten Höhenfunktionen. Für feste Gitterbasen $b_1, \ldots, b_m$ mißt die i–te Höhenfunktion den Abstand des Punktes x (bezüglich der gegebenen Norm) zu dem von $b_1, \ldots, b_{i-1}$ aufgespannten Unterraum. Für allgemeine Normen erfordert die Berechnung der i–ten Höhenfunktion die Lösung eines (nichtlinearen) Optimierungsproblems mit $i-1$ reellen Variablen und ist daher aufwendig. Wir nutzen die Äquivalenz der Normen auf $\mathbb{R}^n$, um die explizite Berechnung der Höhenfunktionen für $i > 1$ zu vermeiden. Dabei nehmen wir in Kauf, daß die Anzahl der zu betrachtenden Linearkombinationen wächst, denn der Aufwand pro Linearkombination wird erheblich kleiner. Für l_p–Normen nutzen wir die Höldersche Ungleichung $| <x, y> | \leq \|x\|_p \|y\|_q$, $1/p + 1/q = 1$, um den Aufwand für die Aufzählung nochmals erheblich zu verringern. Für die l_∞– und l_1–Norm beschleunigen wir dadurch das Aufzählungsverfahren um einen exponentiellen Faktor.

Der Aufwand zur Bestimmung eines kürzesten, von Null verschiedenen Gittervektors hängt entscheidend von der gewählten Basis des jeweiligen Gitters ab. Der Aufwand wird umso geringer, je besser die Basisvektoren die kürzesten Gittervektoren approximieren (siehe Kapitel 4). Wir fassen daher in Kapitel 1 und 2 die wichtigsten Reduktionsbegriffe und deren Eigenschaften zusammen. Der Gaußsche Reduktionsbegriff [Gau01] beschränkt sich auf zweidimensionale Gitter. Gauß–reduzierte Gitterbasen lassen sich effizient bestimmen und bestehen stets aus den beiden kürzesten linear unabhängigen Gittervektoren. Die Begriffe der Hermite–Korkine–Zolotareff–reduzierten Gitterbasen [Her50, KZ72, KZ73, KZ77] und der Minkowski–reduzierten Gitterbasen [Min91] sind für beliebige Dimensionen definiert. Bei diesen Basen ist der erste Basisvektor stets ein kürzester von Null verschiedener Gittervektor. Es ist jedoch kein effizienter Algorithmus zur Konstruktion solcher Basen bekannt.

Lenstra [Len83] liefert ein Polynomialzeit-Verfahren zur Bestimmung eines (bezüglich der euklidischen Norm) kürzesten Gittervektors bei fester Dimension. Der Algorithmus von wurde unter anderem von Kannan [Kan87] weiterentwickelt. Just (vormals Helfrich) [Hel85] gibt ein Polynomialzeit–Verfahren zur Minkowski–Reduktion bei fester Dimension an.

Einen Meilenstein in der Entwicklung von effizienten Algorithmen zur Gitterbasenreduktion in beliebigen Dimensionen stellt der L^3–Algorithmus von Lenstra, Lenstra und Lovász [LLL82] dar. Der L^3–Algorithmus liefert nach polynomial (in der Bitlänge der Eingabe) vielen Schritten eine Gitterbasis, deren (bezüglich der euklidischen Norm) kürzester Vektor höchstens um einen (in der Anzahl der Basisvektoren) exponentiellen Faktor größer ist als der kürzeste von Null verschiedene Gittervektor.

Schnorr [Sch87, Sch94] hat für die euklidische Norm die *Blockreduktion* und *blockreduzierte Gitterbasen* eingeführt, um die kürzesten Gittervektoren besser zu approximieren. L^3–reduzierte Gitterbasen sind blockreduziert mit Blockgröße 2, Hermite–Korkine–Zolotareff–reduzierte Gitterbasen der Dimension m sind blockreduziert mit Blockgröße m. Für kleine Blockgrößen ist die Blockreduktion ähnlich effizient wie die L^3–Reduktion. Bei Blockgröße 20 benötigt die Blockreduktion etwa das 10–fache der Laufzeit einer L^3–Reduktion, liefert aber erheblich kürzere Gittervektoren [SH95]. Eine polynomiale Laufzeitschranke konnte allerdings nicht gezeigt werden.

Lovász und Scarf [LS92] übertragen den Begriff der L^3–reduzierten Gitterbasen auf allgemeine Normen und geben einen Algorithmus zur Berechnung solcher Basen an, der polynomial (in der Größe der Eingabe) viele arithmetische Operationen und Berechnungen der Höhenfunktionen benötigt. Gauß– und blockreduzierte Basen wurden bereits von Kaib [Kai94] für allgemeine Normen analysiert. Kaib gibt auch effiziente Algorithmen zur Bestimmung von Gauß–reduzierten Basen an. Wir entwickeln Algorithmen zur Berechnung von blockreduzierten Basen.

Für große Dimensionen reichen selbst „gut vorreduzierte" Gitterbasen nicht aus, um die deterministischen Verfahren zur Aufzählung von kurzen Gittervektoren in akzeptabler Rechenzeit durchführen zu können. Daher interessieren wir uns für probabilistische Aufzählungsverfahren, die mit einer fest vorgegebenen Wahrscheinlichkeit und möglichst geringem Rechenaufwand einen kürzesten, von Null verschiedenen Gittervektor finden. Als Hilfsmittel verwenden wir die Gaußsche Volumenheuristik. Die Anzahl von Vektoren eines Gitters $L \subset \mathbb{R}^n$ in einer geeigneten Teilmenge $S \subset \operatorname{span}(L)$ wird hierbei durch den Quotienten aus dem Volumen von S und der Determinante des Gitters approximiert. Wir geben in Kapitel 5 einen Algorithmus an, der für jede fest vorgegebene Wahrscheinlichkeit p und zufällige Eingaben beweisbar die Erfolgswahrscheinlichkeit nicht unterschreitet. Dies verbessert insofern das geschnittene Aufzählungsverfahren von Schnorr und Hörner [SH95], als dort keine explizite untere Schranke für die Erfolgswahrscheinlichkeit gezeigt wurde.

Ein wichtiges Anwendungsgebiet der Gitterbasenreduktion sind Angriffe auf kryptographische Systeme. Eines der wesentlichen Ziele der Kryptographie besteht darin, zwei Teilnehmern (Alice und Bob) eine sichere Kommunikation über einen unsicheren Kanal zu ermöglichen. Dabei soll ein Angreifer (Eve) nicht in der Lage sein, durch Abhören des Kanals die übermittelten Nachrichten zu verstehen. Bei *Public–Key*–Kryptosystemen konstruiert Bob ein Paar (PK_B, SK_B) von Schlüsseln und veröffentlicht den *Public Key* PK_B, den Alice zum Verschlüsseln der Nachricht benutzt. Der *Secret–Key* SK_B bleibt geheim und wird von Bob zum Entschlüsseln verwendet.

Eines der ersten Public–Key–Kryptosysteme basiert auf dem Rucksackproblem [MH78]. Bei Rucksack–Kryptosystemen besteht der öffentliche Schlüssel im wesentlichen aus einer Menge $a_1, \ldots, a_n$ von „schwierigen" Gewichten. Alice verschlüsselt eine binäre Nachricht $m = m_1 \cdots m_n$, indem sie $s := \sum_{i=1}^n m_i a_i$ berechnet. Bob verwendet seinen geheimen Schlüssel, um das „schwierige" Rucksackproblem $(n, a_1, \ldots, a_n, s)$ in ein „einfaches" Rucksackproblem $(n, a'_1, \ldots, a'_n, s' = \sum_{i=1}^n m_i a'_i)$ zu transformieren und daraus die ursprüngliche Nachricht m zu rekonstruieren. Ein solches „einfaches" Rucksackproblem ist zum Beispiel eine Folge $a'_1, \ldots, a'_n$ mit $a'_1 = 1$ und $a'_i > \sum_{j=1}^{i-1} a'_j$. Für diese Gewichte ist die Lösung m stets in linearer Zeit zu bestimmen.

Der erste Angriff auf das Rucksack–Kryptosystem von Merkle und Hellman [MH78] erfolgte durch Shamir [Sha82]. Brickell [Bri84] gelang es, das Merkle–Hellman–System mit dem

L^3–Algorithmus in polynomialer Zeit vollständig zu brechen. Coster, Joux, LaMacchia, Odlyzko, Schnorr und Stern [CJL$^+$92] transformieren das Rucksackproblem $(n, a_1, \ldots, a_n, s)$ auf die Suche nach einem kurzen Vektor bezüglich der euklidischen Norm in einem geeigneten Gitter. Sie zeigen, daß ein l_2–Norm–kürzester von Null verschiedener Gittervektor „fast immer" eine Lösung des Rucksackproblems liefert, wenn die *Dichte* $d := n / \log_2(\max_i a_i)$ hinreichend klein (< 0.9408) ist. Mit Hilfe von effizienten Algorithmen zur Blockreduktion bezüglich der euklidischen Norm ist es gelungen, hohe Erfolgsquoten bei der Lösung von Rucksackproblemen zu erzielen [SE94]. Schnorr und Hörner [SH95] brechen mit diesen Verfahren das Rucksack–Kryptosystem von Chor und Rivest [CR88].

Mit Hilfe des Aufzählungsverfahrens bezüglich der l_∞–Norm können wir das Rucksack–Entscheidungsproblem zumindest bis zur Dimension $n = 66$ vollständig und effizient lösen (Kapitel 6.1). Zum Brechen des Rucksack–Kryptosystems von Orton [Ort94] (Kapitel 6.2.2) verwenden wir das l_∞–Norm–Aufzählungsverfahren für ein Gitter mit 246 Basisvektoren. Der Algorithmus findet beweisbar immer die entschlüsselte Nachricht, die Gesamtlaufzeit des Angriffs beträgt dabei nur wenige Minuten.

Eine weitere Anwendungsmöglichkeit der Aufzählung bezüglich der l_∞–Norm stellt die Konstruktion von t–Designs dar (Kapitel 6.4). t–Designs sind eine spezielle Klasse von nichtlinearen Codes mit konstantem Gewicht. Ihre Konstruktion ist bereits für kleine Parameter t schwierig. Betten, Kerber, Kohnert, Laue und Wassermann [BKK$^+$95] gelang erstmals die Konstruktion von 7–Designs mit Hilfe von Gitterbasenreduktionsalgorithmen. Mit dem Aufzählungsverfahren bezüglich der l_∞–Norm lassen sich effizient viele weitere 7–Designs konstruieren.

Coppersmith [Cop96a, Cop96b] verwendet Algorithmen zur Gitterbasenreduktion für die Konstruktion kleiner Nullstellen von Polynomen über endlichen Ringen. Coppersmith und Shamir [CS97] analysieren kryptographische Verfahren mittels Gitterbasenreduktion.

Mein Dank gilt insbesondere meinem akademischen Lehrer, Prof. Dr. Claus Peter Schnorr, für die umfassende Ausbildung und viele fruchtbare Diskussionen. Weiterhin möchte ich mich bei Dr. Michael Kaib und Dr. Carsten Rössner für die Zusammenarbeit bei einigen Forschungsprojekten bedanken. Johannes Merkle und Jean–Pierre Seifert möchte ich für viele nützliche Verbesserungsvorschläge zu dieser Arbeit danken.

Kapitel 1

Einführung

Gitter sind diskrete, additive Untergruppen $L \subset \mathbb{R}^n$. Ein linear unabhängiges Erzeugendensystem $b_1, \ldots, b_m$ von L heißt *Basis* des Gitters $L = L(b_1, \ldots, b_m)$. Jedes Gitter besitzt eine Basis. Umgekehrt ist jede von endlich vielen linear unabhängigen Vektoren erzeugte additive Untergruppe des $\mathbb{R}^n$ diskret, also ein Gitter. Die Anzahl der Basisvektoren eines Gitters heißt *Rang* oder *Dimension* des Gitters und ist unabhängig von der Wahl der Basis.

Das Ziel der *Gitterbasenreduktion* besteht darin, mit effizienten Algorithmen eine gegebene Gitterbasis in eine Basis desselben Gitters zu transformieren, deren Vektoren möglichst kleine Norm besitzen. Solche *reduzierte Basen* wurden für die Dimensionen 2 und 3 bereits von Lagrange [Lag73], Gauß [Gau01] und Dirichlet [Dir50] studiert. Die grundlegenden Arbeiten der Gitterbasenreduktion für beliebige Dimensionen stammen von Hermite [Her50], Korkine und Zolotareff [KZ72, KZ73, KZ77], Minkowski [Min91], Lovász et al. [LLL82, LS92] und Schnorr [Sch87]. Ein häufig verwendetes Maß für die Qualität einer reduzierten Basis sind die sukzessiven Minima, die wir im folgenden definieren.

Sei eine beliebige Norm $\|.\|$ auf $\mathbb{R}^n$ gegeben. Wir bezeichnen mit

$$S_n(M, r, \|.\|)) := \{x \in \mathbb{R}^n : \|x - M\| \leq r\}$$

die n–dimensionale $\|.\|$–Kugel mit Mittelpunkt M und Radius r. Für Kugeln bezüglich der euklidischen Norm schreiben wir kurz $S_n(M, r)$. Häufig verzichten wir auch auf den Index, falls die Dimension der Kugel aus dem Zusammenhang heraus klar ist.

Definition 1.1 *Das* i–te sukzessive Minimum $\lambda_{i,\|.\|}(L)$ *des Gitters L (bezüglich der Norm* $\|.\|$*) ist der minimale Radius einer Kugel um 0, die i linear unabhängige Gittervektoren enthält:*

$$\lambda_{i,\|.\|}(L) := \inf\{r \mid \dim(\mathrm{span}(L) \cap S(0, r, \|.\|)) \geq i\}$$

Insbesondere ist $\lambda_{1,\|.\|}(L)$ die kleinste Norm eines von Null verschiedenen Vektors des Gitters L. Ein Gittervektor x mit $\|x\| = \lambda_{1,\|.\|}(L)$ heißt *kürzester Gittervektor*.

Nicht jedes Gitter L besitzt eine Basis $b_1, \ldots, b_m$ mit $\|b_i\| = \lambda_{i,\|.\|}(L)$ für $i = 1, \ldots, m$. Betrachten wir als Beispiel das von den Einheitsvektoren $e_1, \ldots, e_n \in \mathbb{R}^n$ und $b = \frac{1}{2}\sum_{i=1}^n e_i$ erzeugte Gitter. Für jede l_p–Norm und jedes $n > 2^p$ ist $\lambda_{1,\|.\|_p}(L) = \ldots = \lambda_{n,\|.\|_p}(L) = 1$ und die Einheitsvektoren sind (bis auf das Vorzeichen) die einzigen Gittervektoren mit Norm 1. Sie bilden aber keine Basis des Gitters.

1.1 Reduktionsbegriffe

Für Gitter vom Rang 2 liefert die *Gauß–Reduktion* für beliebige Eingabebasen effizient eine Basis, die aus den beiden kürzesten linear unabhängigen Gittervektoren besteht. Eine geordnete Gitterbasis $b_1, b_2 \in \mathbb{R}^n$ heißt *Gauß–reduziert* (bezüglich der Norm $\|.\|$ auf $\mathbb{R}^n$), wenn

$$\|b_1\| \leq \|b_2\| \leq \|b_1 - b_2\| \leq \|b_1 + b_2\|.$$

In einer Iteration des Gaußschen Reduktionsverfahrens wird ein ganzzahliges Vielfaches des Vektors b_1 von b_2 abgezogen, so daß $\|b_2^{\text{neu}}\| = \min_{t \in \mathbb{Z}} \|b_2 - tb_1\|$. Die Vektoren b_1 und b_2^{neu} werden vertauscht, falls $\|b_1\| > \|b_2^{\text{neu}}\|$ gilt. Andernfalls ist b_1, b_2^{neu} oder $b_1, -b_2^{\text{neu}}$ Gauß–reduziert. Bei Eingabe einer Gitterbasis b_1, b_2 benötigt der Algorithmus höchstens $O(\log(B)) + o(1)$ Iterationen. Dabei ist $B := \max(\|b_1\|, \|b_2\|)/\lambda_{2,\|.\|}(L(b_1, b_2))$ [Kai94].

Zur Einführung der Reduktionsbegriffe und –algorithmen bezüglich der euklidischen Norm für Gitter mit höherem Rang benötigen wir den Begriff des *Orthogonalsystems* sowie das Verfahren zur *Gram–Schmidt–Orthogonalisierung*. Zu einer Gitterbasis $b_1, \ldots, b_m$ definieren wir rekursiv das *Orthogonalsystem* $\hat{b}_1, \ldots, \hat{b}_m$ und die zugehörigen *Gram–Schmidt–Koeffizienten* $\mu_{i,j}$ durch

$$\hat{b}_1 := b_1, \quad \mu_{i,j} := \frac{< b_i, \hat{b}_j >}{< \hat{b}_j, \hat{b}_j >}, \quad \hat{b}_i := b_i - \sum_{j=1}^{i-1} \mu_{i,j}\hat{b}_j, \qquad 1 \leq i, j \leq m.$$

Die *orthogonalen Projektionen* auf $\text{span}(b_1, \ldots, b_{i-1})^\perp$ bezeichnen wir mit π_i:

$$\pi_i : \mathbb{R}^n \to \text{span}(b_1, \ldots, b_{i-1})^\perp, \qquad \pi_i(b_j) := \sum_{t=i}^{j} \mu_{j,t}\hat{b}_t, \qquad i = 1, \ldots, m.$$

Eine Gitterbasis b_1, b_2 ist genau dann Gauß–reduziert bezüglich der euklidischen Norm, wenn $\|b_1\|_2 \leq \|b_2\|_2$ und $0 \leq \mu_{2,1} \leq \frac{1}{2}$.

Die Gram–Schmidt–Orthogonalisierung erlaubt auch eine einfache Berechnung der *Determinante* $\det(L)$ eines Gitters $L(b_1, \ldots, b_m)$, die als m–dimensionales Volumen der *Grundmasche* $\{x = \sum_{i=1}^m x_i b_i \mid 0 \leq x_i < 1\}$ definiert wird:

$$\det(L) := \mathrm{vol}_m\Big(\{x = \sum_{i=1}^m x_i b_i \mid 0 \leq x_i < 1\}\Big).$$

Die Determinante eines Gitters ist von der Wahl der Basis unabhängig. Es gilt

$$\det(L) = \prod_{i=1}^m \|\hat{b}_i\|_2. \tag{1.1}$$

Einer der stärksten Reduktionsbegriffe geht auf Hermite [Her50], Korkine und Zolotrareff [KZ72, KZ73, KZ77] zurück. Der erste Vektor einer HKZ–reduzierten Gitterbasis ist stets ein kürzester Gittervektor:

Definition 1.2 *Eine Gitterbasis* $b_1, \ldots, b_m$ *heißt* reduziert im Sinne von Hermite, Korkine und Zolotareff (bezüglich der euklidischen Norm) *oder kurz* HKZ–reduziert*, wenn gilt*

1. $|\mu_{i,j}| \leq \frac{1}{2}$ *für* $1 \leq j < i \leq m$,
2. $\|\hat{b}_i\|_2 = \lambda_{1,\|.\|_2}(L(\pi_i(b_i), \ldots, \pi_i(b_m))$ *für* $1 \leq i \leq m$.

Bedingung 2 der Definition ist äquivalent zu den beiden folgenden Bedingungen:

2'. $\|b_1\|_2 = \lambda_{1,\|.\|_2}(L)$,

2". $\pi_2(b_2), \ldots, \pi_2(b_m)$ ist eine HKZ–reduzierte Basis des Gitters $L(\pi_2(b_2), \ldots, \pi_2(b_m))$.

Durch diese rekursive Form wird die algorithmische Motivation des Reduktionsbegriffs deutlich. HKZ–reduzierte Gitterbasen approximieren sehr gut die sukzessiven Minima, denn es gilt der folgende

Satz 1.3 [LLS90, Mah38] *Für jede HKZ–reduzierte Basis* $b_1, \ldots, b_m$ *eines Gitters* L *gilt*

$$\frac{4}{i+3}\lambda_{i,\|.\|_2}(L)^2 \leq \|b_i\|_2^2 \leq \frac{i+3}{4}\lambda_{i,\|.\|_2}(L)^2.$$

Es ist kein Verfahren bekannt, das aus einer beliebigen Basis eines Gitters eine HKZ–reduzierte Basis desselben Gitters in polynomial (in der Größe der Eingabe) vielen Schritten konstruiert.

Ein entscheidender Durchbruch auf dem Gebiet der Gitterbasenreduktion gelang Lovász mit der Entwicklung des sogenannten L^3–Algorithmus [LLL82]. Der L^3–Algorithmus erweitert in natürlicher Weise das Gauß–Verfahren auf Gitterbasen beliebigen Ranges. Es führt Gauß–Reduktionsschritte auf zwei aufeinanderfolgenden Basisvektoren b_{k-1}, b_k durch, genauer auf den Anteilen dieser Vektoren orthogonal zu $b_1, \ldots, b_{k-2}$.

Definition 1.4 *Eine Gitterbasis* $b_1, \ldots, b_m \in \mathbb{R}^n$ *heißt* L^3–reduziert mit δ ($\delta \in (0,1]$), *wenn gilt*

1. $|\mu_{i,j}| \leq \frac{1}{2}$ *für* $1 \leq j < i \leq m$,
2. $\delta \|\hat{b}_{k-1}\|_2^2 \leq \|\pi_{k-1}(b_k)\|_2^2$ *für* $k = 2, \ldots, m$.

Die Gitterbasis heißt längenreduziert, *wenn die erste Bedingunng erfüllt ist.*

Die Reduktionsbedingungen sind um so stärker, je größer δ ist. Für $\delta > \frac{1}{4}$ approximieren L^3–reduzierte Gitterbasen die sukzessiven Minima bis auf einen (im Rang des Gitters) exponentiellen Faktor. Die Bedingung $\delta > \frac{1}{4}$ stellt sicher, daß die im folgenden verwendete Konstante $\alpha := 1/(\delta - 1/4)$ positiv ist. L^3–reduzierte Gitterbasen sind für $\delta < 1$ in polynomialer Zeit aus einer beliebigen Gitterbasis zu konstruieren.

Satz 1.5 [LLL82] *Für jede mit* $\delta \in (1/4, 1]$ *L^3–reduzierte Basis* $b_1, \ldots, b_m$ *eines Gitters* L *gilt mit* $\alpha := 1/(\delta - 1/4)$ *für* $j = 1, \ldots, m$

$$\begin{aligned} \alpha^{1-j} &\leq \|\hat{b}_j\|_2^2 \lambda_{j,\|\cdot\|_2}^{-2}(L), && (1.2) \\ \|b_j\|_2^2 \lambda_{j,\|\cdot\|_2}^{-2}(L) &\leq \alpha^{m-1}, && (1.3) \\ \|b_k\|_2^2 &\leq \alpha^{j-1} \|\hat{b}_j\|_2^2 \quad \text{für } k \leq j. && (1.4) \end{aligned}$$

Satz 1.6 [SE94] *Bei ganzzahliger Eingabebasis* $b_1, \ldots, b_m \in \mathbb{Z}^n$ *mit* $M = \max_i \|b_i\|_2^2$ *und* $\delta \in (1/4, 1)$ *führt der L^3–Algorithmus mit iterativer Orthogonalisierung (siehe Kapitel 2.2) höchstens* $O(m^2 n(1 + m \log M))$ *arithmetische Schritte durch. Diese Schritte werden auf ganzen Zahlen durchgeführt, die absolut durch* $mM^m(9/(4\delta - 1))^{(m/2)}$ *beschränkt sind.*

1.2 Übertragung der Reduktionsbegriffe auf allgemeine Normen

Bei vielen Anwendungen (siehe Kapitel 6) suchen wir Gittervektoren, die bezüglich einer anderen als der euklidischen Norm kurz sind. Besondere Bedeutung haben hierbei die l_∞– und l_1–Norm. Daher erscheint eine Übertragung der für die euklidische Norm bekannten Reduktionsbegriffe auf diese Normen sinnvoll. Bei HKZ–reduzierten Gitterbasen bezüglich der euklidischen Norm besitzt b_i minimalen euklidischen Abstand von span$(b_1, \ldots, b_{i-1})$ unter allen von Null verschiedenen Vektoren im Gitter $L(b_i, \ldots, b_m)$. Die natürliche Vorgehensweise bei der Übertragung der Reduktionsbegriffe auf andere Normen besteht darin, den Abstand von span$(b_1, \ldots, b_{i-1})$ nicht bezüglich der euklidischen, sondern bezüglich der gewünschten Norm zu minimieren. Zu diesem Zweck verwenden wir die von Lovász und Scarf [LS92] eingeführten Höhenfunktionen.

Definition 1.7 *Seien $b_1, \ldots, b_m \in \mathbb{R}^n$ linear unabhängige Vektoren, $m \leq n$ und $\|.\|$ eine Norm auf $\mathbb{R}^n$. Die Funktionen $F_i : \mathbb{R}^n \to \mathbb{R}$ mit*

$$F_i(x) := \min_{\xi_1,\ldots,\xi_{i-1} \in \mathbb{R}} \left\| x + \sum_{j=1}^{i-1} \xi_j b_j \right\|, \quad 1 \leq i \leq m+1,$$

heißen Höhenfunktionen, *die Funktionswerte $F_i(x)$* i–te Höhen *von x auf* $\mathrm{span}(b_1, \ldots, b_{i-1})$ *(bezüglich der Norm $\|.\|$).*

Die Höhenfunktionen F_i sind Normen auf $\mathrm{span}(b_1, \ldots, b_{i-1})^\perp$, was leicht nachzurechnen ist. Die Höhenfunktionen bezüglich der euklidischen Norm sind selbst wieder euklidische Normen auf den jeweiligen Unterräumen. Die Länge eines kürzesten Gittervektors läßt sich durch die Höhenfunktionen nach unten beschränken.

Satz 1.8 *Für jede Basis $b_1, \ldots, b_m$ eines Gitters $L \subset \mathbb{R}^n$ und jede Norm $\|.\|$ gilt*

$$\min_{i=1,\ldots,m} F_i(b_i) \leq \lambda_{1,\|.\|}(L).$$

Beweis von Satz 1.8:
Sei $b = u_1 b_1 + \ldots + u_k b_k$ die Darstellung eines kürzesten Gittervektors, wobei $u_k \neq 0$. Dann gilt $\lambda_{1,\|.\|}(L) = \|b\| \geq F_k(b) = |u_k| F_k(b_k) \geq F_k(b_k)$. □

Mit Hilfe der Höhenfunktionen können wir nun die Reduktionsbegriffe auf allgemeine Normen übertragen.

Definition 1.9 *Eine Gitterbasis $b_1, \ldots, b_m$ heißt* reduziert im Sinne von Hermite, Korkine und Zolotareff (bezüglich der Norm $\|.\|$) *oder kurz* HKZ–reduziert, *wenn gilt*

1. *$F_j(b_i) \leq F_j(b_i \pm b_j)$ für alle $j < i$,*
2. *$F_i(b_i) = \lambda_{1,F_i}(L(b_i, \ldots, b_m))$ für $1 \leq i \leq m$.*

Bedingung 2 der Definition ist äquivalent zu den beiden folgenden Bedingungen:

2'. $\|b_1\| = \lambda_{1,\|.\|}(L(b_1, \ldots, b_m))$,

2". $b_2, \ldots, b_m$ ist HKZ–reduzierte Basis des Gitters $L(b_2, \ldots, b_m)$ bezüglich der Norm F_2.

Analog zum Fall der euklidischen Norm approximieren HKZ–reduzierte Gitterbasen auch für allgemeine Normen sehr gut die sukzessiven Minima. Insbesondere ist der erste Basisvektor stets ein kürzester Gittervektor.

Satz 1.10 [LS92] *Für jede HKZ–reduzierte Basis $b_1, \ldots, b_m$ eines Gitters L gilt*

$$\frac{2}{i+1} \leq \|b_i\| / \lambda_{i,\|.\|}(L) \leq \frac{i+1}{2} \quad \textit{für } i = 1, \ldots, m.$$

Die bekannten Algorithmen zur HKZ–Reduktion benötigen exponentielle Laufzeit (im Rang des Gitters). Das Entscheidungsproblem, ob ein durch eine Basis gegebenes Gitter einen Vektor mit l_∞–Norm 1 enthält, ist sogar NP–vollständig. (Das Rucksackproblem läßt sich auf dieses Problem reduzieren; siehe Kapitel 6.1.) Damit besteht kaum Hoffnung, Polynomialzeitalgorithmen zur HKZ–Reduktion für allgemeine Normen zu finden. Lovász und Scarf [LS92] übertragen den Begriff der L^3–reduzierten Basen auf allgemeine Norm und geben einen Algorithmus zu deren Berechnung an, der polynomial (in der Größe der Eingabe) viele arithmetische Operationen und Berechnungen der Höhenfunktionen benötigt. Die sukzessiven Minima werden bis auf einen (im Rang des Gitters) exponentiellen Faktor approximiert.

Definition 1.11 *Sei $\|.\|$ eine Norm auf $\mathbb{R}^n$ und $0 < \Delta \leq 1$. Eine Basis $b_1, \ldots, b_m$ eines Gitters $L \subset \mathbb{R}^n$ heißt* Lovász–Scarf–reduziert mit Δ *oder kurz* LS–reduziert mit Δ, *falls gilt*

1. $F_j(b_i) \leq F_j(b_i \pm b_j)$ *für alle* $j < i$,
2. $\Delta F_i(b_i) \leq F_i(b_{i+1})$ *für* $1 \leq i < m$.

Die Gitterbasis heißt längenreduziert, *wenn die erste Bedingung erfüllt ist.*

Für die euklidische Norm sind die Begriffe der mit δ L^3–reduzierten Gitterbasen und der mit $\Delta = \sqrt{\delta}$ LS–reduzierten Gitterbasen identisch. Die Reduktionsbedingungen sind umso stärker, je größer Δ ist. Für $\Delta > \frac{1}{2}$ approximieren LS–reduzierte Gitterbasen die sukzessiven Minima bis auf einen (im Rang des Gitters) exponentiellen Faktor.

Satz 1.12 [LS92] *Für jede mit $\Delta \in (1/2, 1]$ LS–reduzierte Basis $b_1, \ldots, b_m$ eines Gitters L gilt*

$$\left(\Delta - \frac{1}{2}\right)^{i-1} \leq F_i(b_i) / \lambda_{i,\|.\|}(L) \leq \left(\Delta - \frac{1}{2}\right)^{i-m} \quad \textit{für } i = 1, \ldots, m.$$

Satz 1.13 [LS92] *Der Algorithmus zur LS–Reduktion einer Gitterbasis $b_1, \ldots, b_m \in \mathbb{R}^n$ (siehe Kapitel 2.2) terminiert für festes m und $\Delta \in (1/2, 1)$ in polynomial (in der Größe der Eingabe) vielen Schritten.*

Kapitel 2

Blockreduzierte Gitterbasen

HKZ–reduzierte Gitterbasen liefern zwar sehr kurze Gittervektoren, sind aber nicht effizient zu konstruieren. Auf der anderen Seite sind L^3– und LS–reduzierte Gitterbasen zwar in polynomialer Zeit zu bestimmen, approximieren die kürzesten Gittervektoren aber nur bis auf einen exponentiellen Faktor. Schnorr [Sch87] hat daher eine Hierarchie von Reduktionsbegriffen bezüglich der euklidischen Norm entwickelt (die sogenannten *blockreduzierten Basen*), die die Begriffe der L^3– und HKZ–reduzierten Basen als Extremalfälle umfaßt. Wir betrachten blockreduzierte Basen bezüglich allgemeiner Normen. Eine Gitterbasis $b_1, \ldots, b_m$ ist *blockreduziert* mit Blockgröße β, wenn jeder Block $b_j, \ldots, b_{\min(j+\beta-1,m)}$ von höchstens β aufeinanderfolgenden Basisvektoren HKZ–reduziert ist bezüglich der Höhenfunktion F_j. Der Reduktionsbegriff ist umso stärker, je größer β ist. Blockreduzierte Basen mit Blockgröße 2 sind LS–reduziert, blockreduzierte Basen mit $\beta = m$ sind HKZ–reduziert. Für Blockgröße 20 benötigt die Blockreduktion bezüglich der euklidischen Norm etwa das 10–fache der Laufzeit einer L^3–Reduktion, liefert aber erheblich kürzere Gittervektoren [SH95]. Eine polynomiale Laufzeitschranke konnte allerdings nicht gezeigt werden.

Definition 2.1 *Sei $\beta \geq 2$ eine ganze Zahl und $\Delta \in (0,1]$ reell. Eine Basis $b_1, \ldots, b_m$ eines Gitters $L \subset \mathbb{R}^n$ heißt* (β, Δ)–blockreduziert (bezüglich der Norm $\|.\|$), *wenn für $i = 1, \ldots, m$ gilt:*

1. $F_j(b_i) \leq F_j(b_i \pm b_j)$ *für alle $j < i$,*
2. $\Delta F_i(b_i) \leq \lambda_{1,F_i}(L(b_i, \ldots, b_{\min(i+\beta-1,m)}))$.

Die Basis heißt längenreduziert (bezüglich der Norm $\|.\|$) *, wenn die erste Bedingung erfüllt ist.*

Für jede längenreduzierte Basis $b_1, \ldots, b_m$ eines Gitters $L \subset \mathbb{R}^n$ und $1 \leq j < i \leq m$ gilt

$$F_j(b_i) \leq F_i(b_i) + \frac{1}{2} \sum_{t=j}^{i-1} F_t(b_t). \tag{2.1}$$

Beweis von (2.1):
Sei $\xi_0 \in \mathbb{R}$ eine Minimalstelle von $F_t(b_i + \xi b_t)$, d.h. $F_t(b_i + \xi_0 b_t) = \min_{\xi \in \mathbb{R}} F_t(b_i + \xi b_t) = F_{t+1}(b_i)$. Damit gilt

$$\begin{aligned} F_t(b_i) &= \min_{\mu \in \mathbb{Z}} F_t(b_i + \mu b_t) \;\leq\; F_t(b_i + \lceil \xi_0 \rfloor b_t) \;=\; F_t(b_i + \xi_0 b_t + (\lceil \xi_0 \rfloor - \xi_0) b_t) \\ &\leq F_{t+1}(b_i) + \frac{1}{2} F_t(b_t). \end{aligned}$$

Die Behauptung folgt durch sukzessive Anwendung dieser Ungleichung für $t = j, \ldots, i-1$. □

Für die euklidische Norm läßt sich die Blockreduziertheit auch mit Hilfe der Gram–Schmidt–Koeffizienten ausdrücken:

Definition 2.2 *Sei $\beta \geq 2$ eine ganze Zahl und $\delta \in (0, 1]$ reell. Eine Basis $b_1, \ldots, b_m$ eines Gitters $L \subset \mathbb{R}^n$ heißt (β, δ)–blockreduziert (bezüglich der Norm $\|.\|_2$), wenn für $i = 1, \ldots, m$ gilt:*

1. $|\mu_{i,j}| \leq \frac{1}{2}$ *für alle* $j < i$,
2. $\delta \|\hat{b}_i\|_2^2 \leq \lambda_{1,\|.\|_2}^2 (L(\pi_i(b_i), \ldots, \pi_i(b_{\min(i+\beta-1,m)})))$.

Eine Gitterbasis ist genau dann (β, δ)–blockreduziert, wenn sie (β, Δ)–blockreduziert ist (bezüglich der euklidischen Norm) mit $\Delta = \sqrt{\delta}$.

Für bezüglich der euklidischen Norm längenreduzierte Gitterbasen läßt sich Ungleichung (2.1) noch verschärfen zu

$$\|\pi_j(b_i)\|_2^2 \leq \|\hat{b}_i\|_2^2 + \frac{1}{4} \sum_{t=j}^{i-1} \|\hat{b}_t\|_2^2, \tag{2.2}$$

denn es gilt $F_j^2(b_i) = \|\pi_j(b_i)\|_2^2 = \|\hat{b}_i + \sum_{t=j}^{i-1} \mu_{i,t} \hat{b}_t\|_2^2 = \|\hat{b}_i\|_2^2 + \sum_{t=j}^{i-1} \mu_{i,t}^2 \|\hat{b}_t\|_2^2$ und $|\mu_{i,t}| \leq \frac{1}{2}$.

Für $\beta = 2$ und $\delta \geq \frac{1}{3}$ ist der Begriff der $(2, \delta)$–blockreduzierten Basen identisch mit dem Begriff der L^3–reduzierten Basen. Für $\beta = m$ und $\Delta = 1$ (bzw. $\delta = 1$) fällt der Reduktionsbegriff mit dem Begriff der HKZ–reduzierten Basen zusammen. Für Gitter vom Rang $m = 2$ sind die Begriffe der HKZ– sowie der $(2, 1)$–blockreduzierten Basen bis auf die Reihenfolge und das Vorzeichen der Vektoren äquivalent zum Begriff der *Gauß–reduzierten* Basen in der jeweiligen Norm. Der verallgemeinerte Algorithmus zur Gauß–Reduktion in beliebiger Norm wird ausführlich von Kaib [Kai94] behandelt. Kaib überträgt in seiner Arbeit auch die Schranken von Schnorr [Sch94] für die Güte von $(\beta, 1)$–blockreduzierten Gitterbasen von der euklidischen Norm auf beliebige Norm. Bei praktischen Implementationen verwenden wir stets $\delta < 1$ bzw. $\Delta < 1$. Wir benötigen daher entsprechende Schranken für den Fall der (β, δ)– bzw. (β, Δ)–blockreduzierten Gitterbasen, die wir im folgenden beweisen. In den Kapiteln 2.2–2.5 behandeln wir dann die Algorithmen zur Blockreduktion.

2.1 Eigenschaften blockreduzierter Basen

Im Gegensatz zu allgemeinen Normen sind bei der euklidischen Norm alle auftretenden Höhenfunktionen selbst wieder euklidische Normen. Dies ermöglicht den Beweis besserer Schranken für die Normen der reduzierten Basisvektoren. Wir behandeln daher bei der Analyse der blockreduzierten Basen den Fall der euklidischen Norm gesondert.

2.1.1 Der Fall der euklidischen Norm

Schnorr [Sch94] verwendet für die Schranken der Güte von $(\beta, 1)$–blockreduzierten Basen die Hermite–Konstanten. Die *Hermite–Konstante* γ_β ist definiert als das Supremum des Ausdrucks $\lambda^2_{1,\|.\|_2}(L)(\det L)^{-2/\beta}$ über alle Gitter vom Rang β. Die Hermite–Konstanten $\gamma_2, \gamma_3, \gamma_4$ und γ_5 wurden von Korkine und Zolotareff [KZ72, KZ73, KZ77] ermittelt. Blichfeldt [Bli34] hat γ_6, γ_7 und γ_8 bestimmt.

β	1	2	3	4	5	6	7	8
γ_β^β	1	$\frac{4}{3}$	2	4	8	$\frac{2^6}{3}$	2^6	2^8

Tabelle 2.1 Hermite–Konstanten

Für $\beta \geq 9$ sind die Hermite–Konstanten nicht explizit gegeben. Die obere Schranke

$$\gamma_\beta \leq \frac{2}{\pi}\Gamma(2 + \frac{\beta}{2})^{2/\beta} \tag{2.3}$$

wurde von Blichfeldt [Bli14] bewiesen. Wir kennen für $\beta \to \infty$ die asymptotischen Schranken

$$\frac{\beta}{2\pi e}(1 + o(1)) \leq \gamma_\beta \leq \frac{\beta}{1.14\pi e}(1 + o(1)). \tag{2.4}$$

Die untere Schranke geht auf Minkowski und Hlawka zurück. Die obere Schranke stammt von Kabatiansky und Levenshtein [KL78].

(β, δ)–blockreduzierte Gitterbasen $b_1, \ldots, b_m$ approximieren die sukzessiven Minima bis auf einen (in m/β) exponentiellen Faktor. Die Approximation wird mit wachsender Blockgröße und wachsendem δ besser. Ist β ein konstanter Bruchteil von m, so werden die sukzessiven Minima sogar bis auf einen in m polynomialen Faktor approximiert. Schnorr hat das folgende Theorem 2.3 bereits für $\delta = 1$ gezeigt.

Theorem 2.3 *Sei $b_1, \ldots, b_m$ eine (β, δ)–blockreduzierte Basis des Gitters L mit $2 \le \beta \le m$ und $0 < \delta \le 1$. Dann gilt für $i = 1, \ldots, m$*

$$\frac{\|\hat{b}_i\|_2^2}{\lambda^2_{i,\|.\|_2}(L)} \le \frac{1}{\delta}\left(\frac{\gamma_\beta}{\delta}\right)^{2\frac{m-i}{\beta-1}}, \tag{2.5}$$

$$\frac{\|b_i\|_2^2}{\lambda^2_{i,\|.\|_2}(L)} \le \frac{i+3}{4\delta}\left(\frac{\gamma_\beta}{\delta}\right)^{2\frac{m-1}{\beta-1}}, \tag{2.6}$$

$$\frac{\lambda^2_{i,\|.\|_2}(L)}{\|b_i\|_2^2} \le \frac{i+3}{4\delta}\left(\frac{\gamma_\beta}{\delta}\right)^{2\frac{i-1}{\beta-1}}. \tag{2.7}$$

Beim Beweis von Theorem 2.3 lehnen wir uns eng an den Beweis von Schnorr [Sch94] an. Wir zeigen zunächst das folgende

Lemma 2.4 *Sei $b_1, \ldots, b_m$ eine (β, δ)–blockreduzierte Basis des Gitters $L \subset \mathbb{R}^n$ mit $2 \le \beta \le m$ und $0 < \delta \le 1$. Dann gilt mit $M := \max(\|\hat{b}_{m-\beta+2}\|_2, \ldots, \|\hat{b}_m\|_2)$*

$$\|b_1\|_2 \le \left(\frac{\gamma_\beta}{\delta}\right)^{\frac{m-1}{\beta-1}} M.$$

Beweis von Lemma 2.4:
Wir erweitern die Basis $b_1, \ldots, b_m$ durch $\beta - 2$ linear unabhängige Vektoren zu

$$b_{-\beta+3}, \ldots, b_{-1}, b_0, b_1, \ldots, b_m, \tag{2.8}$$

so daß gilt:

1. $\|b_i\|_2 = \|b_1\|_2$ für $i \le 0$,
2. $<b_i, b_j> = 0$ für $i \le 0,\ i \ne j$ und $j = -\beta+3, \ldots, m-\beta+1$.

Hierzu betten wir die gegebene Basis in $\mathbb{R}^{n+\beta-2}$ ein und setzen für $b_{-\beta+3}, \ldots, b_0$ das $\|b_1\|_2$–fache der Einheitsvektoren in den zusätzlichen Richtungen. Damit ist Basis (2.8) (β, δ)–blockreduziert, d.h. für $i = -\beta+3, \ldots, m-\beta+1$ gilt

$$\delta \|\hat{b}_i\|_2^2 \le \lambda^2_{1,\|.\|_2}(L(\pi_i(b_i), \ldots, \pi_i(b_{i+\beta-1}))).$$

Unter Verwendung der Hermite–Konstante γ_β erhalten wir

$$\delta^{\beta/2}\|\hat{b}_i\|_2^\beta \le \gamma_\beta^{\beta/2}\|\hat{b}_i\|_2\|\hat{b}_{i+1}\|_2 \cdots \|\hat{b}_{i+\beta-1}\|_2 \quad \text{für } i = -\beta+3, \ldots, m-\beta+1.$$

Multiplikation dieser $m-1$ Ungleichungen liefert

$$\begin{aligned}&\delta^{(m-1)\beta/2}\|\hat{b}_{-\beta+3}\|_2^{\beta}\|\hat{b}_{-\beta+4}\|_2^{\beta}\cdots\|\hat{b}_{m-\beta+1}\|_2^{\beta}\\ &\qquad\leq \gamma_\beta^{(m-1)\beta/2}\|\hat{b}_{-\beta+3}\|_2^{1}\|\hat{b}_{-\beta+4}\|_2^{2}\cdots\|\hat{b}_1\|_2^{\beta-1}\|\hat{b}_2\|_2^{\beta}\cdots\|\hat{b}_{m-\beta+1}\|_2^{\beta}\\ &\qquad\qquad\cdot\|\hat{b}_{m-\beta+2}\|_2^{\beta-1}\cdots\|\hat{b}_{m-1}\|_2^{2}\|\hat{b}_m\|_2^{1}.\end{aligned}$$

Dies impliziert

$$\begin{aligned}&\delta^{(m-1)\beta/2}\|\hat{b}_{-\beta+3}\|_2^{\beta-1}\|\hat{b}_{-\beta+4}\|_2^{\beta-2}\cdots\|\hat{b}_0\|_2^{2}\|\hat{b}_1\|_2^{1}\\ &\qquad\leq \gamma_\beta^{(m-1)\beta/2}\|\hat{b}_{m-\beta+2}\|_2^{\beta-1}\cdots\|\hat{b}_{m-1}\|_2^{2}\|\hat{b}_m\|_2^{1}.\end{aligned}$$

Wegen $\|b_i\|_2=\|b_1\|_2$ für $i\leq 0$ folgt daraus

$$\delta^{(m-1)\beta/2}\|b_1\|_2^{\binom{\beta}{2}}\leq\gamma_\beta^{(m-1)\beta/2}M^{\binom{\beta}{2}}\quad\text{und damit}\quad\|b_1\|_2\leq\left(\frac{\gamma_\beta}{\delta}\right)^{\frac{m-1}{\beta-1}}M.\qquad\square$$

Beweis von Theorem 2.3:
zu (2.5): Wir beweisen (2.5) zunächst für $i=1$ durch Induktion über m:

Für $m\leq\beta$ gilt $\sqrt{\delta}\|b_1\|_2\leq\lambda_{1,\|.\|_2}(L)$ nach Definition der (β,δ)–Blockreduziertheit. Daraus folgt die Behauptung unmittelbar wegen $\gamma_\beta/\delta\geq 1$.

Sei nun $m>\beta$ und v ein kürzester Gittervektor.
Für $v\in L(b_1,\ldots,b_{m-1})$ können wir m um 1 erniedrigen und die Behauptung folgt unmittelbar aus der Induktionsannahme. Für $v\notin L(b_1,\ldots,b_{m-1})$ gilt

$$\lambda_{1,\|.\|_2}(L)=\|v\|_2\geq\|\pi_j(v)\|_2\geq\lambda_{1,\|.\|_2}(L(\pi_j(b_j),\ldots,\pi_j(b_m))\geq\sqrt{\delta}\|\hat{b}_j\|_2$$

für $j=m-\beta+1,\ldots,m$. Damit ist $\lambda_{1,\|.\|_2}(L)\geq\sqrt{\delta}\max(\|\hat{b}_{m-\beta+2}\|_2,\ldots,\|\hat{b}_m\|_2)$ und die Behauptung folgt aus Lemma 2.4.

Für $i>1$ ist auch $L_i:=L(\pi_i(b_i),\ldots,\pi_i(b_m))$ (β,δ)–blockreduziert und damit

$$\|\hat{b}_i\|_2\leq\frac{1}{\sqrt{\delta}}\left(\frac{\gamma_\beta}{\delta}\right)^{\frac{m-i}{\beta-1}}\lambda_{1,\|.\|_2}(L_i).$$

Wegen $\lambda_{1,\|.\|_2}(L_i)\leq\lambda_{i,\|.\|_2}(L)$ folgt die Behauptung.
Diese Ungleichung folgt aus der Existenz von i linear unabhängigen Gittervektoren $a_1,\ldots,a_i$ mit euklidischer Norm höchstens $\lambda_i(L)$. Damit gibt es mindestens ein $j\leq i$ mit $\|\pi_i(a_j)\|_2\neq 0$ und es folgt $\lambda_{1,\|.\|_2}(L_i)\leq\|\pi_i(a_j)\|_2\leq\lambda_{i,\|.\|_2}(L)$.

zu (2.6): Jede (β,δ)–blockreduzierte Basis ist auch längenreduziert. Daher gilt

$$\begin{aligned}\|b_i\|_2^2 &= \|\hat{b}_i\|_2^2+\sum_{j=1}^{i-1}\mu_{i,j}^2\|\hat{b}_j\|_2^2\leq\|\hat{b}_i\|_2^2+\frac{1}{4}\sum_{j=1}^{i-1}\|\hat{b}_j\|_2^2\\ &\overset{(2.5)}{\leq} \frac{1}{\delta}\left(\frac{\gamma_\beta}{\delta}\right)^{2\frac{m-i}{\beta-1}}\lambda_{i,\|.\|_2}^2(L)+\frac{1}{4}\sum_{j=1}^{i-1}\frac{1}{\delta}\left(\frac{\gamma_\beta}{\delta}\right)^{2\frac{m-j}{\beta-1}}\lambda_{j,\|.\|_2}^2(L)\\ &\leq \frac{i+3}{4\delta}\left(\frac{\gamma_\beta}{\delta}\right)^{2\frac{m-1}{\beta-1}}\lambda_{i,\|.\|_2}^2(L).\end{aligned}$$

zu (2.7): Nach Definition der sukzessiven Minima gilt

$$\lambda^2_{i,\|\cdot\|_2}(L) \leq \max_{j=1,\ldots,i} \|b_j\|_2^2.$$

Wegen $\|b_j\|_2^2 = \|\hat{b}_j\|_2^2 + \sum_{k=1}^{j-1} \mu_{j,k}^2 \|\hat{b}_k\|_2^2$ und $\mu_{j,k}^2 \leq \frac{1}{4}$ folgt

$$\lambda^2_{i,\|\cdot\|_2}(L) \leq \frac{i+3}{4} \max_{j=1,\ldots,i} \|\hat{b}_j\|_2^2. \tag{2.9}$$

Wir wenden Lemma 2.4 auf die (β,δ)–blockreduzierten Basen $\pi_j(b_j),\ldots,\pi_j(b_i)$ an und erhalten

$$\|\hat{b}_j\|_2^2 \leq \left(\frac{\gamma_\beta}{\delta}\right)^{2\frac{i-j}{\beta-1}} \max_{k=i-\beta+2,\ldots,i} \|\hat{b}_k\|_2^2 \quad \text{für } 1 \leq j \leq i-\beta+1. \tag{2.10}$$

Andererseits gilt nach Definition der (β,δ)–blockreduzierten Basen

$$\delta\|\hat{b}_k\|_2^2 \leq \lambda^2_{1,\|\cdot\|_2}(L(\pi_k(b_k),\ldots,\pi_k(b_i))) \leq \|\pi_k(b_i)\|_2^2 \leq \|b_i\|_2^2 \quad \text{für } i-\beta+2 \leq k \leq i. \tag{2.11}$$

Aus (2.10) und (2.11) folgt

$$\|\hat{b}_j\|_2^2 \leq \frac{1}{\delta}\left(\frac{\gamma_\beta}{\delta}\right)^{2\frac{i-j}{\beta-1}} \|b_i\|_2^2 \quad \text{für } 1 \leq j \leq i.$$

Mit (2.9) folgt die Behauptung. $\square$

Mit Hilfe von Ungleichung (2.2) erhalten wir eine weitere obere Schranke für die Höhen einer (β,δ)–blockreduzierten Gitterbasis, die unabhängig von den (nicht exakt bekannten) Hermite–Konstanten ist.

Satz 2.5 *Sei $b_1,\ldots,b_m$ eine (β,δ)–blockreduzierte Basis des Gitters L mit $2 \leq \beta \leq m$ und $\frac{1}{4} < \delta \leq 1$. Dann gilt mit $\alpha := \frac{1}{\delta-\frac{1}{4}}$ für $1 \leq j \leq m$:*

$$\frac{\|\hat{b}_j\|_2^2}{\lambda^2_{j,\|\cdot\|_2}(L)} \leq \alpha^{m-j}\delta^{m-j-\left\lceil\frac{m-j}{\beta-1}\right\rceil}.$$

Zum Beweis von Satz 2.5 benutzen wir das folgende

Lemma 2.6 *Sei $b_1,\ldots,b_m$ eine (β,δ)–blockreduzierte Gitterbasis mit $2 \leq \beta \leq m$ und $\frac{1}{4} < \delta \leq 1$. Dann gilt mit $\alpha := \frac{1}{\delta-\frac{1}{4}}$*

$$\|\hat{b}_j\|_2^2 \leq \alpha^i \delta^{i-\left\lceil\frac{i}{\beta-1}\right\rceil} \|\hat{b}_{j+i}\|_2^2 \quad \textit{für } 1 \leq j < m \quad \textit{und} \quad 1 \leq i \leq m-j.$$

Beweis von Lemma 2.6:
Zur Abkürzung setzen wir $c_i := \|\hat{b}_i\|_2^2$ für $i = 1, \ldots, m$. Nach Definition der (β, δ)–Blockreduziertheit und (2.2) gilt für jedes $i < \beta$ und alle j:

$$\delta c_j \leq \|\pi_j(b_{j+i})\|_2^2 \leq c_{j+i} + \frac{1}{4} \sum_{t=0}^{i-1} c_{j+t}.$$

Wir erhalten

$$c_j \leq \alpha \Big(c_{j+i} + \frac{1}{4} \sum_{t=1}^{i-1} c_{j+t} \Big).$$

Durch Induktion über i ergibt sich daraus

$$c_j \leq \alpha \left(1 + \frac{\alpha}{4}\right)^{i-1} c_{j+i} \quad \text{für } \; 1 \leq i < \beta\ .$$

Wegen $1 + \frac{\alpha}{4} = \alpha\delta$ folgt

$$c_j \leq \alpha^i \delta^{i-1} c_{j+i} \quad \text{für } 1 \leq i < \beta\ . \tag{2.12}$$

Durch sukzessives Anwenden der Ungleichung (2.12) folgt die Behauptung. □

Beweis von Satz 2.5:
Für jede Basis $b_1, \ldots, b_m$ eines Gitters L gilt $\lambda^2_{j,\|.\|_2}(L) \geq \min\limits_{i=j\ldots,m} \|\hat{b}_i\|_2^2$. Daher existiert ein $k \geq j$ mit $\lambda^2_{j,\|.\|_2}(L) \geq \|\hat{b}_k\|_2^2$. Es folgt

$$\begin{aligned} \lambda^2_{j,\|.\|_2}(L) &\geq \|\hat{b}_k\|_2^2 = \|\hat{b}_{j+(k-j)}\|_2^2 \\ &\overset{\text{Lemma 2.6}}{\geq} \alpha^{-(k-j)} \delta^{\left\lceil \frac{k-j}{\beta-1} \right\rceil - (k-j)} \|\hat{b}_j\|_2^2 \\ &\geq \alpha^{-(m-j)} \delta^{\left\lceil \frac{m-j}{\beta-1} \right\rceil - (m-j)} \|\hat{b}_j\|_2^2. \end{aligned}$$

□

2.1.2 Der Fall allgemeiner Norm

In Analogie zur Hermite–Konstante γ_m definieren wir für beliebige Normen $\|.\|$ auf $\mathbb{R}^m$ die Konstanten

$$\kappa_{m,\|.\|} := \sup_{\substack{b_1,\ldots,b_m \\ \text{Basis von } L \subset \mathbb{R}^m}} \lambda_{1,\|.\|}(L(b_1, \ldots, b_m)) \left(\prod_{i=1}^m F_i(b_i) \right)^{-1/m}$$

und $\kappa_m := \sup \kappa_{m,\|.\|}$, wobei das Supremum über alle Normen auf $\mathbb{R}^m$ gebildet wird.

κ_m ist wohldefiniert, denn es gilt

Satz 2.7 [Kai94] *Für jede Basis $b_1, \ldots, b_m$ eines Gitters $L \subset \mathbb{R}^n$ gilt:*

$$\min_{j=1,\ldots,m} F_j(b_j) \leq \lambda_{1,\|.\|}(L) \leq \left(m! \prod_{i=1}^m F_i(b_i) \right)^{1/m}.$$

Aus diesem Satz ergibt sich unmittelbar die Abschätzung $\sqrt{\gamma_m} \leq \kappa_m \leq m!^{1/m}$ und wir erhalten für $m \to \infty$ die asymptotischen Schranken

$$\sqrt{\frac{m}{2\pi e}}(1+o(1)) \leq \kappa_m \leq \frac{m}{e}(1+o(1)). \tag{2.13}$$

Für $m \geq 1$ gilt $\kappa_{m+1}^{m+1} \geq 2\kappa_m^m$ [Kai94]. Wegen $\kappa_1 = 1$ folgt daraus $\kappa_m^m \geq 2^{m-1}$ und $\kappa_m^2 \geq 2$ für $m \geq 2$. Wegen $2!^{1/2} = \sqrt{2}$ erhalten wir $\kappa_2 = \sqrt{2}$.

Analog zu den (β, δ)–blockreduzierten Basen bezüglich der euklidischen Norm approximieren auch (β, Δ)–blockreduzierte Gitterbasen $b_1, \ldots, b_m$ die sukzessiven Minima bis auf einen (in m/β) exponentiellen Faktor. Dieser Faktor ist jedoch für allgemeine Normen größer als bei der euklidischen Norm. Die Approximation wird mit wachsender Blockgröße und wachsendem Δ besser. Ist β ein konstanter Bruchteil von m, so werden die sukzessiven Minima bis auf einen in m polynomialen Faktor approximiert.

Theorem 2.8 *Seien $2 \leq \beta \leq m$ und $0 < \Delta \leq 1$. Für jede (β, Δ)–blockreduzierte Basis $b_1, \ldots, b_m$ eines Gitters $L \subset \mathbb{R}^n$ gilt:*

$$\frac{\|b_i\|}{\lambda_{i,\|\cdot\|}(L)} \leq \frac{i+1}{4}\left(\frac{\kappa_\beta}{\Delta}\right)^{2\frac{m-1}{\beta-1}} \quad \textit{für } 1 \leq i \leq m, \tag{2.14}$$

$$\frac{\|b_i\|}{\lambda_{i,\|\cdot\|}(L)} \geq \frac{2\Delta}{i+1} \quad \textit{für } 1 \leq i \leq \beta, \tag{2.15}$$

$$\frac{\|b_i\|}{\lambda_{i,\|\cdot\|}(L)} \geq \frac{4}{i+1}\left(\frac{\Delta}{\kappa_\beta}\right)^{2\frac{i-1}{\beta-1}} \quad \textit{für } \beta \leq i \leq m. \tag{2.16}$$

Für $\Delta = 1$ wurde dieses Theorem bereits von Kaib [Kai94] gezeigt. Wir lehnen uns eng an diesen Beweis an und zeigen zunächst das folgende

Lemma 2.9 *Sei $0 < \Delta \leq 1$ und $2 \leq \beta \leq m$. Für jede (β, Δ)–blockreduzierte Basis $b_1, \ldots, b_m$ eines Gitters $L \subset \mathbb{R}^n$ gilt*

$$\|b_1\| \leq \frac{\Delta}{2}\left(\frac{\kappa_\beta}{\Delta}\right)^{2\frac{m-1}{\beta-1}} \max_{i=m-\beta+1,\ldots,m-1} F_i(b_i), \tag{2.17}$$

$$\|b_1\| \leq \frac{1}{2}\left(\frac{\kappa_\beta}{\Delta}\right)^{2\frac{m-1}{\beta-1}} \lambda_{1,\|\cdot\|}(L). \tag{2.18}$$

Beweis von Lemma 2.9:
zu (2.17): Sei $h_i := F_i(b_i)$ die i–te Höhe. Nach Definition der Blockreduziertheit gelten die Ungleichungen

$$\begin{aligned} \Delta^i h_1^i &\leq \lambda_{1,\|\cdot\|}^i(L(b_1, \ldots, b_i)) \quad \text{für} \quad i = 1, \ldots, \beta - 1 \quad \text{und} \\ \Delta^\beta h_i^\beta &\leq \lambda_{1,F_i}^\beta(L(b_i, \ldots, b_{i+\beta-1})) \quad \text{für} \quad i = 1, \ldots, m - \beta. \end{aligned}$$

Nach Definition von κ_β folgt daraus

$$\begin{aligned}\Delta^i h_1^i &\leq \kappa_i^i h_1 \cdots h_i \quad \text{für} \quad i = 1, \ldots, \beta - 1 \quad \text{und} \\ \Delta^\beta h_i^\beta &\leq \kappa_\beta^\beta h_i \cdots h_{i+\beta-1} \quad \text{für} \quad i = 1, \ldots, m - \beta.\end{aligned}$$

Multiplikation dieser Ungleichungen ergibt

$$\Delta^{\binom{\beta}{2}+\beta(m-\beta)} h_1^{\binom{\beta+1}{2}} h_2^\beta \cdots h_{m-\beta}^\beta \leq \kappa_1^1 \kappa_2^2 \cdots \kappa_{\beta-1}^{\beta-1} \kappa_\beta^{\beta(m-\beta)} h_1^\beta h_2^\beta \cdots h_{m-\beta}^\beta h_{m-\beta+1}^{\beta-1} \cdots h_{m-1}^1.$$

Durch Kürzen erhalten wir daraus

$$\begin{aligned}\Delta^{\binom{\beta}{2}+\beta(m-\beta)} h_1^{\binom{\beta}{2}} &\leq \kappa_1^1 \kappa_2^2 \cdots \kappa_{\beta-1}^{\beta-1} \kappa_\beta^{\beta(m-\beta)} h_{m-\beta+1}^{\beta-1} \cdots h_{m-1}^1 \\ &\leq \kappa_1^1 \kappa_2^2 \cdots \kappa_{\beta-1}^{\beta-1} \kappa_\beta^{\beta(m-\beta)} \left(\max_{i=m-\beta+1,\ldots,m-1} F_i(b_i) \right)^{\binom{\beta}{2}}.\end{aligned}$$

Wegen $\kappa_m^m \leq \frac{1}{2} \kappa_{m+1}^{m+1}$ für $m \leq 1$ gilt $\kappa_1^1 \kappa_2^2 \cdots \kappa_{\beta-1}^{\beta-1} \kappa_\beta^{\beta(m-\beta)} \leq \left(\frac{1}{2}\right)^{\binom{\beta}{2}} \kappa_\beta^{\beta(m-1)}$.

zu (2.18): Der Beweis erfolgt durch Induktion über $m \geq \beta$:
Für $m = \beta$ gilt $\Delta \|b_1\| \leq \lambda_{1,\|.\|}(L)$ nach Definition 2.1. Wegen $\kappa_\beta^2 \geq 2$ und $\Delta^{-1} \leq \Delta^{-2}$ folgt daraus die Behauptung.
Sei die Behauptung für $m - 1$ bereits gezeigt. Wir betrachten die Darstellung $b = \sum_{i=1}^m u_i b_i$ eines kürzesten Gittervektors.
Für $u_m = 0$ ist $\lambda_{1,\|.\|}(L) = \lambda_{1,\|.\|}(L(b_1, \ldots, b_{m-1}))$ und die Behauptung folgt aus der Induktionsannahme.
Andernfalls gilt für $m - \beta + 1 \leq i \leq m$ die Ungleichung

$$\lambda_{1,\|.\|}(L) = \|b\| \geq F_i(b) \geq \lambda_{1,F_i}(L(b_i, \ldots, b_m)) \geq \Delta F_i(b_i).$$

Damit gilt $\max_{i=m-\beta+1,\ldots,m-1} F_i(b_i) \leq \lambda_{1,\|.\|}(L) \Delta^{-1}$.
Die Behauptung folgt nun unmittelbar aus (2.17). □

Beweis von Theorem 2.8:
zu (2.14): Nach Definition 2.1 ist $b_j, \ldots, b_m$ (β, Δ)–blockreduziert bezüglich der Norm F_j für $j = 1, \ldots, m$. Mit Lemma 2.9 erhalten wir

$$F_j(b_j) \leq \frac{1}{2} \left(\frac{\kappa_\beta}{\Delta} \right)^{2\frac{m-j}{\beta-1}} \lambda_{1,F_j}(L(b_j, \ldots, b_m)) \quad \text{für } 1 \leq j \leq m - \beta + 1.$$

Für $m - \beta + 1 \leq j \leq m$ gilt wegen $\kappa_\beta^{2\frac{m-1}{\beta-1}} \geq 2$ und $\Delta^{-1} \leq \Delta^{-2\frac{m-1}{\beta-1}}$

$$F_j(b_j) \leq \Delta^{-1} \lambda_{1,F_j}(L(b_j, \ldots, b_m)) \leq \frac{1}{2} \left(\frac{\kappa_\beta}{\Delta} \right)^{2\frac{m-1}{\beta-1}} \lambda_{1,F_j}(L(b_j, \ldots, b_m)).$$

Wegen $\lambda_{1,F_j}(L(b_j,\ldots,b_m)) \le \lambda_{i,\|.\|}(L)$ für $j \le i$ ergibt sich daraus mit Ungleichung (2.1)

$$\begin{aligned}\|b_i\| &\le F_i(b_i) + \frac{1}{2}\sum_{j=1}^{i-1} F_j(b_j) \\ &\le \frac{1}{2}\left(\frac{\kappa_\beta}{\Delta}\right)^{2\frac{m-1}{\beta-1}} \Big(1+\frac{1}{2}(i-1)\Big)\lambda_{i,\|.\|}(L) \\ &= \frac{i+1}{4}\left(\frac{\kappa_\beta}{\Delta}\right)^{2\frac{m-1}{\beta-1}} \lambda_{i,\|.\|}(L) \quad \text{für } 1 \le i \le m.\end{aligned}$$

zu (2.15): Nach Definition der sukzessiven Minima und Ungleichung (2.1) gilt

$$\lambda_{i,\|.\|}(L) \le \max_{j=1,\ldots,i} \|b_j\| \le \frac{i+1}{2} \max_{j=1,\ldots,i} F_j(b_j). \tag{2.19}$$

Aus Definition 2.1 ergibt sich

$$\begin{aligned}F_j(b_j) &\le \frac{1}{\Delta}\min\{F_j(b) | b \in L(b_j,\ldots,b_{\min(j+\beta-1,m)}) - 0\} \\ &\le \Delta^{-1}F_j(b_i) \le \Delta^{-1}\|b_i\| \quad \text{für } \max(1, i-\beta+1) \le j \le i.\end{aligned}$$

Für $1 \le i \le \beta$ erhalten wir daraus $\max_{j=1,\ldots,i} F_j(b_j) \le \Delta^{-1}\|b_i\|$.
zu (2.16): Für $i \ge \beta$ und $1 \le j \le i-\beta+1$ ist $b_j,\ldots,b_i$ (β,Δ)–blockreduziert bezüglich der Norm F_j und wir erhalten mit Lemma 2.9

$$\begin{aligned}F_j(b_j) &\le \frac{1}{2}\left(\frac{\kappa_\beta}{\Delta}\right)^{2\frac{i-j}{\beta-1}} \lambda_{1,F_j}(L(b_j,\ldots,b_i)) \\ &\le \frac{1}{2}\left(\frac{\kappa_\beta}{\Delta}\right)^{2\frac{i-j}{\beta-1}} F_j(b_i) \\ &\le \frac{1}{2}\left(\frac{\kappa_\beta}{\Delta}\right)^{2\frac{i-1}{\beta-1}} \|b_i\|.\end{aligned}$$

Für $i-\beta+1 \le j \le i$ gilt $F_j(b_j) \le \Delta^{-1}\|b_i\| \le \frac{1}{2}\left(\frac{\kappa_\beta}{\Delta}\right)^{2\frac{i-1}{\beta-1}} \|b_i\|$.

Mit Ungleichung (2.19) folgt die Behauptung. □

Bemerkung 2.10 Alle Normen auf $\mathbb{R}^n$ sind äquivalent, d.h. zu jeder Norm $\|.\|$ auf $\mathbb{R}^n$ gibt es positive Konstanten $r = r(\|.\|)$ und $R = R(\|.\|)$, so daß $r\|x\| \le \|x\|_2 \le R\|x\|$ gilt für alle $x \in \mathbb{R}^n$. Damit gilt für jedes Gitter $L \subset \mathbb{R}^n$ die Ungleichung

$$\lambda_{1,\|.\|_2}(L) \le R(\|.\|)\lambda_{1,\|.\|}(L)$$

und wir erhalten mit Theorem 2.3:

Für jede (β, δ)–blockreduzierte Basis $b_1, \ldots, b_m$ eines Gitters $L \subset \mathbb{R}^n$ mit $2 \leq \beta \leq m$ und $0 < \delta \leq 1$ gilt

$$\|b_1\| \leq \frac{R(\|.\|)}{r(\|.\|)} \frac{1}{\sqrt{\delta}} \left(\frac{\gamma_\beta}{\delta}\right)^{\frac{m-1}{\beta-1}} \lambda_{1,\|.\|}(L). \tag{2.20}$$

Der erste Vektor einer bezüglich der euklidischen Norm blockreduzierten Gitterbasis approximiert also das erste sukzessive Minimum auch in jeder anderen Norm in annähernd gleicher Güte.

Zum Vergleich der Schranken (2.14) und (2.20) für l_p–Normen setzen wir $\Delta = \sqrt{\delta}$.

Für l_p–Normen gilt

$$r(\|.\|_p) = \begin{cases} n^{1/2-1/p} & \text{für } p < 2 \\ 1 & \text{für } p > 2 \end{cases} \quad \text{und} \quad R(\|.\|_p) = \begin{cases} 1 & \text{für } p < 2 \\ n^{1/2-1/p} & \text{für } p > 2 \end{cases} \tag{2.21}$$

und damit $R(\|.\|_p)/r(\|.\|_p) \leq \sqrt{n}$.

Wir erhalten

$$\|b_1\|_p \leq \frac{\sqrt{n}}{\Delta} \left(\frac{\gamma_\beta}{\Delta^2}\right)^{\frac{m-1}{\beta-1}} \lambda_{1,\|.\|_p}(L) \tag{2.22}$$

für (β, Δ^2)–blockreduzierte Basen in euklidischer Norm und

$$\|b_1\|_p \leq \frac{1}{2} \left(\frac{\kappa_\beta^2}{\Delta^2}\right)^{\frac{m-1}{\beta-1}} \lambda_{1,\|.\|_p}(L) \tag{2.23}$$

für (β, Δ)–blockreduzierte Basen in l_p–Norm.

Für Blockgrößen β „nahe bei" m und große Dimensionen n liefert die (β, Δ)–Bockreduktion bessere Schranken als die (β, δ)–Bockreduktion mit $\delta = \Delta^2$.

Anders verhält es sich bei konstanten Blockgrößen und wachsendem Rang des Gitters. Für jedes β mit $\kappa_\beta^2 > \gamma_\beta$ und hinreichend großen Rang m ist die Schranke (2.22) kleiner als (2.23). (Für $2 \leq \beta \leq 20$ folgt $\kappa_\beta^2 > \gamma_\beta$ aus $\kappa_\beta^\beta \geq 2^{\beta-1}$ und (2.3). Für $\beta > 20$ ist lediglich $\kappa_\beta^2 \geq \gamma_\beta$ bekannt.)
Im Fall der l_∞–Norm ist (2.23) für $(\beta, \Delta) = (2, 1)$ scharf, wie das folgende Beispiel zeigt: Die Basis

$$B := \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ \\ b_m \end{pmatrix} := \begin{pmatrix} 2^{m-2} & 2^{m-2} & 0 & \cdots & 0 \\ 2^{m-2} & 0 & 2^{m-3} & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ 2^{m-2} & 0 & \cdots & 0 & 1 \\ 1 & 1 & \cdots & 1 & 1 \end{pmatrix}$$

ist LS–reduziert mit $\Delta = 1$ und $\|b_1\|_\infty / \lambda_{1,\|.\|_\infty}(L(B)) = \|b_1\|_\infty / \|b_m\|_\infty = 2^{m-2} = \frac{1}{2}\kappa_2^{2(m-1)}$. Demgegenüber gilt die obere Schranke $\|b_1\|_\infty / \lambda_{1,\|.\|_\infty}(L(B)) \leq \sqrt{n}\sqrt{4/3}^{m-1}$ für jede mit

$\delta = 1$ L^3–reduzierte Gitterbasis. Damit approximieren L^3–reduzierte Gitterbasen das erste sukzessive Minimum in l_∞–Norm im worst case besser als LS-reduzierte Basen. Testläufe mit L^3–Reduktion und LS-Reduktion in l_∞–Norm haben diesen Effekt bestätigt; im Mittel war bei identischen Eingabebasen die l_∞–Norm des ersten Vektors einer L^3–reduzierten Basis kleiner als die einer LS–reduzierten Basis desselben Gitters.

2.2 Verfahren zur Blockreduktion

Wir behandeln zunächst wieder den Fall der euklidischen Norm. Hierfür sind bereits effiziente Algorithmen bekannt, die wir im folgenden vorstellen.

Für $\beta = 2$ und $1/4 < \delta < 1$ benötigt der folgende Algorithmus zur L^3–Reduktion mit iterativer Orthogonalisierung bei ganzzahliger Eingabebasis polynomial (in der Länge der Eingabe) viele arithmetische Operationen ([SE94], siehe auch Satz 1.6).

Algorithmus L^3–Reduktion
EINGABE: $n, m, \quad b_1, \ldots, b_m \in \mathbb{R}^n, \quad \delta \quad$ mit $\quad 0 \le \delta \le 1$

1. $k := 2$ (Bei Eintritt in Stufe k ist die Basis $b_1, \ldots, b_{k-1}$ bereits mit δ L^3–reduziert und es liegen die Gram–Schmidt–Koeffizienten $\mu_{i,j}$ für $1 \le j < i < k$ sowie die Höhenquadrate $c_i = \|\hat{b}_i\|_2^2$ für $i = 1, \ldots, k-1$ vor.)

2. WHILE $k \le m$
    IF $k = 2$ THEN $c_1 := \|b_1\|_2^2$
    FOR $j = 1, \ldots, k-1$
        $\mu_{k,j} := (< b_k, b_j > - \sum_{i=1}^{j-1} \mu_{j,i}\mu_{k,i}c_i)/c_j$
    $c_k :=< b_k, b_k > - \sum_{j=1}^{k-1} \mu_{k,j} c_j$

3. (längenreduziere b_k)
    FOR $j = k-1, \ldots, 1$
        $\mu := \lceil \mu_{k,j} \rfloor$
        FOR $i = 1, \ldots, j-1$
            $\mu_{k,i} := \mu_{k,i} - \mu\mu_{j,i}$
        $\mu_{k,j} := \mu_{k,j} - \mu$
        $b_k := b_k - \mu b_j$

4. IF $\delta c_{k-1} > c_k + \mu_{k,k-1}^2 c_{k-1}$
    THEN vertausche b_k und b_{k-1}
        $k := \max(k-1, 2)$
    ELSE $k := k+1$

END while

AUSGABE: mit δ L^3–reduzierte Basis $b_1, \ldots, b_m$

Um das Gitter nicht zu verändern, müssen die Basistransformationen in exakter Arithmetik durchgeführt werden. Bei praktischen Implementierungen beschränken wir uns daher auf ganzzahlige Gitterbasen. Eine Beschleunigung des Algorithmus kann dadurch erzielt werden, daß wesentliche Teile der Berechnungen in Gleitpunkt–Arithmetik durchgeführt werden. Lediglich die Transformationen der Basis und einige Korrekturschritte sind in exakter Arithmetik durchzuführen. Die Details dieses Verfahrens finden sich in [SE94].

In der Praxis hat sich die Ersetzung von Schritt 4 durch die von Schnorr und Euchner [SE94] vorgeschlagene Regel der „t tiefen Einfügungen" bewährt, die im allgemeinen dazu führt, daß die reduzierte Basis aus kürzeren Vektoren besteht. b_k wird an der Stelle mit minimalem Index i ($i \in \{1, \ldots, \min(t, k-1), k-1\}$), in die Basis einsortiert, an der $\|\pi_i(b_k)\|_2^2 < \delta c_i$ gilt. Für festes t bleibt die Laufzeit des Algorithmus polynomial in n, m und der Länge der Eingabe.

4. (t tiefe Einfügungen)
   $c := \|b_k\|_2^2, \quad T := \min(t, k-1), \quad i := 1$
   WHILE $i < T$
       IF $\delta c_i > c$
       THEN $(b_1, \ldots, b_k) := (b_1, \ldots, b_{i-1}, b_k, b_i, \ldots, b_{k-1})$
           $k := \max(i-1, 2)$
           GOTO 2.
       ELSE $c := c - \mu_{k,i}^2 c_i$
           $i := i + 1$
   IF $\delta c_{k-1} > c_k + \mu_{k,k-1}^2 c_{k-1}$
   THEN vertausche b_k und b_{k-1}
       $k := \max(k-1, 2)$
   ELSE $k := k + 1$

Für $\beta > 2$ ist kein Verfahren zur (β, δ)–Blockreduktion mit bewiesener polynomialer Laufzeitschranke bekannt. (Lediglich für $\beta = 3$ und $\delta \in [\frac{1}{2}, \frac{1}{2}\sqrt{3})$ können wir einen Algorithmus mit polynomialer Laufzeitschranke angeben; siehe Kapitel 4.1.1.) In der Praxis hat sich der folgende Algorithmus von Schnorr [SE94] zur (β, δ)–Blockreduktion für kleine Blockgrößen ($\beta \leq 30$) als effizient erwiesen. Der Kern des Algorithmus ist das Aufzählungsverfahren ENUM(j, k), das eine ganzzahlige, von Null verschiedene Minimalstelle $(u_j, \ldots, u_k)$ des Ausdrucks

$$c_j(\widetilde{u}_j, \ldots, \widetilde{u}_k) := \left\| \pi_j\Big(\sum_{i=j}^{k} \widetilde{u}_i b_i\Big) \right\|_2^2 \tag{2.24}$$

bestimmt. In Kapitel 2.3 und 2.4 stellen wir grundsätzliche Methoden für diese Aufzählung vor, die in Kapitel 3 für l_p–Normen verfeinert werden.

Algorithmus (β, δ)**–Blockreduktion**
EINGABE: $b_1, \ldots, b_m \in \mathbb{R}^n$, $\quad \beta$ mit $2 \leq \beta \leq m$, $\quad \delta$ mit $0 \leq \delta \leq 1$
1. längenreduziere $b_1, \ldots, b_\beta$, $\; j := m-1, z := 0$
2. WHILE $z < m-1$
    $j := j+1$, IF $j = m$ THEN $j := 1$
    $k := \min(j+\beta-1, m)$
    ENUM(j,k) (ENUM bestimmt die ganzzahligen Koeffizienten $(u_j, \ldots, u_k)$
        der Darstellung eines Vektors $b_j^{\text{neu}} = \sum_{i=j}^k u_i b_i$
        mit $\bar{c}_j := \|\pi_j(b_j^{\text{neu}})\|_2^2 = \lambda^2_{1,\|.\|_2}(L(\pi_j(b_j), \ldots, \pi_j(b_k))))$
    $h := \min(k+1, m)$
    IF $\bar{c}_j < \delta c_j$
    THEN ergänze $b_1, \ldots, b_{j-1}, b_j^{\text{neu}}$
        zu einer Basis $b_1, \ldots, b_{j-1}, b_j^{\text{neu}}, \ldots, b_k^{\text{neu}}, b_{k+1}, \ldots, b_m$ von L
        längenreduziere $b_1, \ldots, b_{j-1}, b_j^{\text{neu}}, \ldots, b_h^{\text{neu}}$
        $z := 0$
    ELSE längenreduziere $b_1, \ldots, b_h$
        $z := z+1$
  END while
AUSGABE: (β, δ)–blockreduzierte Basis $b_1, \ldots, b_m$

Die Variable z zählt die aufeinanderfolgenden Positionen j, an denen die Ungleichung $\delta\|\hat{b}_j\|_2^2 \leq \lambda^2_{1,\|.\|_2}(L(\pi_j(b_j), \ldots, \pi_j(b_k)))$ gilt. Falls die Ungleichung nicht erfüllt ist, fügen wir den Vektor b_j^{neu} in die Basis ein und setzen den Zähler wieder auf 0. j wird zyklisch über den Zahlen $1, \ldots, m-1$ variiert. Die Position $j = m$ wird ausgelassen, denn die Ungleichung gilt stets für $j = m$. Offensichtlich ist die Basis (β, δ)–blockreduziert, falls $z = m-1$.

Alternativ kann die Längenreduktion in Schritt 1 und nach dem Einfügen des Vektors b_j^{neu} auch durch eine L^3–Reduktion mit δ ersetzt werden. Dies führt bei praktischen Experimenten fast immer zu kleineren Laufzeiten und kürzeren Basisvektoren.

Bei der Ergänzung der Vektoren $b_1, \ldots, b_{j-1}, b_j^{\text{neu}} = \sum_{i=j}^k u_i b_i$ zu einer Basis von L unterscheiden wir zwei Fälle. Sei $g := \max\{i : j \leq i \leq k, u_i \neq 0\}$.

Falls $|u_g| = 1$ ist, dann ist $b_1, \ldots, b_{j-1}, b_j^{\text{neu}}, b_j, \ldots, b_{g-1}, b_{g+1}, \ldots, b_m$ eine Basis des Gitters, d.h. wir können den Vektor b_g aus der Basis entfernen und dafür den Vektor b_j^{neu} in die Basis einsortieren.

Für $|u_g| > 1$ transformieren wir die Basis $b_1, \ldots, b_m$ mittels des erweiterten Euklidischen Algorithmus so zu $b_1, \ldots, b_{j-1}, b'_j, \ldots, b'_g, b_{g+1}, \ldots, b_m$, daß $b_j^{\text{neu}} = \sum_{i=j}^g u'_i b'_i$ mit $|u'_g| = 1$. Dies ist stets möglich, denn für die Ausgabe von ENUM(j,k) gilt stets $\text{ggT}(u_j, \ldots, u_k) = 1$. Andernfalls wäre auch $b := b_j^{\text{neu}}/\text{ggT}(u_j, \ldots, u_k)$ ein Gittervektor mit

$\|\pi_j(b)\|_2^2 = \bar{c}_j/\mathrm{ggT}(u_j,\ldots,u_k)^2$ und damit $\bar{c}_j > \lambda^2_{1,\|\cdot\|_2}(L(\pi_j(b_j),\ldots,\pi_j(b_k)))$. Wir können b'_g aus der Basis entfernen und dafür den Vektor b_j^{neu} in die Basis einsortieren.

Die folgende Routine BASIS formalisiert dieses Verfahren. Bei Eintritt in Schritt 3 gilt dabei stets $b_j^{\text{neu}} = \sum_{i=j}^k u_i b_i$.

Algorithmus BASIS
EINGABE: $(u_j,\ldots,u_k) \in \mathbb{Z}^{k-j-1} - 0^{k-j-1}, \quad b_j,\ldots,b_k$
1. $b_j^{\text{neu}} = \sum_{i=j}^k u_i b_i$
2. $g := \max\{t : j \le t \le k, u_t \neq 0\}$
3. WHILE $|u_g| > 1$
   $i := \max\{t : j \le t < g, u_t \neq 0\}$ (ein solches i existiert immer, denn es gilt $\mathrm{ggT}(u_j,\ldots,u_g) = 1$)
   $q := \lceil u_g/u_i \rfloor$
   $u_i := u_g - q \cdot u_i$
   $u_g := u_i^{\text{alt}}$
   $b_g := q \cdot b_g + b_i$
   $b_i := b_g^{\text{alt}}$
4. FOR $i = g,\ldots,j+1 \qquad b_i := b_{i-1}$
5. $b_j := b_j^{\text{neu}}$

AUSGABE: $b_j,\ldots,b_k$

Die Algorithmen für den Fall der euklidischen Norm lassen sich weitgehend auf den allgemeinen Fall übertragen. Für allgemeine Normen und $\beta = 2$ führt der folgende Algorithmus zur LS–Reduktion für festes m und $\frac{1}{2} < \Delta < 1$ polynomial (in der Länge der Eingabe) viele Schritte durch [LS92].

Algorithmus LS–Reduktion
EINGABE: $b_1,\ldots,b_m \in \mathbb{R}^n, \quad \Delta \quad \text{mit} \quad 0 \le \Delta \le 1$
$k := 2$
WHILE $k \le m$
  (längenreduziere b_k)
  FOR $j = k-1,\ldots,1$
    bestimme eine ganzzahlige Minimalstelle μ_j von $F_j(b_k - \mu b_j)$
    $b_k := b_k - \mu_j b_j$
  IF $F_{k-1}(b_k) < \Delta F_{k-1}(b_{k-1})$
  THEN vertausche b_k und b_{k-1},
    setze $k := \max(k-1, 2)$
  ELSE setze $k := k+1$

AUSGABE: mit Δ LS–reduzierte Basis $b_1,\ldots,b_m$

Analog zum Fall der euklidischen Norm ist auch für den folgenden Algorithmus zur (β, Δ)–Blockreduktion für $\beta \geq 3$ keine (in β) polynomiale obere Schranke für die Anzahl der Iterationen von Schritt 2 bekannt.

Algorithmus (β, Δ)–**Blockreduktion**
EINGABE: $b_1, \ldots, b_m \in \mathbb{R}^n$, β mit $2 \leq \beta \leq m$, Δ mit $0 \leq \Delta \leq 1$
1. längenreduziere $b_1, \ldots, b_m$, $j := m - 1, z := 0$
2. WHILE $z < m - 1$
   $j := j + 1$, IF $j = m$ THEN $j := 1$
   $k := \min(j + \beta - 1, m)$
   ENUM(j, k) (ENUM bestimmt die ganzzahligen Koeffizienten $(u_j, \ldots, u_k)$
   der Darstellung eines Vektors $b_j^{\text{neu}} = \sum_{i=j}^{k} u_i b_i$
   mit $\bar{F}_j := F_j(b_j^{\text{neu}}) = \lambda_{1,F_j}(L(b_j, \ldots, b_k)))$
   $h := \min(k + 1, m)$
   IF $\bar{F}_j < \Delta F_j(b_j)$
   THEN ergänze $b_1, \ldots, b_{j-1}, b_j^{\text{neu}}$
   zu einer Basis $b_1, \ldots, b_{j-1}, b_j^{\text{neu}}, \ldots, b_k^{\text{neu}}, b_{k+1}, \ldots, b_m$ von L
   längenreduziere $b_1, \ldots, b_{j-1}, b_j^{\text{neu}}, \ldots, b_h^{\text{neu}}$
   $z := 0$
   ELSE längenreduziere $b_1, \ldots, b_h$
   $z := z + 1$
   END while
AUSGABE: (β, Δ)–blockreduzierte Basis $b_1, \ldots, b_m$

Alternativ zu den Längenreduktionen können im Algorithmus (β, Δ)–Blockreduktion LS–Reduktionen mit Δ durchgeführt werden.

2.3 Bestimmung kürzester Gittervektoren

Der Kern der Verfahren zur Blockreduktion ist die Routine ENUM(j,k) zur Bestimmung eines Vektors in $L(b_j, \ldots, b_k)$ mit minimaler Norm F_j. Gesucht sind ganze Zahlen $u_j, \ldots, u_k$ mit

$$(u_j, \ldots, u_k) \neq (0, \ldots, 0) \quad \text{und } \bar{F}_j := F_j(\sum_{i=j}^{k} u_i b_i) = \lambda_{1,F_j}(L(b_j, \ldots, b_k)).$$

Wir realisieren diese Prozedur durch eine rekursive Aufzählung aller Koeffizientenvektoren $(\tilde{u}_t, \ldots, \tilde{u}_k) \in \mathbb{Z}^{k-t+1}$, $t = k, \ldots, j$, für die $F_t(\sum_{i=t}^{k} \tilde{u}_i b_i) < F_j(b_j)$ gilt. Wir setzen $(u_j, \ldots, u_k) := (\tilde{u}_j, \ldots, \tilde{u}_k)$ für einen Koeffizientenvektor mit minimalem F_j unter allen aufgezählten, von Null verschiedenen Koeffizientenvektoren. Diese Vorgehensweise ist offensichtlich korrekt, denn es gilt $F_t(x) \leq F_j(x + \sum_{i=j}^{t-1} z_i b_i)$ für alle $x \in \mathbb{R}^n$ und $z_i \in \mathbb{R}$ sowie $\lambda_{1,F_j}(L(b, \ldots, b_k)) \leq F_j(b_j)$.

Bei Eintritt in Stufe $t > j$ der Aufzählung sind die Koeffizienten $\widetilde{u}_k, \ldots, \widetilde{u}_t$ bereits fixiert. Wir durchlaufen für $\widetilde{u}_{t-1}$ alle ganzen Zahlen, für die $F_{t-1}(\sum_{i=t-1}^{k} \widetilde{u}_i b_i) < F_j(b_j)$ gilt und gehen jeweils zur Stufe $t-1$ über. Wegen $F_t(-x) = F_t(x)$ für alle $x \in \mathbb{R}^n$ können wir uns im Fall $\widetilde{u}_k = \ldots = \widetilde{u}_t = 0$ auf nichtnegative ganze Zahlen $\widetilde{u}_{t-1}$ beschränken. Jeder Festlegung eines Koeffizienten $\widetilde{u}_t$ ordnen wir eine Kante in einem Suchbaum der Tiefe $k - j + 2$ zu (siehe Abbildung 2.1). Die Wurzel des Baums identifizieren wir mit Stufe $k + 1$, die Söhne der Knoten auf Stufe $t + 1$ werden mit Stufe t identifiziert. Einen Knoten auf Stufe t, dessen Pfad von der Wurzel aus mit $\widetilde{u}_k, \ldots, \widetilde{u}_t$ markiert ist, nennen wir $(\widetilde{u}_t, \ldots, \widetilde{u}_k)$. Für $t > j$ ordnen wir jeder ganzen Zahl $\widetilde{u}_{t-1}$ mit $F_{t-1}(\sum_{i=t-1}^{k} \widetilde{u}_i b_i) < F_j(b_j)$ eine Kante von $(\widetilde{u}_t, \ldots, \widetilde{u}_k)$ zu einem Sohn $(\widetilde{u}_{t-1}, \ldots, \widetilde{u}_k)$ zu, die wir mit $\widetilde{u}_{t-1}$ markieren. Für mindestens ein Blatt $(\widetilde{u}_j, \ldots, \widetilde{u}_k)$ auf Stufe $t = j$ gilt $F_j(\sum_{i=j}^{k} \widetilde{u}_i b_i) = \lambda_{1,F_j}(L(b, \ldots, b_k))$. Wir geben $(u_j, \ldots, u_k) := (\widetilde{u}_j, \ldots, \widetilde{u}_k)$ und $\bar{F}_j := F_j(\sum_{i=j}^{k} \widetilde{u}_i b_i)$ aus.

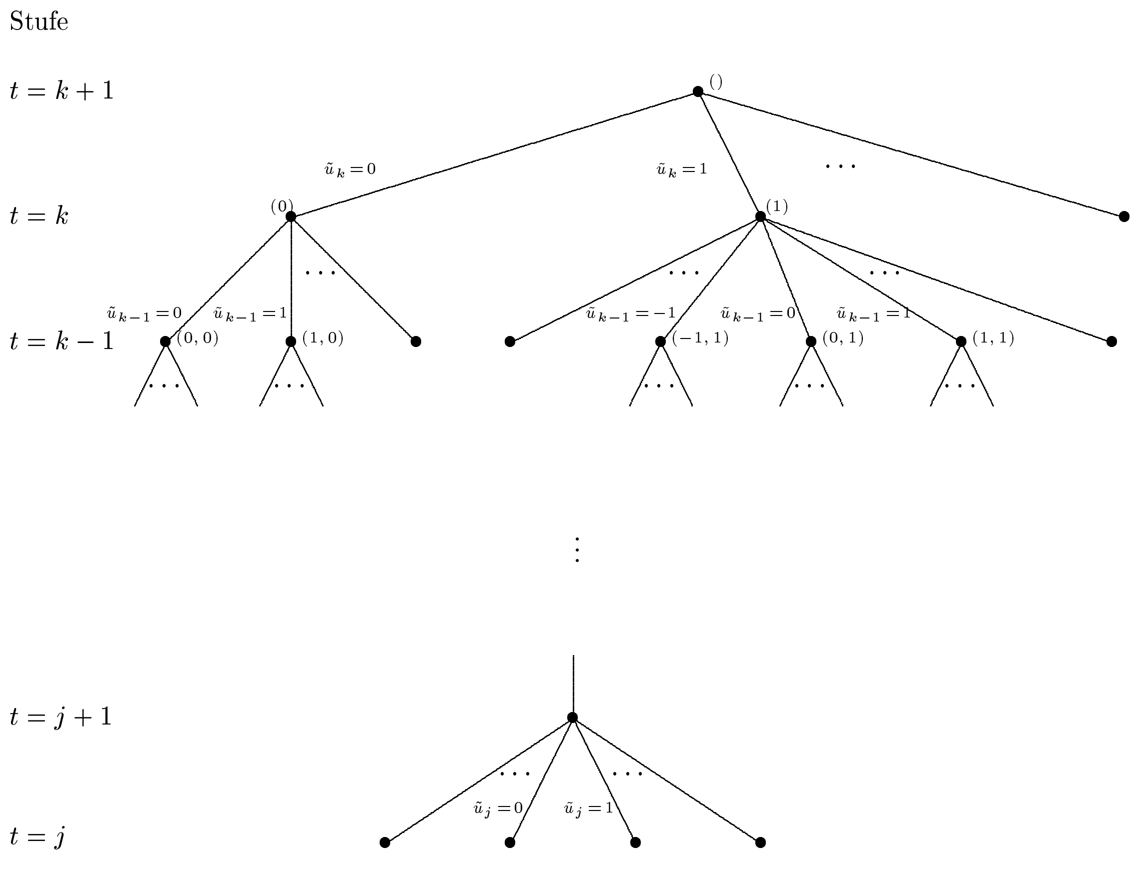

Abbildung 2.1 Baumstruktur des Aufzählungsverfahrens

Wir organisieren die Aufzählung durch Erstellen von Listen, die alle Wurzeln von Teilbäumen enthalten, die noch nicht abgearbeitet wurden.

Ein generelles Aufzählungskonzept
EINGABE: $j, \quad k, \quad b_1, \ldots, b_k$
1. Initialisierung:
   $(u_j, \ldots, u_k) := (1, 0, \ldots, 0)$
   $U_i := \{(\widetilde{u}_i, 0, \ldots, 0) \in \mathbb{N} \times \{0\}^{k-i} : F_i(\widetilde{u}_i b_i) < F_j(b_j)\}$ für $j+1 \leq i \leq k$.
2. Solange noch nicht–abgearbeitete Listen vorhanden sind:
   entferne ein Element $(\widetilde{u}_t, \ldots, \widetilde{u}_k)$ aus einer Liste U_t.
   Im Fall $t > j$ füge alle Knoten $(\widetilde{u}_{t-1}, \ldots, \widetilde{u}_k)$ mit $\widetilde{u}_{t-1} \in \mathbb{Z}$ zur Liste U_{t-1} hinzu,
   für die $F_{t-1}(\sum_{i=t-1}^{k} \widetilde{u}_i b_i) < F_j(\sum_{i=j}^{k} u_i b_i)$ gilt.
   Im Fall $t = j$, $(\widetilde{u}_j, \ldots, \widetilde{u}_k) \neq (0, \ldots, 0)$ und $F_j(\sum_{i=j}^{k} \widetilde{u}_i b_i) < F_j(\sum_{i=j}^{k} u_i b_i)$
   setze $(u_j, \ldots, u_k) := (\widetilde{u}_j, \ldots, \widetilde{u}_k)$.

AUSGABE: $(u_j, \ldots, u_k), \ F_j(\sum_{i=j}^{k} u_i b_i)$

Das Ziel einer optimalen Ausgestaltung dieses „Branch and Bound"–Konzepts besteht darin, den Gesamtaufwand für die Abarbeitung aller auftretenden Listen zu minimieren. Dies kann zum einen dadurch versucht werden, die Gesamtzahl der auftretenden Listeneinträge zu minimieren, zum anderen durch die Minimierung des Aufwands pro Listeneintrag. Diese beiden Ansätze sind im allgemeinen gegenläufig, d.h. zur Verkleinerung der Gesamtzahl der Listeneinträge ist in der Regel ein größerer Aufwand pro Listeneintrag erforderlich und umgekehrt. Im folgenden werden einige Optimierungsansätze diskutiert.

Die Vorgehensweise „Breadth First Search" besteht darin, in Schritt 2 stets das maximale t zu wählen, für das die Liste U_t nicht leer ist. Nachteilig bei dieser Vorgehensweise wirkt sich der große Speicherbedarf aus. Andererseits treten viele gleichartige Rechenschritte auf, die parallel abgearbeitet werden können. Damit bietet sich „Breadth First Search" für Parallel– bzw. Vektorrechner an.

Bei der Vorgehensweise „Depth First Search" wird in Schritt 2 stets das minimale t gewählt, für das die Liste U_t nicht leer ist. Diese Methode ist für sequentielle Rechner mit ihrem in der Regel kleineren Hauptspeicher vorteilhafter, denn die Listen müssen in diesem Fall nicht explizit erzeugt und abgearbeitet werden.

2.4 Aufzählungsverfahren in Depth–First–Search–Ordnung

Das folgende Aufzählungsverfahren ADFS ist universell für jede beliebige Norm verwendbar und liefert stets die Darstellung eines bezüglich der Höhenfunktion F_j kürzesten Vektors im Gitter $L(b_j, \ldots, b_k)$. Die Anzahl der Iterationen von Schritt 2 ist für $(2, \Delta)$–blockreduzierte Eingaben beschränkt durch $(\Delta - \frac{1}{2})^{-O((k-j+1)^2)}$ (siehe hierzu Kapitel 4.2.1).

Wir betrachten für festes $(\widetilde{u}_{t+1},\dots,\widetilde{u}_k)$ die $\widetilde{u}_t$ in der Reihenfolge ansteigender $F_t(\sum_{i=t}^k \widetilde{u}_i b_i)$. Hierzu bestimmen wir zunächst eine reelle Minimalstelle z_t für $F_t(\sum_{i=t+1}^k \widetilde{u}_i b_i + z b_t)$. Die ganzzahlige Minimalstelle ist dann wegen der Konvexität von F_t entweder $l_t = \lfloor z_t \rfloor$ oder $r_t = \lceil z_t \rceil$. Ist $F_t(\sum_{i=t+1}^k \widetilde{u}_i b_i + l_t b_t) \geq F_j(\sum_{i=j}^k u_i b_i)$, dann gilt dies auch für alle $\widetilde{u}_t < l_t$ und wir können die Aufzählung nach links abbrechen, d.h. die Teilbäume mit Wurzeln $\widetilde{u}_t \leq l_t$ müssen nicht durchlaufen werden. Analog können wir die Aufzählung nach rechts abbrechen, falls $F_t(\sum_{i=t+1}^k \widetilde{u}_i b_i + r_t b_t) \geq F_j(\sum_{i=j}^k u_i b_i)$ ist. Mit $\bar{F}_j := F_j(\sum_{i=j}^k u_i b_i)$ bezeichnen wir das aktuelle Minimum der Höhenfunktion unter allen bisher durchlaufenen Blättern. Zu Beginn setzen wir $\bar{F}_j := F_j(b_j)$, $(\widetilde{u}_j,\dots,\widetilde{u}_k) := (u_j,\dots,u_k) := (1,0,\dots,0)$. Dadurch wird das Minimum von F_j in den Teilgittern $L(b_j,\dots,b_s)$ mit sukzessive wachsendem s gesucht. Außerdem beschränken wir uns auf den Fall $\widetilde{u}_s > 0$, d.h. der Vektor b_s geht stets mit positivem Vorzeichen in die Suche ein. Damit werden Redundanzen in der Aufzählung vermieden.

Algorithmus ADFS

EINGABE: $j,\ k,\ b_1,\dots,b_k$

1. Initialisierung:
   $(u_j,\dots,u_k) := (1,0,\dots,0),\ (\widetilde{u}_j,\dots,\widetilde{u}_k) := (1,0,\dots,0),\ \bar{F}_j := F_j(b_j),\ s := t := j$
2. WHILE $t \leq k$

         IF $F_t(\sum_{i=t}^s \widetilde{u}_i b_i) < \bar{F}_j$
         THEN IF $t > j$
               THEN $t := t-1$
                     bestimme $z_t \in \mathbb{R}$ mit $F_t(z_t b_t + \sum_{i=t+1}^s \widetilde{u}_i b_i) = F_{t+1}(\sum_{i=t+1}^s \widetilde{u}_i b_i)$
                     $l_t := \lfloor z_t \rfloor,\ r_t := l_t + 1$
                     IF $F_t(l_t b_t + \sum_{i=t+1}^s \widetilde{u}_i b_i) < F_t(r_t b_t + \sum_{i=t+1}^s \widetilde{u}_i b_i)$
                     THEN $\widetilde{u}_t := l_t,\ l_t := l_t - 1$
                     ELSE $\widetilde{u}_t := r_t,\ r_t := r_t + 1$
               ELSE $(u_j,\dots,u_k) := (\widetilde{u}_j,\dots,\widetilde{u}_k),\ \bar{F}_j := F_j(\sum_{i=j}^k u_i b_i)$
         ELSE $t := t+1$
              IF $t \geq s$
              THEN $\widetilde{u}_t := \widetilde{u}_t + 1,\ s := t$
              ELSE IF $F_t(l_t b_t + \sum_{i=t+1}^s \widetilde{u}_i b_i) < F_t(r_t b_t + \sum_{i=t+1}^s \widetilde{u}_i b_i)$
                    THEN $\widetilde{u}_t := l_t,\ l_t := l_t - 1$
                    ELSE $\widetilde{u}_t := r_t,\ r_t := r_t + 1$

   END while

AUSGABE: $(u_j,\dots,u_k),\ F_j(\sum_{i=j}^k u_i b_i)$

Bei der Verwendung des Aufzählungsverfahrens als Unterprogramm für die (β,Δ)–Blockreduktion mit $\Delta < 1$ sind wir nur an Vektoren $b_j^{\text{neu}} = \sum_{i=j}^k u_i b_i$ mit $F_j(b_j^{\text{neu}}) < \Delta F_j(b_j)$ interessiert. In diesem Fall können wir daher $\bar{F}_j$ mit $\Delta F_j(b_j)$ initialisieren und damit den

Aufwand für die Aufzählung verringern.

2.5 Berechnung der Höhenfunktionen

Entscheidend für die Effizienz des Aufzählungsverfahrens ADFS ist neben der Zahl der Iterationen auch die Effizienz der Berechnung von F_t und der zugehörigen Koeffizienten. Für allgemeine Normen ist hierfür ein (nichtlineares) Optimierungsproblem in $t-1$ reellen Variablen zu lösen. Im Fall der l_∞–Norm erhalten wir ein lineares Optimierungsproblem, das für rationale Eingaben in polynomialer Zeit lösbar ist (siehe Kapitel 2.5.2). Für l_p–Normen können wir F_t mit Hilfe der Ellipsoidmethode in polynomialer Zeit gut approximieren (siehe Kapitel 2.5.3). Für Ellipsoidnormen und insbesondere für die euklidische Norm sind die Minimalstellen der Höhenfunktionen eindeutig bestimmt und lassen sich effizient bestimmen. Das Aufzählungsverfahren ist daher in diesem Fall besonders effizient. In Kapitel 3 modifizieren wir mit Hilfe der Hölderschen Ungleichung das l_2–Norm–Aufzählungsverfahren so, daß damit auch l_p–Norm–kürzeste Gittervektoren effizient bestimmt werden können.

Für die euklidische Norm gilt:

$$\begin{aligned} F_i(x) &= \|\pi_i(x)\|_2 = \left\| x - \sum_{j=1}^{i-1} \frac{<x, \hat{b}_j>}{<\hat{b}_j, \hat{b}_j>} \right\|_2 && (2.25) \\ &= \left\| x + \sum_{j=1}^{i-1} z_j b_j \right\|_2 \quad \text{mit } z_j = -\frac{<x, \hat{b}_j>}{<\hat{b}_j, \hat{b}_j>} + \sum_{t=j+1}^{i-1} z_t \frac{<x, \hat{b}_t>}{<\hat{b}_t, \hat{b}_t>} \text{ für } j = i-1, \ldots, 1, \\ F_i^2(x) &= \|x\|_2^2 - \sum_{j=1}^{i-1} \left(\frac{<x, \hat{b}_j>}{<\hat{b}_j, \hat{b}_j>} \right)^2 \|\hat{b}_j\|_2^2 = F_{i+1}^2(x) + \left(\frac{<x, \hat{b}_i>}{<\hat{b}_i, \hat{b}_i>} \right)^2 \|\hat{b}_i\|_2^2. && (2.26) \end{aligned}$$

Für allgemeine Normen ist die Minimalstelle $(z_1, \ldots, z_{i-1})$ von $\|b_i + \sum_{j=1}^{i-1} \xi_j b_j\|$ nicht immer eindeutig bestimmt. Wir betrachten hierzu ein Beispiel in der l_∞–Norm auf $\mathbb{R}^2$:

Sei $b_2 = (0, 1)$. Für $b_1 = (1, 1)$ ist $F_2(b_2) = \|b_2 - \frac{1}{2} b_1\|_\infty = \frac{1}{2}$ und $\xi_1 = -\frac{1}{2}$ ist die eindeutige Minimalstelle der Höhenfunktion. Für $b_1 = (1, 0)$ gilt $F_2(b_2) = \|b_2 + \xi_1 b_1\|_\infty = 1$ für jedes $\xi_1 \in [-1, 1]$.

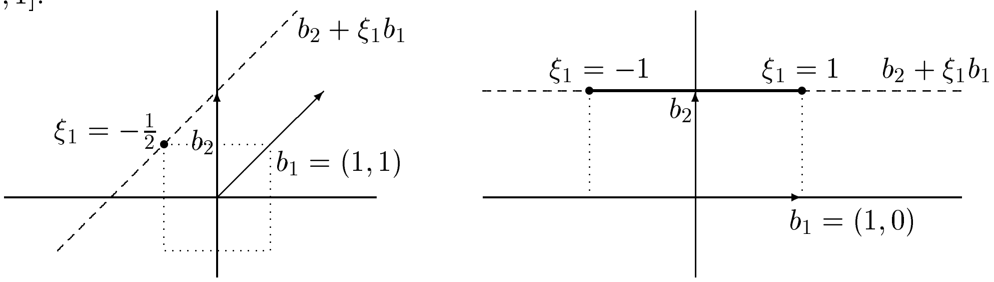

Abbildung 2.2 Minimalstellen der Höhenfunktion in l_∞–Norm

2.5.1 Berechnung von F_2

Für beliebige Norm $\|.\|$ ist zur Berechnung der Höhenfunktion $F_2(x) = \min_{\xi \in \mathbb{R}} \|x + \xi b_1\|$ eine Minimalstelle z_1 der konvexen Funktion $f_x(z) := \|x + zb_1\|$ zu bestimmen. Zu diesem Zweck bietet sich eine Intervallschachtelung an.
Offensichtlich ist $O_2 := \|x\|$ eine obere Schranke für $F_2(x)$. Aus der Äquivalenz der Normen auf $\mathbb{R}^n$ folgt die untere Schranke $U_2 := \|x\|_2 / R(\|.\|)$. Für jedes $z_1 \in \mathbb{R}$ mit $f_x(z_1) = F_2(x)$ gilt $|z_1| \leq 2\|x\|/\|b_1\|$ aufgrund der Dreiecksungleichung.
Der folgende Algorithmus F2 gibt für beliebige Eingabevektoren $x, b_1 \in \mathbb{R}^n$ und jedes $\epsilon > 0$ nach $O(n \log \frac{R(\|.\|)}{\epsilon r(\|.\|)})$ arithmetischen Operationen und $O(\log \frac{R(\|.\|)}{\epsilon r(\|.\|)})$ Berechnungen der Norm ein Paar z, O_2 aus mit $F_2(x) \leq O_2 := \|x + zb_1\| \leq (1+\epsilon) F_2(x)$.

Algorithmus F2
EINGABE: $x, b_1 \in \mathbb{R}^n,\ \epsilon > 0$
1. $O_2 := \|x\|,\ U_2 := \frac{\|x\|_2}{R(\|.\|)},\ z := 0,\ L := -2\frac{\|x\|}{\|b_1\|},\ R := 2\frac{\|x\|}{\|b_1\|}$
2. WHILE $(R - L > \frac{\epsilon}{\|b_1\|} U_2)$ AND $(O_2 > (1+\epsilon) U_2)$
   $t_i := L + \frac{i}{4}(R - L)$ für $i = 0, \ldots, 4$
   bestimme i_0 so, daß $\|x + t_{i_0} b_1\| = \min_{i=0,\ldots,4} \|x + t_i b_1\|$
   IF $\|x + t_{i_0} b_1\| < O_2$ THEN $O_2 := \|x + t_{i_0} b_1\|,\ z := t_{i_0}$
   $L := t_{\max(i_0 - 1, 0)},\ R := t_{\min(i_0 + 1, 4)}$

AUSGABE: z, O_2

Nach jedem Durchlauf von Schritt 2 gilt $L \leq z \leq R$ und $\|x + Lb_1\| \geq \|x + zb_1\| \leq \|x + Rb_1\|$. Wegen der Konvexität der Norm existiert daher stets ein $z_1 \in [L, R]$ mit $f_x(z_1) = F_2(x)$. Am Ende des Verfahrens ist also entweder $|z - z_1| \leq \frac{\epsilon}{\|b_1\|} U_2$ oder $O_2 \leq (1+\epsilon) U_2$ und wir erhalten jeweils $F_2(x) \leq O_2 \leq (1+\epsilon) F_2(x)$.

Bei jedem Durchlauf von Schritt 2 wird die Intervallänge entweder halbiert oder geviertelt. Damit ist die Anzahl der Iterationen durch $1 + \log_2(\frac{R^{\text{Start}} - L^{\text{Start}}}{R^{\text{Ende}} - L^{\text{Ende}}}) \leq 5 + \log_2(\frac{R(\|.\|)}{\epsilon r(\|.\|)})$ beschränkt. Für alle l_p–Normen gilt $R(\|.\|_p)/r(\|.\|_p) \leq \sqrt{n}$, d.h. der Algorithmus ist in diesem Fall polynomial in n und ϵ^{-1}.

Für die l_1– resp. l_∞–Norm gibt Kaib [Kai94] Algorithmen an, die $F_2(x)$ zusammen mit einer ganzzahligen Minimalstelle ξ_1 von $\|x + \xi b_1\|$ in $O(n)$ resp. $O(n \log n)$ arithmetischen Operationen bestimmen.

2.5.2 Berechnung der Höhenfunktionen in der l_∞–Norm

Im Fall der l_∞–Norm ergibt sich ein lineares Optimierungsproblem:
Gegeben seien $b_1, \ldots, b_m$ und $x \in \mathbb{R}^n$.
Zu berechnen ist $F_i(x) = \min\limits_{z_j \in \mathbb{R}} \|x + \sum\limits_{j=1}^{i-1} z_j b_j\|_\infty$.

Mit $z_i := \|x + \sum\limits_{j=1}^{i-1} z_j b_j\|_\infty$ müssen folgende Ungleichungen erfüllt sein:

$$\begin{aligned} z_i &\geq 0, \\ -z_i &\leq x_k + \sum_{j=1}^{i-1} z_j b_{j,k} \leq z_i \quad \text{für alle} \quad k \in \{1, \ldots, n\}. \end{aligned} \tag{2.27}$$

Außerdem ist z_i zu minimieren.

Wir erhalten das folgende lineare Optimierungsproblem:

$$\begin{aligned} z_i &\geq 0, \\ z_i &\leq \|x\|_\infty, \\ \sum_{j=1}^{i-1} z_j b_{j,k} - z_i &\leq -x_k, \\ -\sum_{j=1}^{i-1} z_j b_{j,k} - z_i &\leq x_k \quad \text{für alle} \quad k \in \{1, \ldots, n\}, \\ \text{Zielfunktion: } Q(x) &= z_i \text{ (zu minimieren).} \end{aligned} \tag{2.28}$$

Dieses Optimierungsproblem ist für rationale Eingaben in polynomialer Zeit (in der Bitlänge der Eingabe) mittels Ellipsoid–Methode [PS82] oder Karmarkars Algorithmus [Kar84] lösbar. In der Praxis ist das Simplexverfahren [BS91] effizienter.

2.5.3 Berechnung der Höhenfunktionen für l_p–Normen

In Verbindung mit der Hölderschen Ungleichung liefert die Ellipsoid–Methode ein Verfahren, mit dem wir F_i für beliebige l_p–Normen in polynomialer Zeit beliebig genau approximieren können. Der Kern der Ellipsoid–Methode besteht darin, die Lösungen eines gegebenen Problems in einem immer kleiner werdenden Ellipsoid einzuschließen. In jeder Iteration wird eine Hyperebene durch den Mittelpunkt des Ellipsoids konstruiert, so daß die Lösungsmenge ganz in einer der beiden Hälften des Ellipsoids liegt. Danach wird ein neues Ellipsoid konstruiert, das die noch zulässige Hälfte umfaßt und ein um einen konstanten Faktor kleineres Volumen bestitzt. Wir konstruieren die Hyperebene so, daß die l_p–Norm auf der Hyperebene im Mittelpunkt des Ellipsoids minimiert wird. Dadurch können wir die Hälfte des Ellipsoids ausschließen, die durch die Hyperebene vom Ursprung getrennt wird. Die l_p–Norm der Mittelpunkte konvergiert dann gegen den Wert der Höhenfunktion.

Höldersche Ungleichung: Sei $p \geq 1$, $q = \infty(1)$ für $p = 1(\infty)$ und $q = \frac{p}{p-1}$ sonst. Dann gilt für alle $x, y \in \mathbb{R}^n$

$$|<x,y>| \leq \|x\|_p \|y\|_q. \tag{2.29}$$

Satz 2.11 *Sei $p \geq 1$. Zu jedem $x \in \mathbb{R}^n$ läßt sich in $O(n)$ arithmetischen Operationen eine $(n-1)$–dimensionale Hyperebene $H \subset \mathbb{R}^n$ mit $x \in H$ und $\|x\|_p = \min_{z \in H} \|z\|_p$ konstruieren.*

Beweis von Satz 2.11:
Für $x = 0$ ist die Behauptung mit jeder Hyperebene durch den Ursprung richtig. Wir betrachten den Fall $x \neq 0$ und konstruieren einen Vektor $y \in \mathbb{R}^n$ mit $x - y \perp y$ und $\|y\|_2^2 = \|x\|_p \|y\|_q$. Hierbei ist $q = \infty(1)$ für $p = 1(\infty)$ und $q = \frac{p}{p-1}$ sonst.

Mit $H := y + \text{span}(y)^\perp$ folgt $x \in H$ und aufgrund der Hölderschen Ungleichung gilt $\|y\|_2^2 = <z,y> \leq \|z\|_p \|y\|_q$ für alle $z \in H$, d.h. $\|x\|_p = \min_{z \in H} \|z\|_p$.

Im Fall $p = 1$ setzen wir $y_i := \frac{1}{n} \|x\|_1 \text{sign}(x_i)$ für $1 \leq i \leq n$.
Im Fall $p = \infty$ setzen wir $y_i := x_i$ für $i = \min\{j : |x_j| = \|x\|_\infty\}$ und $y_i := 0$ sonst.
Im Fall $1 < p < \infty$ setzen wir $y_i := \frac{\|x\|_p^p}{\sum_{j=1}^n |x_j|^{2/(q-1)}} |x_i|^{1/(q-1)} \text{sign}(x_i)$ für $1 \leq i \leq n$.

Die geforderten Eigenschaften lassen sich leicht nachrechnen. In jedem der 3 Fälle genügen $O(n)$ arithmetische Operationen zur Bestimmung des Vektors y und damit zur Konstruktion der gesuchten Hyperebene. □

Kapitel 3

Vollständige Aufzählung in l_p–Normen

3.1 Aufzählung in der euklidischen Norm

Im Algorithmus ADFS werden für festes $(\widetilde{u}_{t+1}, \ldots, \widetilde{u}_k)$ die $\widetilde{u}_t$ in der Reihenfolge ansteigender $F_t(\sum_{i=t}^k \widetilde{u}_i b_i)$ durchlaufen. Wie wir bereits in Kapitel 2.5 gesehen haben, läßt sich in der euklidischen Norm sowohl die reelle Minimalstelle z_t als auch das Minimum $F_t(z_t b_t + \sum_{i=t}^k \widetilde{u}_i b_i)$ mit Hilfe des Gram–Schmidt–Orthogonalsystems effizient bestimmen. Mit den Bezeichnungen

$$\begin{aligned} \widetilde{c}_t &:= \left\| \pi_t (\sum_{i=t}^k \widetilde{u}_i b_i) \right\|_2^2 \quad \text{und} \\ c_t &:= \|\hat{b}_t\|_2^2 = \|\pi_t(b_t)\|_2^2 \quad \text{für } 1 \leq t \leq k \end{aligned}$$

erhalten wir

$$F_t^2 \Big(\sum_{i=t}^k \widetilde{u}_i b_i \Big) = \widetilde{c}_t = \widetilde{c}_{t+1} + \Big(\widetilde{u}_t + \sum_{i=t+1}^k \widetilde{u}_i \mu_{i,t} \Big)^2 c_t.$$

Mit $y_t := \sum_{i=t+1}^k \widetilde{u}_i \mu_{i,t}$ ist $-y_t$ die (eindeutig bestimmte) reelle Minimalstelle. Das ganzzahlige Minimum erhalten wir aufgrund der Symmetrie

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

an der Stelle $v_t := \lceil -y_t \rfloor$. Je nachdem, ob $v_t > -y_t$ ist oder nicht, liefert die Wahl von $\widetilde{u}_t$ in der Reihenfolge $v_t, v_t - 1, v_t + 1, v_t - 2, \ldots$ oder $v_t, v_t + 1, v_t - 1, v_t + 2, \ldots$ eine Folge nicht abnehmender Werte für $\widetilde{c}_t$, die sich effizient aus $\widetilde{c}_{t+1}$, $\widetilde{u}_t$, y_t und c_t berechnen lassen. Anstelle der Werte l_t und r_t führen wir nun die Information mit, welche der beiden Folgen wir durchlaufen (in der Variablen δ_t) sowie die aktuelle Differenz $\widetilde{u}_t - v_t$ (in der Variablen Δ_t). Damit ergibt sich für die euklidische Norm das folgende effiziente Aufzählungsverfahren von Schnorr und Euchner [SE94]:

Algorithmus ENUM

EINGABE: j, k, c_i für $i = j, \ldots, k$ und
$\mu_{i,t}$ für $j \leq t < i \leq k$

1. $s := t := j,\ \bar{c}_j := c_j,\ \widetilde{u}_j := u_j := 1,$
   $v_j := y_j := \triangle_j := 0,\ \delta_j := 1,$
   FOR $i = j+1, \ldots, k+1$
   $\widetilde{c}_i := u_i := \widetilde{u}_i := v_i := y_i := \triangle_i := 0,\ \delta_i := 1$
2. WHILE $t \leq k$
   $\widetilde{c}_t := \widetilde{c}_{t+1} + (y_t + \widetilde{u}_t)^2 c_t$
   IF $\widetilde{c}_t < \bar{c}_j$
   THEN IF $t > j$
   THEN $t := t-1,\ y_t := \sum_{i=t+1}^{s} \widetilde{u}_i \mu_{i,t},$
   $\widetilde{u}_t := v_t := \lceil -y_t \rfloor,\ \triangle_t := 0$
   IF $\widetilde{u}_t > -y_t$
   THEN $\delta_t := -1$
   ELSE $\delta_t := 1$
   ELSE $\bar{c}_j := \widetilde{c}_j,\ u_i := \widetilde{u}_i$ für $i = j, \ldots, k$
   ELSE $t := t+1$
   $s := \max(s, t)$
   IF $t < s$ THEN $\triangle_t := -\triangle_t$
   IF $\triangle_t \delta_t \geq 0$ THEN $\triangle_t := \triangle_t + \delta_t$
   $\widetilde{u}_t := v_t + \triangle_t$
   END while

AUSGABE: die Minimalstelle $(u_j, \ldots, u_k) \in \mathbb{Z}^{k-j+1} - 0^{k-j+1}$
und das Minimum $\bar{c}_j$ von $c_j(u_j, \ldots, u_k)$

Mit diesem Verfahren können wir — nach geeigneter Vorreduktion der Gitterbasis — effizient einen kürzesten Vektor bezüglich der euklidischen Norm in ganzzahligen Gittern bis etwa zur Dimension 50 bestimmen, indem wir $j = 1, k = m$ setzen. Als Vorreduktion genügt eine (20,0.99)–Blockreduktion; die Gesamtlaufzeit auf HP–Workstations der Serie 710 (technische Daten siehe Seite 79) beträgt einige Minuten.

Bei der Verwendung des Algorithmus ENUM als Unterprogramm für die (β, δ)–Blockreduktion mit $\delta < 1$ sind wir nur an Vektoren b_j^{neu} mit $\bar{c}_j < \delta c_j$ interessiert. Wir können daher zu Beginn $\bar{c}_j := \delta c_j$ setzen und dadurch den Aufwand für die Aufzählung verringern.

3.1.1 Aufzählungsverfahren für beliebige Norm ohne Berechnung der Höhenfunktionen

Sei $b_1, \dots, b_m \in \mathbb{R}^n$ Basis eines Gitters L, $x \in \mathbb{R}^n$ und $\|.\|$ eine beliebige Norm auf $\mathbb{R}^n$.

Für $2 \le t \le m+1$ existieren $\xi_1, \dots, \xi_{t-1} \in \mathbb{R}$ mit $F_t(x) = \|x + \sum_{i=1}^{t-1} \xi_i b_i\|$. Aus der Äquivalenz der Normen auf $\mathbb{R}^n$ folgt

$$r(\|.\|)\|\pi_t(x)\| \le \|\pi_t(x)\|_2 \le \Big\|x + \sum_{i=1}^{t-1} \xi_i b_i\Big\|_2 \le R(\|.\|)\Big\|x + \sum_{i=1}^{t-1} \xi_i b_i\Big\|.$$

Mit $F_t(x) = F_t(\pi_t(x)) \le \|\pi_t(x)\|$ erhalten wir unmittelbar

$$F_t(x) \le \frac{\|\pi_t(x)\|_2}{r(\|.\|)} \tag{3.1}$$

und

$$F_t(x) \ge \frac{\|\pi_t(x)\|_2}{R(\|.\|)}. \tag{3.2}$$

Geometrisch bedeutet dies, daß die orthogonale Projektion $\pi_t(x)$ des Vektors x auf $\text{span}(b_1, \dots, b_{t-1})^\perp$ stets in der n–dimensionalen Kugel um den Ursprung mit Radius $F_t(x)R(\|.\|)$ liegt:

$$\pi_t(x) \in S_n\Big(0, F_t(x)\, R(\|.\|)\Big). \tag{3.3}$$

Im Algorithmus ADFS schneiden wir einen Teilbaum mit Wurzel $(\widetilde{u}_t \dots, \widetilde{u}_k)$ ab, sobald $F_t(\sum_{i=t}^k \widetilde{u}_i b_i) \ge \bar{F}_j$ gilt. Dies ist gleichbedeutend damit, daß $\sum_{i=t}^k \widetilde{u}_i b_i + \text{span}(b_1, \dots, b_{t-1})$ keinen Punkt im Inneren der Kugel $S_n(0, \bar{F}_j, \|.\|)$ enthält. Die Berechnung von $F_t(\sum_{i=t}^k \widetilde{u}_i b_i)$ ist für allgemeine Normen im Vergleich zur Berechnung von $\|\pi_t(\sum_{i=t}^k \widetilde{u}_i b_i)\|_2$ meist erheblich aufwendiger (siehe Kapitel 2.5). Wir nutzen das Verfahren ENUM in Verbindung mit Ungleichung (3.2), um diese Berechnungen zu umgehen.

Wir initialisieren $\bar{F}_j := F_j(b_j)$, $\bar{c}_j := (\bar{F}_j R(\|.\|))^2$ und ersetzen die Zeile

ELSE $\bar{c}_j := \widetilde{c}_j$, $u_i := \widetilde{u}_i$ für $i = j, \dots, k$

durch

ELSE IF $F_j(\sum_{i=t}^k \widetilde{u}_i b_i) < \bar{F}_j$
  THEN $\bar{F}_j := F_j(\sum_{i=t}^k \widetilde{u}_i b_i)$, $u_i := \widetilde{u}_i$ für $i = j, \dots, k$,
    $\bar{c}_j := (\bar{F}_j R(\|.\|))^2$
  ELSE $\triangle_t := -\triangle_t$
    IF $\triangle_t \delta_t \ge 0$ THEN $\triangle_t := \triangle_t + \delta_t$
    $\widetilde{u}_t := v_t + \triangle_t$

Am Ende des Algorithmus geben wir $\bar{F}_j$ und $(u_j, \dots, u_k)$ aus.

Wir erkaufen den erheblichen Laufzeitgewinn für einen einzelnen Knoten des Suchbaums durch die größere Anzahl der zu durchlaufenden Knoten, denn wir können den Teilbaum mit

Wurzel $(\widetilde{u}_t, \ldots, \widetilde{u}_k)$ erst dann abschneiden, wenn $\pi_t(\sum_{i=t}^k \widetilde{u}_i b_i)$ nicht im Inneren der Kugel $S_n(0, \bar{F}_j\, R(\|.\|))$ liegt. Die Zahl der zusätzlichen Knoten hängt entscheidend von der Wahl der Basis ab. Praktische Tests für die l_∞–Norm mit L^3–reduzierten Gitterbasen ($\delta = 0.99$) bis Dimension 30 haben gezeigt, daß das Verfahren ADFS erheblich langsamer ist als der modifizierte Algorithmus ENUM.

Ungleichung (3.2) stellt einen ersten Zusammenhang zwischen den Höhenfunktionen und der euklidischen Norm der orthogonalen Projektionen her, den wir im folgenden vertiefen. Wir nutzen die Höldersche Ungleichung, um für beliebige l_p–Normen schärfere untere Schranken für die Werte der Höhenfunktionen zu bestimmen. Die verbesserten Schranken führen bei den Verfahren zur Aufzählung kurzer Gittervektoren zu einem erheblichen Effizienzgewinn.

3.2 Effiziente Aufzählungsverfahren zur Bestimmung eines kürzesten Gittervektors bezüglich beliebiger l_p–Norm

Im folgenden sei stets $q = \infty(1)$ für $p = 1(\infty)$ und $q = \frac{p}{p-1}$ für $1 < p < \infty$. Die Höhenfunktionen F_t beziehen sich stets auf die l_p–Norm. Entscheidend für die Beschleunigung der Aufzählung von kurzen Gittervektoren bezüglich beliebiger l_p–Normen ist das folgende

Theorem 3.1 *Seien $b_1, \ldots, b_{n-1} \in \mathbb{R}^n$ linear unabhängig. Dann gilt für alle $x \in \mathbb{R}^n$ und $1 \leq t \leq n$*

$$F_t(x) \geq \Big| \sum_{i=t}^n \lambda_i \|\pi_i(x)\|_2^2 \Big| \Big/ \Big\| \sum_{i=t}^n \lambda_i \pi_i(x) \Big\|_q \tag{3.4}$$

für alle $\lambda_t, \ldots, \lambda_n \in \mathbb{R}$ mit $\sum_{i=t}^n \lambda_i \pi_i(x) \neq 0$.

Beweis von Theorem 3.1:
Wegen $F_t(x) = F_t(\pi_t(x))$ gibt es reelle Zahlen $\xi_1, \ldots, \xi_{t-1}$ mit $F_t(x) = \|\pi_t(x) + \sum_{i=1}^{t-1} \xi_i b_i\|_p$.

Zur Abkürzung schreiben wir $y := \pi_t(x) + \sum_{i=1}^{t-1} \xi_i b_i$.

Für $i \geq t$ gilt $<y, \pi_i(x)> = <\pi_i(x), \pi_i(x)>$ und wir erhalten mit der Hölderschen Ungleichung

$$\begin{aligned} \left| \sum_{i=t}^n \lambda_i \|\pi_i(x)\|_2^2 \right| &= \left| \sum_{i=t}^n \lambda_i <\pi_i(x), \pi_i(x)> \right| = \left| \sum_{i=t}^n \lambda_i <y, \pi_i(x)> \right| = \left| <y, \sum_{i=t}^n \lambda_i \pi_i(x)> \right| \\ &\leq \|y\|_p \Big\| \sum_{i=t}^n \lambda_i \pi_i(x) \Big\|_q = F_t(x) \Big\| \sum_{i=t}^n \lambda_i \pi_i(x) \Big\|_q . \end{aligned}$$

□

Aufgrund der Invarianz der euklidischen Norm gegenüber Drehungen des $\mathbb{R}^n$ um den Ursprung benötigen wir bei der Aufzählung in euklidischer Norm nur die Höhenquadrate der

Vektoren $\sum_{i=t}^{m} \widetilde{u}_i b_i$. Die orthogonalen Vektoren $\pi_t(\sum_{i=t}^{m} \widetilde{u}_i b_i)$ müssen nicht explizit berechnet werden. Dies ändert sich bei der Verwendung einer anderen l_p–Norm, denn diese sind nicht gegen Drehungen invariant.

Wir führen zunächst eine Kurzschreibweise für die orthogonalen Vektoren ein.

Bezeichnung 3.2 $w_t = w_t(\widetilde{u}_t, \ldots, \widetilde{u}_m) := \pi_t(\sum_{i=t}^{m} \widetilde{u}_i b_i)$.

Mit $y_t := \sum_{i=t+1}^{m} \widetilde{u}_i \mu_{i,t}$ gilt $w_t = w_{t+1} + (\widetilde{u}_t + y_t)\hat{b}_t$ und $\|w_t\|_2^2 = \|w_{t+1}\|_2^2 + (\widetilde{u}_t + y_t)^2 c_t$.

Offensichtlich ist $F_t(\sum_{i=t}^{m} \widetilde{u}_i b_i) = F_t(w_t)$.

Wir suchen nach einem Vektor $b \in L = L(b_1, \ldots, b_m) \subset \mathbb{R}^n$ mit $\|b\|_p = \lambda_{1,\|.\|_p}(L)$.

Sei $\bar{b} = \sum_{i=1}^{m} u_i b_i \in L \setminus \{0\}$ der Vektor mit minimaler l_p–Norm unter allen bereits aufgezählten Gittervektoren. Zu Beginn setzen wir $\bar{b} := b_1$, d.h. $(u_1, \ldots, u_m) = (1, 0, \ldots, 0)$. Wir können die Suche nach einem kürzesten Gittervektor im Teilbaum mit Wurzel $(\widetilde{u}_t, \ldots, \widetilde{u}_m)$ abbrechen, sobald $F_t(w_t) \geq \|\bar{b}\|_p$ gilt.

Mit Theorem 3.1 erhalten wir

$$\Big|\sum_{i=t}^{m} \lambda_i \widetilde{c}_i\Big| \Big/ \Big\|\sum_{i=t}^{m} \lambda_i w_i\Big\|_q \geq \|\bar{b}\|_p \text{ für ein } (\lambda_t, \ldots, \lambda_m) \in \mathbb{R}^{m-t+1} \text{ mit } \sum_{i=t}^{m} \lambda_i w_i \neq 0$$
$$\Rightarrow \quad F_t(w_t) \geq \|\bar{b}\|_p. \tag{3.5}$$

Für das Abbruchkriterium (3.5) ist eine Minimalstelle $(\lambda_t, \ldots, \lambda_m)$ der Funktion $f(\mu_t, \ldots, \mu_m) := \|\bar{b}\|_p \|\sum_{i=t}^{m} \mu_i w_i\|_q - \widetilde{c}_t$ mit $\mu_i \in \mathbb{R}$, $\sum_{i=t}^{m} \mu_i \widetilde{c}_i = \widetilde{c}_t$ zu bestimmen. Die Aufzählung kann abgebrochen werden, falls $f(\lambda_t, \ldots, \lambda_m)$ negativ ist. Das Minimum der Funktion ist in polynomialer Laufzeit mit Hilfe der Ellipsoidmethode zu approximieren (im wesentlichen ist $\|\sum_{i=t}^{m} \mu_i w_i\|_q$ auf einer $m-t$–dimensionalen Hyperebene zu minimieren und damit eine $m-t+1$–te Höhenfunktion in der l_q–Norm zu berechnen). Damit ist der Aufwand für die volle Ausnutzung von (3.5) vergleichbar mit dem Aufwand zur Berechnung der Höhenfunktion. Ziel ist es daher, eine optimale Auswahl von Vektoren $(\lambda_t, \ldots, \lambda_m)$ zu treffen, für die der Test durchgeführt wird. Für $(\lambda_t, \ldots, \lambda_m) = (1, 0, \ldots, 0)$ erhalten wir

$$\widetilde{c}_t / \|w_t\|_q \geq \|\bar{b}\|_p \Rightarrow F_t(w_t) \geq \|\bar{b}\|_p. \tag{3.6}$$

(3.6) liefert ein Abbruchkriterium, das wir mit linear vielen arithmetischen Operationen testen können.

Für $(\lambda_t, \ldots, \lambda_m) = (1, -1, 0, \ldots, 0)$ ergibt sich

$$|\widetilde{u}_t + y_t| \geq \|\bar{b}\|_p \|\hat{b}_t\|_q / c_t \Rightarrow F_t(w_t) \geq \|\bar{b}\|_p. \tag{3.7}$$

Damit können wir a priori die Anzahl der möglichen Werte für $\widetilde{u}_t$ bei festem $(\widetilde{u}_{t+1}, \ldots, \widetilde{u}_m)$ beschränken.

Lemma 3.3 *Die Aufzählung in Richtung $\lambda_t(w_t - w_{t+1})$ kann bereits abgebrochen werden, wenn $\|\bar{b}\|_p \|\sum_{i=t}^m \lambda_i w_i\|_q < \sum_{i=t}^m \lambda_i \tilde{c}_i$ gilt für ein $(\lambda_t, \ldots, \lambda_m)$ mit $\lambda_t \neq 0$.*

Beweis von Lemma 3.3: Wir betrachten die Gerade $w_{t+1} + \mathbb{R}\hat{b}_t$. Alle w_t', die in Richtung $\lambda_t(w_t - w_{t+1})$ nach w_t aufzuzählen wären, liegen auf dieser Geraden und sind von der Form $w_t' = \lambda w_t + (1-\lambda) w_{t+1}$ mit $\lambda \mathrm{sign}(\lambda_t) \geq 1$.

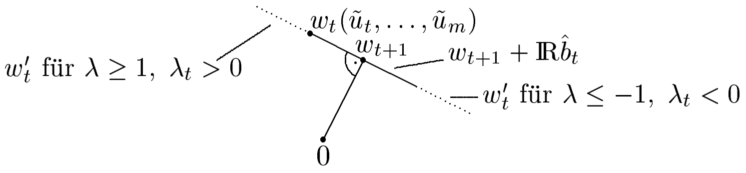

Abbildung 3.1 Abbruch der Aufzählung in Richtung $\lambda_t(w_t - w_{t+1})$

Mit $w_i' := w_i$ für $t+1 \leq i \leq m$, $\lambda_t' := \frac{\lambda_t}{\lambda}$, $\lambda_{t+1}' := \lambda_{t+1} + \lambda_t - \frac{\lambda_t}{\lambda}$ und $\lambda_i' := \lambda_i$ für $t+2 \leq i \leq m$ ist $\sum_{i=t}^m \lambda_i' w_i' = \sum_{i=t}^m \lambda_i w_i$ und wir erhalten

$$\begin{aligned}
\|\bar{b}\|_p \|\textstyle\sum_{i=t}^m \lambda_i' w_i'\|_q &= \|\bar{b}\|_p \|\textstyle\sum_{i=t}^m \lambda_i w_i\|_q \\
&< \textstyle\sum_{i=t}^m \lambda_i \|w_i\|_2^2 \\
&= \lambda_t \|w_t\|_2^2 + \lambda_{t+1} \|w_{t+1}'\|_2^2 + \textstyle\sum_{i=t+2}^m \lambda_i' \|w_i'\|_2^2 \\
&= \lambda \lambda_t' \|w_t\|_2^2 + \frac{1-\lambda}{\lambda} \lambda_t \|w_{t+1}'\|_2^2 + \textstyle\sum_{i=t+1}^m \lambda_i' \|w_i'\|_2^2 \\
&= \lambda_t'((1-\lambda)\tilde{c}_{t+1} + \lambda \tilde{c}_t) + \textstyle\sum_{i=t+1}^m \lambda_i' \|w_i'\|_2^2 \\
&= \lambda_t'(\tilde{c}_{t+1} + \lambda(\tilde{u}_t + y_t)^2 c_t) + \textstyle\sum_{i=t+1}^m \lambda_i' \|w_i'\|_2^2 \\
&= \lambda_t'(\lambda - \lambda^2)(\tilde{u}_t + y_t)^2 c_t + \textstyle\sum_{i=t}^m \lambda_i' \|w_i'\|_2^2 \\
&= \lambda_t(1-\lambda)(\tilde{u}_t + y_t)^2 c_t + \textstyle\sum_{i=t}^m \lambda_i' \|w_i'\|_2^2.
\end{aligned}$$

Wegen $\lambda \mathrm{sign}(\lambda_t) \geq 1$ ist $\lambda_t(1-\lambda) \leq 0$. Es folgt

$$\|\bar{b}\|_p \|\textstyle\sum_{i=t}^m \lambda_i' w_i'\|_q < \textstyle\sum_{i=t}^m \lambda_i' \|w_i'\|_2^2.$$

Damit gilt auch $F_t(w_t') > \|\bar{b}\|_p$, d.h. wir müssen w_t' nicht aufzählen. $\square$

Zur Vereinfachung der Darstellung wird der Test im folgenden Algorithmus auf den Fall $\lambda_t > 0$ und $\sum_{i=t}^s \lambda_i \tilde{c}_i > 0$ eingeschränkt. Damit wird die Aufzählung stets nur in Richtung $w_t - w_{t+1}$ abgebrochen. Ohne diese Einschränkung müßten bei jedem Abbruch aufwendige Fallunterscheidungen durchgeführt werden.

Algorithmus ENUM$_p$

EINGABE: $m, n,$

$c_i,\ b_i,\ \hat{b}_i$ für $i = 1, \ldots, m$

$\mu_{i,t}$ für $1 \le t < i \le m$

1. $s := t := 1,\ \widetilde{u}_1 := u_1 := \eta_1 := \delta_1 := 1,\ \ v_1 := y_1 := \Delta_1 := 0$

   $w_1 := (0, \ldots, 0),\ \ c := \|b_1\|_p$

   FOR $i = 2, \ldots, m+1$

   $\quad \widetilde{c}_i := u_i := \widetilde{u}_i := v_i := y_i := \Delta_i := 0, \eta_i := \delta_i := 1$

   $\quad w_i := (0, \ldots, 0)$

2. WHILE $t \le m$

   $\quad \widetilde{c}_t := \widetilde{c}_{t+1} + (y_t + \widetilde{u}_t)^2 c_t$

   $\quad$ IF $\widetilde{c}_t < (R(\|.\|_p)c)^2$

   $\quad$ THEN $w_t := w_{t+1} + (y_t + \widetilde{u}_t)\hat{b}_t$

   $\qquad$ IF $t > 1$

   $\qquad$ THEN IF SCHNITT$_p(s, w_t, \ldots, w_s, \widetilde{c}_t, \ldots, \widetilde{c}_s, c)$=1

   $\qquad\quad$ THEN IF $\eta_t = 1$ THEN GOTO 3.

   $\qquad\qquad \eta_t := 1\ \Delta_t := -\Delta_t$

   $\qquad\qquad$ IF $\Delta_t \delta_t \ge 0$ THEN $\Delta_t := \Delta_t + \delta_t$

   $\qquad\qquad \widetilde{u}_t := v_t + \Delta_t$

   $\qquad\quad$ ELSE $t := t - 1, \eta_t := \Delta_t := 0$

   $\qquad\qquad y_t := \sum\limits_{i=t+1}^{s} \widetilde{u}_i \mu_{i,t}$

   $\qquad\qquad \widetilde{u}_t := v_t := \lceil -y_t \rfloor$

   $\qquad\qquad$ IF $\widetilde{u}_t > -y_t$

   $\qquad\qquad$ THEN $\delta_t := -1$

   $\qquad\qquad$ ELSE $\delta_t := 1$

   $\qquad$ ELSE IF $\|w_1\|_p < c$

   $\qquad\quad$ THEN $(u_1, \ldots, u_s) := (\widetilde{u}_1, \ldots, \widetilde{u}_s),\ c := \|w_1\|_p$

3. $\quad$ ELSE $t := t + 1$

   $\qquad s := \max(t, s)$

   $\qquad$ IF $\eta_t = 0$

   $\qquad$ THEN $\Delta_t := -\Delta_t$

   $\qquad\quad$ IF $\Delta_t \delta_t \ge 0$ THEN $\Delta_t := \Delta_t + \delta_t$

   $\qquad$ ELSE $\Delta_t := \Delta_t + \delta_t$

   $\qquad \widetilde{u}_t := v_t + \Delta_t$

   END while

AUSGABE: $(u_1, \ldots, u_m),\ c = \left\| \sum\limits_{i=1}^{m} u_i b_i \right\|_p$

Algorithmus SCHNITT$_p$
EINGABE: $s, w_t, \ldots, w_s, \widetilde{c}_t, \ldots, \widetilde{c}_s, c$
Bestimme das Minimum F der Funktion $f(\lambda_t, \ldots, \lambda_s) := c\|\sum_{i=t}^{s} \lambda_i w_i\|_q - \sum_{i=t}^{s} \lambda_i \widetilde{c}_i$ über einer geeigneten Teilmenge der Vektoren $(\lambda_t, \ldots, \lambda_s)$ mit $\lambda_t > 0$ und $\sum_{i=t}^{s} \lambda_i \widetilde{c}_i > 0$.
(In der Praxis ist meist die Beschränkung auf $(\lambda_t, \ldots, \lambda_s) = (1, 0, \ldots, 0)$ am effizientesten.)
AUSGABE: 1, falls $F < 0$ und 0 sonst

Bemerkung 3.4 η_t gibt an, in wievielen Richtungen der Aufzählungsprozeß auf Stufe t bereits abgebrochen wurde. Ist $\eta_t = 1$ und SCHNITT$_p(\ldots) = 1$, dann kann in beiden Richtungen abgebrochen werden, d.h. wir können t um 1 erhöhen. Redundanzen im Aufzählungsprozeß werden vermieden, indem $\widetilde{u}_s$ stets positiv gewählt wird. Damit ist auf Stufe s nur eine Richtung abzuarbeiten; η_s kann bereits mit 1 initialisiert werden.
c ist die l_p–Norm des aktuell kürzesten gefundenen Gittervektors.

Die Anzahl der Knoten, die im Laufe des Algorithmus ENUM$_p$ durchlaufen werden, ist in etwa proportional zum Volumen der Menge, in der die orthogonalen Vektoren w_t liegen müssen, damit nicht abgeschnitten wird (siehe hierzu auch Kapitel 4.3). Ohne Aufruf von SCHNITT$_p$ ist diese Menge die l_2–Norm–Kugel $S_n(0, R(\|.\|_p)\|\bar{b}\|_p)$. Wir zeigen, daß bereits durch die Beschränkung auf $(\lambda_t, \ldots, \lambda_m) = (1, 0, \ldots, 0)$ eine Einschränkung auf die Menge

$$\{x \in \mathbb{R}^n : \|x\|_2^2 \leq \|\bar{b}\|_p \|x\|_q\}$$

erreicht wird. Für die Anwendungen (siehe Kapitel 6) sind die l_∞– und l_1–Norm von besonderer Bedeutung. In diesen beiden Fällen bedeutet dies eine Verringerung des Volumens um einen exponentiellen Faktor (siehe Kapitel 3.3 und 3.4). Der Aufwand für diesen Schnitt ist linear. Wir geben eine geometrische Interpretation von Theorem 3.1 und entwickeln daraus eine alternative Darstellung der Höhenfunktionen.

Seien $b_1, \ldots, b_{n-1} \in \mathbb{R}^n$ linear unabhängige Vektoren, $x \in \mathbb{R}^n$, $1 \leq t \leq n$ und $d > 0$. Wir betrachten die beiden folgenden Mengen im $\mathbb{R}^n$:

$$\begin{aligned} \mathcal{U}_d &:= \bigcup_{\substack{z \in \mathbb{R}^n \\ \|z\|_p \leq d}} \partial S_n\left(\frac{z}{2}, \frac{\|z\|_2}{2}\right) = \bigcup_{\substack{z \in \mathbb{R}^n \\ \|z\|_p = d}} S_n\left(\frac{z}{2}, \frac{\|z\|_2}{2}\right), \\ \mathcal{M}_{t,x} &:= \partial S_n\left(\frac{\pi_t(x)}{2}, \frac{\|\pi_t(x)\|_2}{2}\right) \bigcap \operatorname{span}(b_1, \ldots, b_{t-1})^\perp. \end{aligned}$$

$\mathcal{U}_d$ ist die Vereinigung aller Thaleskugeln zu Punkten im $\mathbb{R}^n$ mit l_p–Norm höchstens d.
$\mathcal{M}_{t,x}$ ist die Thaleskugel zu $\pi_t(x)$ im $\operatorname{span}(b_1, \ldots, b_{t-1})^\perp$.
Wir zeigen, daß $F_t(x) > d$ gilt, falls $\pi_t(x) \notin U_d$ oder $\mathcal{M}_{t,x} \not\subset U_d$.

Theorem 3.5 *Seien $b_1, \ldots, b_{n-1} \in \mathbb{R}^n$ linear unabhängig. Dann gilt für alle $x \in \mathbb{R}^n$ und $1 \leq t \leq n$*

$$F_t(x) = \begin{cases} \max\limits_{z \in \mathcal{M}_{t,x} \setminus \{0\}} \left\{ \frac{\|z\|_2^2}{\|z\|_q} \right\} & \text{, falls } \pi_t(x) \neq 0, \\ 0 & \text{, sonst.} \end{cases} \tag{3.8}$$

Insbesondere ist für $\pi_n(x) \neq 0$ und $\pi_i(x) \neq \pi_j(x)$ für $t \leq i < j \leq n$

$$F_t(x) = \max_{\substack{\lambda_t, \ldots, \lambda_n \in \mathbb{R} \\ \sum_{i=t}^n \lambda_i \pi_i(x) \neq 0}} \left| \sum_{i=t}^n \lambda_i \|\pi_i(x)\|_2^2 \right| \Big/ \Big\| \sum_{i=t}^n \lambda_i \pi_i(x) \Big\|_q. \tag{3.9}$$

Wir zeigen zunächst

$$\mathcal{U}_d = \{y \in \mathbb{R}^n : \|y\|_2^2 \leq d\|y\|_q\}. \tag{3.10}$$

Beweis von (3.10):

„$\subseteq$": Sei $y \in \mathcal{U}_d$. Nach Definition von $\mathcal{U}_d$ existiert ein $z \in \mathbb{R}^n$ mit $y \in \partial S_n\left(\frac{z}{2}, \frac{\|z\|_2}{2}\right)$ und $\|z\|_p \leq d$. Es folgt unmittelbar $y \perp z - y$ und mit der Hölderschen Ungleichung ergibt sich daraus $\|y\|_2^2 = \ <y, y> \ = \ <z, y> \ \leq \|z\|_p \|y\|_q \leq d\|y\|_q$.

„$\supseteq$": Sei $y \in \mathbb{R}^n$ mit $\|y\|_2^2 = c\|y\|_q$, $c \leq d$. Offensichtlich gilt $0 \in \mathcal{U}_d$. Für $y \neq 0$ konstruieren wir einen Vektor $z = (z_1, \ldots, z_n) \in \mathbb{R}^n$ mit $y \perp z - y$ und $\|z\|_p = c$. Daraus folgt unmittelbar die Behauptung.

Wir unterscheiden die Fälle $p = 1$, $p = \infty$ und $1 < p < \infty$:

Im Fall $p = 1$ bestimmen wir $i_0 := \min\{i : |y_i| = \|y\|_\infty\}$ und setzen

$$z_i := \begin{cases} \|y\|_2^2 / y_i & \text{für } i = i_0, \\ 0 & \text{sonst.} \end{cases}$$

Im Fall $p = \infty$ setzen wir $z_i := \operatorname{sign}(y_i)\|y\|_2^2/\|y\|_1$ für $1 \leq i \leq n$.

Im Fall $1 < p < \infty$ setzen wir $z_i := \operatorname{sign}(y_i)|y_i|^{q-1}\|y\|_2^2/\|y\|_q^q$ für $1 \leq i \leq n$.

Die geforderten Eigenschaften lassen sich in jedem der 3 Fälle leicht nachrechnen. □

Aus (3.6) und (3.10) folgt unmittelbar

$$\pi_t(x) \in \mathcal{U}_{F_t(x)}. \tag{3.11}$$

Als nächstes zeigen wir die Inklusion

$$\mathcal{M}_{t,x} \subset \mathcal{U}_{F_t(x)}. \tag{3.12}$$

Sei o.B.d.A. $\pi_n(x) \neq 0$, $\pi_i(x) \neq \pi_j(x)$ für $t \leq i < j \leq n$. (Dies können wir stets durch geeignete Wahl von $b_t, \ldots, b_{n-1}$ erreichen, ohne $\pi_t(x), \mathcal{M}_{t,x}, F_t(x)$ oder $\mathcal{U}_{F_t(x)}$ zu verändern.) Dann ist $\operatorname{span}(\pi_t(x), \ldots, \pi_n(x)) = \operatorname{span}(b_1, \ldots, b_{t-1})^\perp$ und jedes $z \in \mathcal{M}_{t,x}$ läßt sich darstellen in der Form

$$z = \sum_{i=t}^n \lambda_i \pi_i(x) \text{ mit } \lambda_i \in \mathbb{R}.$$

Nach Definition von $\mathcal{M}_{t,x}$ gilt $\|z - \frac{1}{2}\pi_t(x)\|_2 = \frac{1}{2}\|\pi_t(x)\|_2$. Es folgt

$$\sum_{i=t}^{n} \|\pi_i(x)\|_2^2 \Big(\lambda_i^2 + 2\lambda_i \sum_{j=t}^{i-1} \lambda_j\Big) = \sum_{i=t}^{n} \lambda_i \|\pi_i(x)\|_2^2 \tag{3.13}$$

$$\text{und damit} \quad \|z\|_2^2 = \sum_{i=t}^{n} \lambda_i \|\pi_i(x)\|_2^2. \tag{3.14}$$

Mit (3.4) erhalten wir

$$\|z\|_2^2 \leq F_t(x) \Big\| \sum_{i=t}^{n} \lambda_i \pi_i(x) \Big\|_q = F_t(x) \|z\|_q.$$

Mit (3.10) folgt $z \in \mathcal{U}_{F_t(x)}$. □

Beweis von Theorem 3.5:

zu (3.8): Für $\pi_t(x) = 0$ ist die Aussage trivial.

Im Fall $\pi_t(x) \neq 0$ setzen wir zur Abkürzung $c := \max\limits_{z \in \mathcal{M}_{t,x} \setminus \{0\}} \{\|z\|_2^2 / \|z\|_q\}$.

Aus (3.10) und (3.12) folgt unmittelbar

$$F_t(x) \geq c. \tag{3.15}$$

Zu zeigen bleibt also die Ungleichung

$$F_t(x) \leq c. \tag{3.16}$$

Nach Definition von c und (3.10) gilt

$$\begin{aligned} \mathcal{M}_{t,x} &\subset \mathcal{U}_c \cap \operatorname{span}(b_1, \ldots, b_{t-1})^{\perp} \\ &= \bigcup_{\substack{z \in \mathbb{R}^n \\ \|z\|_p \leq c}} \Big(\partial S_n\Big(\frac{z}{2}, \frac{\|z\|_2}{2}\Big)\Big) \cap \operatorname{span}(b_1, \ldots, b_{t-1})^{\perp}. \end{aligned}$$

Zu jeder k–dimensionalen Kugel $K \subset \mathcal{U}_c$ mit $k \leq n$ und $0 \in \partial K$ existiert ein $z \in \mathbb{R}^n$ mit $\|z\|_p \leq c$ und $K \subset S_n(\frac{z}{2}, \frac{\|z\|_2}{2})$.

Daher existiert ein $z \in \mathbb{R}^n$ mit $\|z\|_p \leq c$ und $\mathcal{M}_{t,x} \subset S_n(\frac{z}{2}, \frac{\|z\|_2}{2})$.

Damit ist $z - \pi_t(x) \perp \operatorname{span}(b_1, \ldots, b_{t-1})^{\perp}$, d.h. es existieren $\mu_1, \ldots, \mu_{t-1} \in \mathbb{R}$ mit $z = \pi_t(x) + \sum_{i=1}^{t-1} \mu_i b_i$. Es folgt

$$F_t(x) = \min_{\xi_1, \ldots, \xi_{t-1} \in \mathbb{R}} \Big\| \pi_t(x) + \sum_{i=1}^{t-1} \xi_i b_i \Big\|_p \leq \Big\| \pi_t(x) + \sum_{i=1}^{t-1} \mu_i b_i \Big\|_p = \|z\|_p \leq c.$$

zu (3.9): Nach Voraussetzung läßt sich jedes $z \in \mathcal{M}_{t,x}$ schreiben als $\sum_{i=1}^{t-1} \lambda_i \pi_i(x)$ mit $\lambda_i \in \mathbb{R}$. Wegen (3.14) und (3.8) folgt unmittelbar

$$\begin{aligned} F_t(x) &\geq \max_{\substack{\lambda_t,\ldots,\lambda_n \in \mathbb{R} \\ \sum_{i=t}^n \lambda_i \pi_i(x) \neq 0}} \sum_{i=t}^{n} \lambda_i \|\pi_i(x)\|_2^2 / \| \sum_{i=t}^{n} \lambda_i \pi_i(x) \|_q \\ &\geq \max_{z \in \mathcal{M}_{t,x} \setminus \{0\}} \|z\|_2^2 / \|z\|_q = F_t(x). \end{aligned}$$

□

Korollar 3.6 *Seien $b_1, \ldots, b_{n-1} \in \mathbb{R}^n$ linear unabhängig. Dann gilt für alle $x \in \mathbb{R}^n$*

$$F_t(x)\|\pi_t(x)\|_q \geq \|\pi_t(x)\|_2^2 \quad \text{für } 1 \leq t \leq n, \tag{3.17}$$

$$F_n(x)\|\pi_n(x)\|_q = \|\pi_n(x)\|_2^2. \tag{3.18}$$

Für $t = n$ liefert also die orthogonale Projektion unmittelbar die Höhe bezüglich einer beliebigen l_p–Norm. Für $t < n$ erhalten wir eine untere Schranke.

3.3 Effiziente Aufzählungsverfahren zur Bestimmung eines l_∞–Norm–kürzesten Gittervektors

Wir beschränken uns auf den Fall ganzzahliger Gitter. Damit ist auch die l_∞–Norm des kürzesten Gittervektors ganzzahlig und wir können in Algorithmus ENUM$_p$ wegen $R(\|.\|_\infty) = \sqrt{n}$ die Abfrage „$\tilde{c}_t < (R(\|.\|_\infty)c)^2$" durch die Abfrage „$\tilde{c}_t \leq n(c-1)^2$" ersetzen.

In praktischen Anwendungen (z.B. Lösen von Rucksackproblemen) werden häufig Gittervektoren mit l_∞–Norm 1 gesucht. In diesem Fall kann c mit 2 initialisiert und die Aufzählung abgebrochen werden, sobald ein Vektor mit l_∞–Norm 1 gefunden wurde. Damit ergibt sich ein deterministisches Verfahren, mit dessen Hilfe *alle* Rucksackprobleme bis Dimension 66 effizient gelöst werden können (siehe Kapitel 6.1).

Wir untersuchen den Effizienzgewinn des einfachsten Tests mit $(\lambda_t, \ldots, \lambda_m) = (1, 0, \ldots, 0)$ gegenüber der vollständigen Aufzählung in der euklidischen Norm. Hierzu zeigen wir zunächst das folgende

Lemma 3.7 *i)* $F_t(w_t) \leq d \Rightarrow \exists v \in \{\pm d\}^n$ *mit* $\|w_t - \frac{1}{2}v\|_2^2 \leq \frac{nd^2}{4}$,

ii) $\min_{v \in \{\pm d\}^n} \|w_t - \frac{1}{2}v\|_2^2 \leq \frac{nd^2}{4} \Leftrightarrow \|w_t\|_2^2 \leq d\|w_t\|_1$.

Beweis von Lemma 3.7:

zu $i)$ wegen $F_t(w_t) \leq d$ schneidet die zu w_t orthogonale Hyperebene die l_∞–Norm–Kugel $\{x \in \mathbb{R}^n : \|x\|_\infty \leq d\}$. Daher existiert ein Eckpunkt $v \in \{\pm d\}^n$, für den w_t und $v - w_t$ orthogonal sind oder einen stumpfen Winkel einschließen. w_t liegt also in der „Thaleskugel" zu 0 und v mit Mittelpunkt $v/2$ und Radius $\frac{d}{2}\sqrt{n}$.
Damit ist $\|w_t - \frac{1}{2}v\|_2^2 \leq \|\frac{1}{2}v\|_2^2 = \frac{nd^2}{4}$.

zu $ii)$ $\min\limits_{v\in\{\pm d\}^n} \|w_t - \frac{1}{2}v\|_2^2 = \, <w_t, w_t> - \max\limits_{v\in\{\pm d\}^n} <w_t, v> + \frac{nd^2}{4} = \|w_t\|_2^2 - d\|w_t\|_1 + \frac{nd^2}{4}$. $\square$

Wir suchen also innerhalb der Vereinigung der 2^n Kugeln um $(\pm\frac{d}{2}, \ldots, \pm\frac{d}{2})$ mit Radius $\frac{d}{2}\sqrt{n}$ anstelle einer Kugel um 0 mit Radius $d\sqrt{n}$. In Dimension 2 ergibt sich folgende Figur (die schraffierte Fläche ist die Vereinigung der kleinen Kugeln):

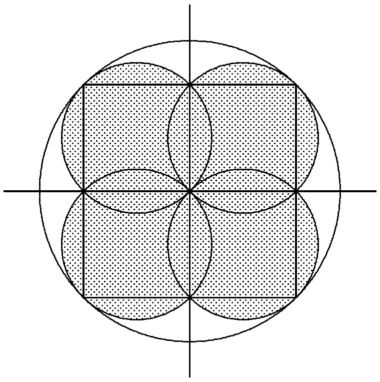

Abbildung 3.2 Aufzählungsbereich in l_∞–Norm

Für das Volumen V_n der Vereinigung der kleinen Kugeln gilt:

$$V_n = (2d)^n \int\limits_0^{\frac{1}{2}+\frac{1}{2}\sqrt{n}} \cdots \int\limits_0^{\frac{1}{2}+\frac{1}{2}\sqrt{n}} \chi(x_1, \ldots, x_n) \;\; dx_n \ldots dx_1.$$

Dabei ist $\chi(x_1, \ldots, x_n) = \begin{cases} 1 & \text{, falls } \sum\limits_{i=1}^{n} x_i^2 \leq \sum\limits_{i=1}^{n} x_i, \\ 0 & \text{, sonst.} \end{cases}$

Durch Variablentransformation erhalten wir hieraus

$$V_n = d^n \int\limits_{-1}^{\sqrt{n}} \int\limits_{\max\left(-1,-\sqrt{n-x_1^2}\right)}^{\sqrt{n-x_1^2}} \cdots \int\limits_{\max\left(-1,-\sqrt{n-x_1^2-\ldots-x_{n-1}^2}\right)}^{\sqrt{n-x_1^2-\ldots-x_{n-1}^2}} 1 \;\; dx_n \cdots dx_1.$$

Damit ist $V_2 = (2+\pi)d^2$ und $V_3 = (2+4\pi)d^3$. Der Volumenanteil an der großen Kugel ist also $\frac{2+\pi}{2\pi}$ für $n = 2$ und $\frac{2+4\pi}{4\sqrt{3}\pi}$ für $n = 3$. Die Auswertung des Integrals erfordert aufwendige Fallunterscheidungen und liefert keine geschlossene Form für beliebige n, so daß eine exakte Berechnung für $n > 3$ nicht durchgeführt wurde. Stattdessen wurden die Volumenanteile mit Hilfe der Monte–Carlo–Methode approximiert, die folgende Werte für die jeweiligen Dimensionen liefert:

n	2	3	4	5	6	7
Volumenanteil	0.8183	0.6691	0.5482	0.4495	0.3687	0.3028
n	8	9	10	15	20	25
Volumenanteil	0.2485	0.2040	0.1676	0.0626	0.0234	0.0088

Tabelle 3.1 Volumenanteil der kleinen Kugeln für die l_∞–Norm

Damit ergibt sich für das Volumen der Vereinigung der kleinen Kugeln etwa das $\left(\frac{2+\pi}{2\pi}\right)^{n-1}$–fache des Volumens der großen Kugel. Der durch den Test entstehende zusätzliche Aufwand für jeden Knoten des Suchbaums ist linear, der Gesamtaufwand für die Aufzählung wird also um einen exponentiellen Faktor verkleinert.

3.4 Effiziente Aufzählungsverfahren zur Bestimmung eines l_1–Norm–kürzesten Gittervektors

Wir beschränken uns wieder auf den Fall ganzzahliger Gitter. Damit ist auch die l_1–Norm des kürzesten Gittervektors ganzzahlig und wir können wegen $R(\|.\|_1) = 1$ die Abfrage „$\widetilde{c}_t < (R(\|.\|_1)c)^2$“ durch die Abfrage „$\widetilde{c}_t \le (c-1)^2$“ ersetzen.

Die Aufzählung bezüglich der l_1–Norm kann zum Beispiel zur Faktorisierung von ganzen Zahlen verwendet werden (siehe hierzu auch Kapitel 6.5).

Wir untersuchen erneut den Effizienzgewinn, den wir durch den einfachsten Abbruchtest mit $(\lambda_t, \ldots, \lambda_m) = (1, 0, \ldots, 0)$ erzielen können.

Lemma 3.8 *Mit* $V := \left\{(v_1, \ldots, v_n) \in \{0, \pm d\}^n | v_i = \pm d \text{ für genau ein } i \in \{1, \ldots, n\}\right\}$ *gilt*

i) $F_t(w_t) \le d \Rightarrow \exists v \in V$ *mit* $\|w_t - \frac{1}{2}v\|_2^2 \le \frac{d^2}{4}$,

ii) $\min\limits_{v \in V} \|w_t - \frac{1}{2}v\|_2^2 \le \frac{d^2}{4} \Leftrightarrow \|w_t\|_2^2 \le d\|w_t\|_\infty$.

Beweis von Lemma 3.8:

zu $i)$ wegen $F_t(w_t) \leq d$ schneidet die zu w_t orthogonale Hyperebene die l_1–Norm–Kugel $\{x \in \mathbb{R}^n : \|x\|_1 \leq d\}$. Daher existiert ein Eckpunkt $v \in V$, für den w_t und $v - w_t$ orthogonal sind oder einen stumpfen Winkel einschließen. w_t liegt also in der „Thaleskugel“ zu 0 und v.
Damit ist $\|w_t - \frac{1}{2}v\|_2^2 \leq \|\frac{1}{2}v\|_2^2 = \frac{d^2}{4}$.

zu $ii)$ $\min\limits_{v \in V} \|w_t - \frac{1}{2}v\|_2^2 = \; <w_t, w_t> \; - \max\limits_{v \in V} \; <w_t, v> \; + \frac{d^2}{4} = \|w_t\|_2^2 - d \cdot \|w_t\|_\infty + \frac{d^2}{4}$. □

Wir suchen also innerhalb der Vereinigung von $2n$ Kugeln mit Radius $\frac{d}{2}$ anstelle einer Kugel mit Radius d. In Dimension 2 ergibt sich folgende Figur (die schraffierte Fläche ist die Vereinigung der kleinen Kugeln):

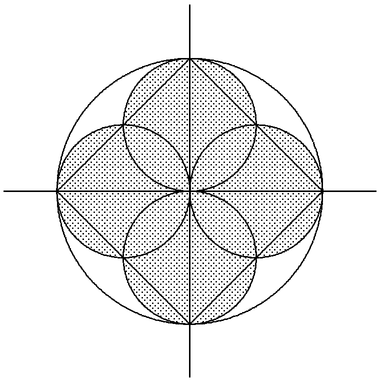

Abbildung 3.3 Aufzählungsbereich in l_1–Norm

Das Volumen V_n der Vereinigung der $2n$ kleinen Kugeln mit halbem Radius ist höchstens das $2n2^{-n} = n2^{1-n}$–fache und mindestens das 2^{-n}–fache des Volumens der großen Kugel. Der durch den Test entstehende zusätzliche Aufwand für jeden Knoten des Suchbaums ist linear, der Gesamtaufwand für die Aufzählung wird also um einen exponentiellen Faktor verkleinert.

Den exakten Wert für das Volumen V_n liefert das Integral

$$V_n = \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} \chi(x_1, \ldots, x_n) dx_n \cdots dx_1 \text{ mit}$$

$$\chi(x_1, \ldots, x_n) = \begin{cases} 1, & \text{falls } \sum_{i=1}^n x_i^2 \leq d \max(x_1, \ldots, x_n) \\ 0, & \text{sonst.} \end{cases}$$

Durch Variablentransformation erhalten wir hieraus

$$V_n = n!(2d)^n \int_0^1 \int_0^{\min\left(x_1,\sqrt{x_1-x_1^2}\right)} \cdots \int_0^{\min\left(x_{n-1},\sqrt{x_1-\sum_{i=1}^{n-1} x_i^2}\right)} 1 \; dx_n \cdots dx_1 .$$

Für $n = 2$ ist der Anteil am Volumen der großen Kugel exakt $\frac{2+\pi}{2\pi}$. Analog zum Fall der l_∞–Norm erfordert auch die Auswertung dieses Integrals aufwendige Fallunterscheidungen und liefert keine geschlossene Form für beliebige n. Die Monte–Carlo–Methode liefert folgende Volumenanteile für die jeweiligen Dimensionen:

n	2	3	4	5	6	7
Volumenanteil	0.8183	0.5991	0.4074	0.2625	0.1627	0.0976
n	8	9	10	15	20	25
Volumenanteil	0.0571	0.0328	0.0185	$8.969 * 10^{-4}$	$3.759 * 10^{-5}$	$1.320 * 10^{-6}$

Tabelle 3.2 Volumenanteil der kleinen Kugeln für die l_1–Norm

Kapitel 4

Laufzeitanalysen

4.1 Laufzeit der Blockreduktion in der euklidischen Norm

Die entscheidende Idee bei der Laufzeitanalyse des L^3–Algorithmus ist die Betrachtung der Determinantenquadrate $D_i := \prod_{j=1}^{i} \|\pi_j(b_j)\|_2^2$ der Teilgitter $L(b_1, \ldots, b_i)$ für $i = 1, \ldots, m-1$ sowie deren Produkt $D := \prod_{i=1}^{m-1} D_i$. Ein Austausch $b_k \leftrightarrow b_{k-1}$ bewirkt $D_{k-1}^{\text{neu}} < \delta D_{k-1}$. Die Werte D_i für $i \neq k-1$ werden durch den Austausch nicht verändert. Nach jedem Austauschschritt gilt daher $D^{\text{neu}} < \delta D$. Die übrigen Schritte des L^3–Algorithmus verändern D nicht. Damit ist die Zahl der Austauschschritte nach oben beschränkt durch $\log_{\frac{1}{\delta}} \frac{D^{\text{Start}}}{D^{\text{Ende}}}$. Mit $M := \max_i \|b_i\|_2^2$ gilt $D^{\text{Start}} \leq M^{\binom{m}{2}}$. Für ganzzahlige Gitter sind alle Determinantenquadrate D_i positive ganze Zahlen und damit $D^{\text{Ende}} \geq 1$. Folglich ist die Anzahl der Austauschschritte des L^3–Algorithmus für ganzzahlige Eingaben durch $\binom{m}{2} \log_{\frac{1}{\delta}} M$ beschränkt. Die Anzahl der Iterationen der Schritte 2–4 ist höchstens $m-1+2*$Anzahl der Austausche. Jede Iteration benötigt nur polynomial viele arithmetische Operationen. Insgesamt benötigt der L^3–Algorithmus also nur polynomial viele arithmetische Operationen.

Im Algorithmus (β, δ)–Blockreduktion bewirkt das Einfügen eines Vektors b_j^{neu} mittels BASIS zwar ebenfalls $D_j^{\text{neu}} < \delta D_j$, aber es werden auch die Größen $D_{j+1}, \ldots, D_{k-1}$ verändert. Damit ist nicht gesichert, daß sich D durch jeden Einfügeschritt erniedrigt. (Es ist nicht einmal gesichert, daß D am Ende der Reduktion minmal ist.) Unter der heuristischen Annahme, daß D „im Mittel" durch das Einfügen von b_j^{neu} mindestens um den Faktor δ abnimmt, erhalten wir eine polynomiale obere Schranke für die Anzahl der Iterationen (L^3–Reduktionen und Aufrufe von ENUM). Wir werden in Kapitel 4.2.2 beweisen, daß der Aufwand für eine Enumeration bei Eingabe einer mit δ L^3–reduzierten Basis durch $(1/(\delta-\frac{1}{4}))^{O(\beta^2)}$ beschränkt ist. Damit läßt sich unter der heuristischen Annahme für jede feste Blockgröße β und jedes $\delta \in (\frac{1}{4}, 1)$ eine polynomiale Laufzeitschranke des Algorithmus (β, δ)–Blockreduktion beweisen. Die Annahme wurde bei vielen praktischen Testläufen bestätigt, ist aber im worst case falsch. Durch einige Modifikationen am Algorithmus ist es möglich, eine polynomiale Laufzeitschranke für $\beta = 3$ und $\frac{1}{2} \leq \delta < \frac{1}{2}\sqrt{3}$ zu beweisen. Im Fall ganzzahliger Eingaben lassen sich analog zum L^3–Algorithmus die Absolutwerte der Zähler und Nenner aller im Verlauf des Verfahrens auftretenden rationalen Zahlen polynomial in der Eingabe beschränken.

4.1.1 Polynomiale Laufzeit der Blockeduktion mit Blockgröße 3

Wir analysieren den folgenden Algorithmus zur $(3, \delta)$–Blockreduktion. Zur Abkürzung schreiben wir wieder $c_j := \|\hat{b}_j\|_2^2 = \|\pi_j(b_j)\|_2^2$.

Algorithmus Block–3–Reduktion
EINGABE: $b_1, \ldots, b_m \in \mathbb{R}^n$, δ mit $\frac{1}{2} \leq \delta < \frac{1}{2}\sqrt{3}$
1. $\delta' := \delta^2 + \frac{1}{4}$ (es gilt $\delta \leq \delta' < 1$)
   L^3–reduziere $b_1, \ldots, b_m$ mit δ'
   $j := m - 2,\ z := 0$
2. WHILE $z < m - 2$
      $j := j + 1$
      IF $j = m - 1$ THEN j := 1
      bestimme $(u_j, u_{j+1}, u_{j+2}) \in \{0, \pm 1\}^2 \times \{1, 2\}$ mit
      $\bar{c}_j := \|\pi_j(\sum_{i=j}^{j+2} u_i b_i)\|_2^2 = \min\limits_{(\tilde{u}_j, \tilde{u}_{j+1}, \tilde{u}_{j+2}) \in \{0,\pm 1\}^2 \times \{1,2\}} \|\pi_j(\sum_{i=j}^{j+2} \tilde{u}_i b_i)\|_2^2$
      $h := \min(j + 3, m)$
      IF $\bar{c}_j < \delta c_j$
      THEN $i_0 := \max\{i : |u_i| = 1, j \leq i \leq j + 2\}$
         $b := \sum_{i=j}^{j+2} u_i b_i$
         entferne den Vektor b_{i_0} aus der Basis und füge den Vektor b
         an die j–te Stelle der Basis ein
         L^3–reduziere $b_1, \ldots, b_h$ mit δ' ab Stufe j
         $z := 0$
      ELSE L^3–reduziere $b_1, \ldots, b_h$ mit δ' ab Stufe $j + 2$
         setze $z := 0$, falls im L^3–Algorithmus zwei Vektoren ausgetauscht wurden
         sonst $z := z + 1$
   END while
AUSGABE: $(3, \delta)$–blockreduzierte Basis $b_1, \ldots, b_m$

Die Variable z zählt die aufeinanderfolgenden Positionen j, an denen die Bedingungen der Blockreduziertheit erfüllt sind. Wird ein Vektor b eingefügt oder während der L^3–Reduktion ein Austausch zweier Basisvektoren vorgenommen, dann kann die Reduktionsbedingung an einer anderen Position wieder verletzt werden. Daher wird z in diesen Fällen auf 0 zurückgesetzt.

Wir zeigen zunächst die Korrektheit des Algorithmus.

Lemma 4.1 *Die Ausgabe des Algorithmus Block–3–Reduktion ist $(3, \delta)$–blockreduziert und zusätzlich L^3–reduziert mit $\delta' = \delta^2 + \frac{1}{4}$.*

Beweis von Lemma 4.1: Nach Schritt 2 ist stets $b_1, \ldots, b_h$ L^3–reduziert mit δ'. h durchläuft zyklisch die $m-3$ Werte $4, 5, \ldots, m$. Am Ende des Algorithmus wird Schritt 2 noch $m-2$ mal ohne Einfügeschritt durchlaufen. Für einen dieser Durchläufe ist $h = m$. Die Ausgabebasis ist daher L^3–reduziert mit δ'.

Zum Beweis der $(3, \delta)$–Blockreduziertheit ist zu zeigen:

i) $|\mu_{i,j}| \leq \frac{1}{2}$ für $1 \leq j < i \leq m$ (folgt unmittelbar aus der L^3–Reduziertheit),

ii) $\delta c_{m-1} \leq \lambda_1^2(L(\pi_{m-1}(b_{m-1}), \pi_{m-1}(b_m)))$ (wegen der L^3–Reduziertheit mit $\delta' \geq \delta$ gilt sogar $\delta c_{m-1} \leq \delta' c_{m-1} \leq \lambda_1^2(L(\pi_{m-1}(b_{m-1}), \pi_{m-1}(b_m))))$,

iii) $\delta c_j \leq \lambda_1^2(L(\pi_j(b_j), \pi_j(b_{j+1}), \pi_j(b_{j+2})))$ für $j = 1, \ldots, m-2$.

zu iii) Es genügt zu zeigen, daß einer der angegebenen Koeffizientenvektoren (u_j, u_{j+1}, u_{j+2}) die Darstellung eines Vektors $b \in L(b_j, b_{j+1}, b_{j+2})$ liefert mit

$$\|\pi_j(b)\|_2^2 = \lambda_1^2(L(\pi_j(b_j), \pi_j(b_{j+1}), \pi_j(b_{j+2}))) < \delta c_j, \tag{4.1}$$

falls ein solcher Vektor existiert.

Sei also $b = u_j b_j + u_{j+1} b_{j+1} + u_{j+2} b_{j+2}$, $(u_j, u_{j+1}, u_{j+2}) \in \mathbb{Z}^3 \setminus 0$ mit (4.1). Dann ist

$$\|\pi_j(b)\|_2^2 = u_{j+2}^2 c_{j+2} + (u_{j+1} + u_{j+2}\mu_{j+2,j+1})^2 c_{j+1} + (u_j + u_{j+1}\mu_{j+1,j} + u_{j+2}\mu_{j+2,j})^2 c_j. \tag{4.2}$$

Jeder der drei Summanden ist nichtnegativ. Daher gilt stets

$$u_j = \lceil -u_{j+1}\mu_{j+1,j} - u_{j+2}\mu_{j+2,j} \rfloor, \tag{4.3}$$

denn für diese Wahl von u_j für festes (u_{j+1}, u_{j+2}) wird der dritte Summand von (4.2) minimal.

Es gilt $u_{j+2} \neq 0$, denn aufgrund der L^3–Reduziertheit ist $\|\pi_j(u_j b_j + u_{j+1} b_{j+1})\|_2^2 \geq \delta' c_j$ für $(u_j, u_{j+1}) \neq (0, 0)$ und $\delta' \geq \frac{1}{2}$.

Aus Symmetriegründen können wir uns auf den Fall $u_{j+2} > 0$ beschränken. Mit Lemma 2.6 erhalten wir $\delta c_j > \|\pi_j(b)\|_2^2 \geq u_{j+2}^2 c_{j+2} \geq u_{j+2}^2 (\delta' - \frac{1}{4})^2 c_j = u_{j+2}^2 \delta^4 c_j$. Wegen $\delta \geq \frac{1}{2}$ folgt daraus $u_{j+2} \leq 2$.

Mit (4.2) und Lemma 2.6 gilt

$$\begin{aligned} \delta c_j &> u_{j+2}^2 c_{j+2} + (u_{j+1} + u_{j+2}\mu_{j+2,j+1})^2 c_{j+1} \\ &\geq u_{j+2}^2 \Big(\delta' - \frac{1}{4}\Big)^2 c_j + (u_{j+1} + u_{j+2}\mu_{j+2,j+1})^2 \Big(\delta' - \frac{1}{4}\Big) c_j. \end{aligned}$$

Durch Umformen erhalten wir

$$|u_{j+1} + u_{j+2}\mu_{j+2,j+1}| < \sqrt{\frac{\delta - u_{j+2}^2(\delta' - \frac{1}{4})^2}{\delta' - \frac{1}{4}}} = \sqrt{\frac{1 - u_{j+2}^2 \delta^3}{\delta}}.$$

Für $u_{j+2} = 1$ ergibt sich $|u_{j+1} + \mu_{j+2,j+1}| < \frac{1}{2}\sqrt{7}$ und damit $|u_{j+1}| \leq 1$ wegen $|\mu_{j+2,j+1}| \leq \frac{1}{2}$. Wegen $|\mu_{i,j}| \leq \frac{1}{2}$ und (4.3) gilt $|u_j| \leq 1$.

Für $u_{j+2} = 2$ ergibt sich $|u_{j+1} + 2\mu_{j+2,j+1}| < 1$ und damit ebenfalls $|u_{j+1}| \leq 1$.

Mit (4.2) und Lemma 2.6 gilt

$$\begin{aligned} \delta c_j &> u_{j+2}^2 c_{j+2} + (u_j + u_{j+1}\mu_{j+1,j} + u_{j+2}\mu_{j+2,j})^2 c_j \\ &\geq 4\delta^4 c_j + (u_j + u_{j+1}\mu_{j+1,j} + 2\mu_{j+2,j})^2 c_j. \end{aligned}$$

Wegen $\delta \geq \frac{1}{2}$ folgt daraus $|u_j + u_{j+1}\mu_{j+1,j} + 2\mu_{j+2,j}| < \sqrt{\delta - 4\delta^4} \leq \frac{1}{2}$.
Mit $|u_{j+1}| \leq 1$ und $|\mu_{i,j}| \leq \frac{1}{2}$ folgt $|u_j| \leq 1$. Insbesondere gilt $d := ggT(u_j, \ldots, u_{j+2}) = 1$. Andernfalls wäre auch $\frac{1}{d}b$ ein Gittervektor; dieser Vektor hätte ein kleineres Höhenquadrat als b. Dies steht im Widerspruch zur Minimalität von $\|\pi_j(b)\|_2^2$. $\square$

Theorem 4.2 *Für jede ganzzahlige Gitterbasis $b_1, \ldots, b_m \in \mathbb{Z}^n$ und jedes $\delta \in [\frac{1}{2}, \frac{1}{2}\sqrt{3})$ terminiert der Algorithmus Block–3–Reduktion nach $O\Big(m^2 n(1 + m^3 \log_{1/\delta'} M)\Big)$ arithmetischen Operationen. Hierbei ist $\delta' := \delta^2 + \frac{1}{4}$ und $M := \max_{i=1,\ldots,m} \|b_i\|_2^2$.*

Der Beweis von Theorem 4.2 lehnt sich eng an den Beweis der Laufzeitschranke für den L^3–Algorithmus an. Anstelle des Produkts $\prod_{i=1}^{m-1} D_i$ betrachten wir $D := \prod_{i=1}^{m-1} D_i^{m-i}$ und zeigen, daß D bei jedem Austausch $b_k \leftrightarrow b_{k-1}$ und bei jedem Einfügen eines Vektors mindestens um einen konstanten Faktor kleiner wird und sich bei allen übrigen Schritten nicht verändert. Wir beweisen, daß zwischen zwei Veränderungen von D nur polynomial viele arithmetische Operationen durchgeführt werden und D sich nur polynomial oft verändern kann. Hierfür stellen wir zunächst die benötigten Hilfsmittel bereit:

Lemma 4.3 *Nach Einfügen des Vektors b an die j–te Stelle der Basis gilt*

$$\begin{aligned} c_j^{\text{neu}} &< \delta c_j, && (4.4) \\ c_{j+1}^{\text{neu}} &\leq \frac{1}{\delta^2}\, c_{j+1}, && (4.5) \\ \max_{1\leq i\leq m} c_i^{\text{neu}} &\leq \max_{1\leq i\leq m} c_i, && (4.6) \\ c_j^{\text{neu}} c_{j+1}^{\text{neu}} c_{j+2}^{\text{neu}} &= c_j c_{j+1} c_{j+2}. && (4.7) \end{aligned}$$

Beweis von Lemma 4.3:
zu (4.4):
b wird nur dann an die j–te Stelle der Basis eingefügt, wenn $c_j^{\text{neu}} = \|\pi_j(b)\|_2^2 < \delta c_j$.
zu (4.5),(4.6):
Für $u_{j+2} = 1$ gilt $b_j^{\text{neu}} = b$ und $b_i^{\text{neu}} = b_{i-1}$ für $i = j+1, j+2$.
Die übrigen Basisvektoren bleiben unverändert. Damit ist $c_i^{\text{neu}} \leq c_{i-1} \leq \frac{1}{\delta' - \frac{1}{4}}\, c_i$ für $i = j+1, j+2$.

Für $u_{j+2} = 2$ und $|u_{j+1}| = 1$ gilt $b_j^{\text{neu}} = b$ und $b_{j+1}^{\text{neu}} = b_j$.
Die übrigen Basisvektoren bleiben unverändert. Wir erhalten $c_{j+1}^{\text{neu}} \leq c_j \leq \frac{1}{\delta' - \frac{1}{4}}\, c_{j+1}$ und $c_{j+2}^{\text{neu}} \leq \|\pi_{j+1}(b_{j+2})\|_2^2 = c_{j+2} + \mu_{j+2,j+1}^2 c_{j+1} < \frac{\delta}{4} c_j + \frac{1}{4} c_{j+1} \leq c_{j+1} \leq \frac{1}{\delta' - \frac{1}{4}} c_{j+2}$.

Für $u_{j+2} = 2$ und $u_{j+1} = 0$ gilt $|u_j| = 1$ wegen $\mathrm{ggT}(u_j, \ldots, u_{j+2}) = 1$. Wir erhalten $b_j^{\text{neu}} = b$.
Die übrigen Basisvektoren bleiben unverändert.
Damit ist $c_{j+1}^{\text{neu}} \leq \|\pi_j(b_{j+1})\|_2^2 = c_{j+1} + \frac{1}{4} c_j \leq \left(1 + \frac{1}{\delta' - \frac{1}{4}}\right) c_{j+2} + \frac{1}{4} c_j$ und
$c_{j+2}^{\text{neu}} \leq \|\pi_j(b_{j+2})\|_2^2 = c_{j+2} + \frac{1}{4} c_{j+1} + \frac{1}{4} c_j$. Wegen $u_{j+2} = 2$ ist $c_{j+2} < \frac{\delta}{4} c_j$ und damit $c_{j+1}^{\text{neu}} \leq \frac{1+\delta}{4\delta} c_j \leq c_j \leq \frac{1}{\delta' - \frac{1}{4}} c_{j+1}, \quad c_{j+2}^{\text{neu}} \leq \left(\frac{\delta}{4} + \frac{1}{2}\right) \max_{1 \leq i \leq m} c_i \leq \max_{1 \leq i \leq m} c_i$.

Mit $\delta' - \frac{1}{4} = \delta^2$ folgt die Behauptung.

zu (4.7):
Die Determinante des Teilgitters $L(b_1, \ldots, b_{j+2})$ bleibt unverändert. □

Lemma 4.4 *Nach Einfügen des Vektors b an die j–te Stelle der Basis gilt $D^{\text{neu}} < \delta D^{\text{alt}}$.*

Beweis von Lemma 4.4:
Nach Lemma 4.3 gilt $c_j^{\text{neu}} < \delta c_j$ und $c_{j+1}^{\text{neu}} \leq \frac{1}{\delta^2} c_{j+1}$. Das Produkt $c_j c_{j+1} c_{j+2}$ sowie die übrigen c_i bleiben unverändert. Damit gilt nach dem Einfügen

$$D_i^{\text{neu}} \begin{cases} < \delta D_i & \text{für } i = j, \\ < \frac{1}{\delta} D_i & \text{für } i = j+1, \\ = D_i & \text{sonst.} \end{cases}$$

Insgesamt folgt daraus $D^{\text{neu}} < \delta D^{\text{alt}}$. □

Lemma 4.5 *Jeder Austauschschritt im L^3–Algorithmus bewirkt $D^{\text{neu}} < \delta' D^{\text{alt}}$. Die übrigen Schritte des L^3–Algorithmus verändern D nicht.*

Beweis von Lemma 4.5:
Ein Austausch der Vektoren b_{k-1} und b_k bewirkt $c_{k-1}^{\text{neu}} < \delta' c_{k-1}$. Das Produkt $c_{k-1} c_k$ sowie die übrigen c_i bleiben unverändert. Damit gilt nach dem Austauschschritt $b_k \leftrightarrow b_{k-1}$

$$D_i^{\text{neu}} \begin{cases} < \delta' D_i & \text{für } i = k-1, \\ = D_i & \text{sonst.} \end{cases}$$

Insgesamt folgt daraus $D^{\mathrm{neu}} < \delta'^{m-k+1} D \leq \delta' D$. Die übrigen Schritte des L^3–Algorithmus verändern keines der c_i. Damit bleibt auch D unverändert. $\square$

Beweis von Theorem 4.2: Nach Lemma 4.5 und Lemma 4.4 bewirkt jeder Austauschschritt $b_k \leftrightarrow b_{k-1}$ im L^3–Algorithmus $D^{\mathrm{neu}} < \delta' D$ und jedes Einfügen eines Vektors $D^{\mathrm{neu}} < \delta D$. Die übrigen Schritte des Algorithmus Block–3–Reduktion verändern D nicht. Damit liegen zwischen zwei Veränderungen von D höchstens $O(m)$ Längenreduktionen eines Vektors und $m-2$ Enumerationen. Für die Längenreduktion eines Vektors sind $O(mn)$ arithmetische Operationen erforderlich. Die Bestimmung von $\bar{c}_j$ benötigt nur konstant viele Operationen. Zum Berechnen und Einfügen des Vektors b sind $O(n)$ arithmetische Operationen nötig. Insgesamt werden also höchstens $O(m^2 n)$ arithmetische Operationen durchgeführt, bevor D um den Faktor δ' verkleinert wird. Für die Eingabebasis gilt $D \leq M^{\binom{m+1}{3}}$. Wegen der Ganzzahligkeit der Basis sind alle D_i und damit auch D stets positiv und ganzzahlig. Die Gesamtzahl der arithmetischen Operationen ist also für ganzzahlige Eingaben beschränkt durch

$$O\left(m^2 n \left(1 + \log_{1/\delta'} M^{\binom{m+1}{3}}\right)\right) = O\left(m^2 n \left(1 + m^3 \log_{1/\delta'} M\right)\right). \qquad \square$$

Bemerkung 4.6 Für reelle Eingabebasen geht lediglich die Ganzzahligkeit von D verloren. Damit ist die 1 nicht mehr notwendigerweise eine untere Schranke für D. Mit Hilfe der *Minkowskischen Ungleichung*

$$\prod_{t=1}^{i} \lambda_{t,\|.\|_2}(L) \leq \gamma_i^i \det(L(b_1, \ldots, b_i)) \tag{4.8}$$

erhalten wir

$$D \geq \lambda_{1,\|.\|_2}^{(m^3-m)/3}(L) \prod_{i=1}^{m-1} \gamma_i^{-i(m-i)}.$$

Die Gesamtzahl der arithmetischen Operationen ist also für reelle Eingabebasen beschränkt durch

$$O\left(m^2 n \left(1 + m^3 \log_{1/(\delta^2+\frac{1}{4})} \left(mM/\lambda_{1,\|.\|_2}^2(L)\right)\right)\right).$$

Wir schätzen nun noch die Größe der auftretenden Zahlen bei ganzahliger Eingabebasis ab.

Satz 4.7 *Bei Eingabe einer Gitterbasis $b_1, \ldots, b_m \in \mathbb{Z}^n$ und $\delta \in [\frac{1}{2}, \frac{1}{2}\sqrt{3})$ sind alle Zähler und Nenner der im Algorithmus Block–3–Reduktion auftretenden rationalen Zahlen absolut beschränkt durch $\left(\frac{3}{2}\right)^{2m} \frac{m+3}{4} M^{2m} \delta^{4-4m}$. Hierbei ist $M := \max_{1 \leq i \leq m} \|b_i\|_2^2$.*

Beweis von Satz 4.7: Während der L^3–Reduktion in Schritt 1 wird nach Satz 1.6 stets mit rationalen Zahlen gerechnet, deren Zähler und Nenner absolut beschränkt sind durch $mM^m(9/(4\delta'-1))^{(m/2)} = m(3M/(2\delta))^m$. Die Schranke für die Größe der im L^3–Algorithmus auftretenden Zahlen folgt aus der Eigenschaft, daß bei Eintritt in Stufe k des L^3–Algorithmus stets $b_1,\ldots,b_{k-1}$ mit δ' L^3–reduziert ist.

Bei der Bestimmung von $\bar{c}_j$ gilt wegen der L^3–Reduziertheit von $b_1,\ldots,b_{j+2}$

$$\begin{aligned}
|\mu_{i,t}| &\le \frac{1}{2}, \quad 1 \le t < i \le j+2,\\
c_i &\le M, \quad 1 \le i \le j+2,\\
\|b_i\|_2^2 &\le \frac{i+3}{4}M, \quad 1 \le i \le j+2,\\
\mu_{i,t}D_t &\in \mathbb{Z}, \quad 1 \le t < i \le j+2,\\
c_iD_{i-1} &\in \mathbb{Z}, \quad 1 \le i \le j+2.
\end{aligned}$$

Dadurch treten im Laufe der Berechnung von $\bar{c}_j$ nur rationale Zahlen auf, deren Zähler und Nenner absolut durch $15M^{2m}$ beschränkt sind.

Nach Einfügen des Vektors b gilt

$$\left|\mu_{j,t}^{\text{neu}}\right| \le 2 \quad \text{für } 1 \le t < j, \tag{4.9}$$

$$\|b_i^{\text{neu}}\|_2^2 \le \begin{cases} 4iM & \text{für } i = j, \\ \frac{i+3}{4}M & \text{sonst.} \end{cases} \tag{4.10}$$

zu (4.9): $|\mu_{j,t}^{\text{neu}}| = \left|\dfrac{< b_j^{\text{neu}}, \hat{b}_t >}{< \hat{b}_t, \hat{b}_t >}\right| = \left|\sum_{i=j}^{j+2} u_i \dfrac{< b_i, \hat{b}_t >}{< \hat{b}_t, \hat{b}_t >}\right| = \sum_{i=j}^{j+2} |u_i \mu_{i,t}| \le \dfrac{1}{2}\sum_{i=j}^{j+2}|u_i| \le 2.$

zu (4.10): Für $i < j$ und $i > j+2$ werden die Vektoren b_i durch das Einfügen nicht verändert; die Behauptung gilt daher wegen der L^3–Reduziertheit von $b_1,\ldots,b_{j-1}$.
Für b_j^{neu} gilt

$$\begin{aligned}
\|b_j^{\text{neu}}\|_2^2 &= \|u_jb_j + u_{j+1}b_{j+1} + u_{j+2}b_{j+2}\|_2^2\\
&= c_j^{\text{neu}} + \sum_{t=1}^{j-1}\left(\mu_{j,t}^{\text{neu}}\right)^2 c_t\\
&\le \delta M + 4(j-1)M\\
&\le 4jM.
\end{aligned}$$

Die Vektoren b_{j+1}^{neu} und b_{j+2}^{neu} wurden höchstens um eine Position nach hinten geschoben. Damit folgt die Behauptung unmittelbar aus der L^3–Reduziertheit der Ausgangsbasis.

Wir müssen nun noch die Größe der Zahlen beschränken, die im Laufe der L^3–Reduktionen von Schritt 2 auftreten. Sei also $b_1,\ldots,b_{k-1}$ L^3–reduziert mit δ'. Wir betrachten die Veränderung der Größen $\|b_k\|_2^2$ und $|\mu_{k,i}|$ für $i < k$ im Verlauf der Längenreduktion von b_k:

Seien $b_k^{(k)} := b_k^{\text{Start}}$ und $\mu_{k,t}^{(k)} := \mu_{k,t}^{\text{Start}}$ für $t = 1, \ldots, k-1$ die Größen zu Beginn der Längenreduktion sowie

$$\begin{aligned}
b_k^{(i)} &:= b_k^{(i+1)} - \lceil \mu_{k,i}^{(i+1)} \rfloor b_i \quad \text{für } 1 \leq i < k, \\
\mu_{k,t}^{(i)} &:= \mu_{k,t}^{(i+1)} - \lceil \mu_{k,i}^{(i+1)} \rfloor \mu_{i,t} \quad \text{für } 1 \leq t \leq i, \\
\mu_{k,t}^{(i)} &:= \mu_{k,t}^{(i+1)} \quad \text{für } i+1 \leq t < k
\end{aligned}$$

die Schritte der Längenreduktion von b_k.

Mit $M_i := \max\limits_{1 \leq t < i} |\mu_{k,t}^{(i)}|$ erhalten wir $M_i \leq M_{i+1} + (M_{i+1} + \frac{1}{2})\frac{1}{2} = \frac{3}{2} M_{i+1} + \frac{1}{4}$ und damit

$$\begin{aligned}
M_i &\leq \left(\frac{3}{2}\right)^{k-i} (M_k + \frac{1}{2}) - \frac{1}{2}, \\
\|b_k^{(i)}\|_2^2 &= \|\hat{b}_k\|_2^2 + \sum_{t=1}^{k-1} \left(\mu_{k,t}^{(i)}\right)^2 \|\hat{b}_t\|_2^2 \\
&\leq M \left(1 + (i-1) M_i^2 + \frac{k-i}{4}\right).
\end{aligned}$$

Nach der Längenreduktion gilt also stets $\|b_k\|_2^2 \leq \frac{k+3}{4} M$.

Nach Einfügen des Vektors b ist $M_j \leq 2$ und im Verlauf der Längenreduktion von $b_j = b$ gilt

$$\begin{aligned}
\|b_j\|_2^2 &\leq M \max_{1 \leq i \leq j} \left(1 + (i-1) M_i^2 + \frac{j-i}{4}\right) \\
&\leq M m \left(\frac{3}{2}\right)^{2m}, \\
|\mu_{j,i}| &\leq \left(\frac{3}{2}\right)^m.
\end{aligned}$$

Für alle anderen Längenreduktionen eines Vektors b_k gilt $M_k^2 \leq \frac{k+3}{4} M \delta^{4-2k}$, denn nach Satz 1.5 ist $|\mu_{k,i}|^2 = |<b_k, \hat{b}_i>|^2 \, \|\hat{b}_i\|_2^{-4} \leq \|b_k\|_2^2 \, \|\hat{b}_i\|_2^{-2} \leq \frac{k+3}{4} M \delta^{2k-2}$. Damit gilt im Verlauf der Längenreduktion von b_k stets

$$\begin{aligned}
\|b_k\|_2^2 &\leq \left(\frac{3}{2}\right)^{2k-1} \frac{k+3}{4} M^2 \delta^{4-2k}, \\
|\mu_{k,j}| &\leq \left(\frac{3}{2\delta^2}\right)^{k-1} \left(M \frac{k+3}{4}\right)^{1/2}.
\end{aligned}$$

Wegen $\mu_{i,j} D_j \in \mathbb{Z}$ und $\|\hat{b}_j\|_2^2 D_{j-1} = D_j \in \mathbb{Z}$ sind alle auftretenden Nenner durch M^{m-1} beschränkt. $\square$

Bemerkung 4.8 Für $\frac{1}{2} \leq \delta < \frac{1}{2}\sqrt[3]{4} = 0.7937\ldots$ approximiert der Algorithmus Block–3–Reduktion die sukzessiven Minima des Gitters besser als eine L^3–Reduktion mit $\delta' = \delta^2 + \frac{1}{4}$. Einerseits ist die Ausgabebasis mit δ' L^3–reduziert und wir erhalten mit Satz 1.5

$$\frac{\|\hat{b}_j\|_2^2}{\lambda^2_{j,\|\cdot\|_2}(L)} \leq \left(\frac{1}{\delta' - \frac{1}{4}}\right)^{m-j} = \delta^{2(j-m)}.$$

Andererseits folgt mit Theorem 2.3 aus der $(3, \delta)$–Blockreduziertheit

$$\frac{\|\hat{b}_j\|_2^2}{\lambda^2_{j,\|\cdot\|_2}(L)} \leq \frac{1}{\delta}\left(\frac{\sqrt[3]{2}}{\delta}\right)^{m-j}.$$

Für $\delta < \frac{1}{2}\sqrt[3]{4}$ (d.h. für $\delta' < \frac{1}{2}\sqrt[3]{2} + \frac{1}{4} = 0.8799\ldots$) ist die Schranke aus Theorem 2.3 schärfer.

4.2 Worst–Case–Analyse der Aufzählungsverfahren

Bei jeder Iteration der Blockreduktionsalgorithmen aus Kapitel 2.2 wird eine Aufzählung durchgeführt. Die Laufzeiten der Blockreduktionsalgorithmen hängen also entscheidend von den Laufzeiten für die Aufzählungsverfahren ab. Wir analysieren zunächst das Aufzählungsverfahren ADFS für den Fall allgemeiner Norm.

4.2.1 Worst–Case–Analyse des Algorithmus ADFS

Schritt 1 des Algorithmus ADFS (siehe Seite 31) wird genau einmal durchlaufen. Hierfür werden $O(k - j + 1)$ arithmetische Operationen benötigt. Für jeden Durchlauf von Schritt 2 werden $O(n(k - j + 1)) = O(n\beta)$ arithmetische Operationen und $O(1)$ Berechnungen der Höhenfunktion durchgeführt. Wir schätzen im folgenden die Anzahl A der Durchläufe von Schritt 2 nach oben ab.

Sei $A(t, x)$ die Anzahl der auf Stufe t aufgezählten Knoten $(\tilde{u}_t, \ldots, \tilde{u}_k)$ mit $F_t(\sum_{i=t}^{s} \tilde{u}_i b_i) < x$.

Bei Eintritt in Stufe t gilt stets $F_{t+1}(\sum_{i=t+1}^{k} \tilde{u}_i b_i) < F_j(b_j)$. Damit wird die Stufe höchstens $A(t+1, F_j(b_j))$ mal von $t+1$ auf t erniedrigt und $A(t, F_j(b_j)) + 1$ mal von $t-1$ auf t erhöht. Die Stufe $t = j$ bleibt höchstens $A(j, F_j(b_j))$ mal unverändert. Wir erhalten

$$A \leq k - j + 1 + A(j, F_j(b_j)) + 2\sum_{t=j+1}^{k} A(t, F_j(b_j)). \tag{4.11}$$

Wegen $F_t(x) \geq F_{t+1}(x)$ gilt

$$A(t, F_j(b_j)) \leq A(t+1, F_j(b_j)) \max \#\left\{\tilde{u}_t \in \mathbb{Z} \,\middle|\, F_t(\textstyle\sum_{i=t}^{k} \tilde{u}_i b_i) < F_j(b_j)\right\},$$

wobei das Maximum über alle ganzzahligen, von 0 verschiedenen Vektoren $(\widetilde{u}_{t+1}, \ldots, \widetilde{u}_k)$ gebildet wird.

Für festes $(\widetilde{u}_{t+1}, \ldots, \widetilde{u}_k)$ sei $\widetilde{b}_{t+1} := \sum_{i=t+1}^{k} \widetilde{u}_i b_i$ und $F_{t+1}(\widetilde{b}_{t+1}) := F_t(\widetilde{b}_{t+1} + z b_t) < F_j(b_j)$.

Damit gilt für jedes $\widetilde{u}_t \in \mathbb{Z}$ mit $F_t(\widetilde{b}_{t+1} + \widetilde{u}_t b_t) < F_j(b_j)$

$$\begin{aligned}
F_j(b_j) &> F_t(\widetilde{b}_{t+1} + \widetilde{u}_t b_t) \\
&= F_t((z - \widetilde{u}_t) b_t - (z b_t + \widetilde{b}_{t+1})) \\
&\geq F_t((z - \widetilde{u}_t) b_t) - F_t(z b_t + \widetilde{b}_{t+1})) \\
&= |z - \widetilde{u}_t| F_t(b_t) - F_{t+1}(\widetilde{b}_{t+1}) \\
&> |z - \widetilde{u}_t| F_t(b_t) - F_j(b_j)
\end{aligned}$$

und wir erhalten

$$|z - \widetilde{u}_t| < 2 \frac{F_j(b_j)}{F_t(b_t)}.$$

Es ergibt sich also

$$A(t, F_j(b_j)) < \prod_{i=t}^{k} \Big(4 \frac{F_j(b_j)}{F_i(b_i)} + 1 \Big). \tag{4.12}$$

Es gibt Folgen von Gitterbasen, für die die Quotienten $F_j(b_j)/F_i(b_i)$ und damit die obere Schranke für die Anzahl der Iterationen von Schritt 2 beliebig groß werden. Das folgende Beispiel zeigt, daß dies nicht nur für die obere Schranke, sondern auch für die tatsächliche Anzahl der Iterationen gilt:

Wir betrachten das durch die folgende Basis aufgespannte Gitter L:

$$\begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix} = \begin{pmatrix} 2^a & 0 & 0 \\ 0 & 2^{2a} & 0 \\ 0 & 2^a & 1 \end{pmatrix}, \quad a \in \mathbb{N}$$

Wir suchen einen in l_∞–Norm kürzesten Gittervektor. Hierzu bestimmen wir zunächst die Anzahl der Iterationen des Algorithmus ADFS bei Eingabe $b_1, b_2, b_3,\ j = 1, k = 3$:

Es gilt $F_1(b_1) = 2^a,\ F_1(b_2) = F_2(b_2) = 2^{2a},\ F_1(b_3) = F_2(b_3) = 2^a,\ F_3(b_3) = 1$.
Damit wird Schritt 2 genau einmal für $s = 1$ und $s = 2$ durchlaufen.
Für $s < 3$ ist $F_1(\sum_{i=1}^{s} \widetilde{u}_i b_i) \geq F_1(b_1) = 2^a$.
Für jedes $\widetilde{u}_3 \in M := \{1, \ldots, 2^a - 1\}$ gilt

$$\begin{aligned}
F_3(\widetilde{u}_3 b_3) &= \widetilde{u}_3 < 2^a \quad \text{und} \\
\min_{\widetilde{u}_2 \in \mathbb{Z}} F_2(\widetilde{u}_3 b_3 + \widetilde{u}_2 b_2) &= \min_{\widetilde{u}_2 \in \mathbb{Z}} \max(|\widetilde{u}_3 2^a + \widetilde{u}_2 2^{2a}|, \widetilde{u}_3) \geq 2^a.
\end{aligned}$$

Damit wird Schritt 2 für jeden der $2^a - 1$ Werte von $\widetilde{u}_3 \in M$ genau einmal auf Stufe $t = 3$ und einmal auf Stufe $t = 2$ durchlaufen. Die Zahl der Iterationen wächst damit exponentiell in a. Es kann also keine obere Schranke für die Anzahl der Iterationen des Algorithmus ADFS geben, die nur von der Blockgröße abhängt und für jede beliebige Eingabebasis gilt.

Wir betrachten nun den Aufwand zur Bestimmung von $\lambda_{1,\|\cdot\|_\infty}(L)$, wenn zunächst eine $(2,\Delta)$–Blockreduktion (mit $\Delta > 2^{-a}$) durchgeführt und anschließend der Algorithmus ADFS aufgerufen wird:

Nach 4 Iterationen der WHILE–Schleife des Algorithmus $(2,\Delta)$–REDUCE erhalten wir die Basis

$$\begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix} = \begin{pmatrix} 2^a & 0 & 0 \\ 0 & 2^a & 1 \\ 0 & 0 & -2^a \end{pmatrix}.$$

Bei Eigabe dieser Basis in ADFS wird Schritt 2 nur 4 mal durchlaufen. Die Anzahl der Berechnungsschritte (arithmetische Operationen bzw. Berechnung der Funktionswerte F_t) ist konstant und damit unabhängig von a.

Das Beispiel hat gezeigt, daß die Anzahl der Iterationen des Algorithmus ADFS entscheidend von der Wahl der Basis des Gitters abhängt. Wir schätzen nun die Anzahl der Iterationen für den Fall mit Δ LS–reduzierter Basen ab. (Solche Basen können für $\Delta < 1$ nach Satz 1.13 in polynomial vielen Schritten konstruiert werden.)

Für LS–reduzierte Basen gilt $\Delta F_{t-1}(b_{t-1}) \leq F_{t-1}(b_t) \leq F_t(b_t) + \frac{1}{2}F_{t-1}(b_{t-1})$ nach Definition 2.1 und wir erhalten durch Induktion

$$F_j(b_j)/F_t(b_t) \leq (\Delta - \frac{1}{2})^{j-t}. \tag{4.13}$$

Durch Einsetzen dieser Schranke in Ungleichung (4.12) ergibt sich

$$\begin{aligned} A(t, F_j(b_j)) &< \prod_{i=t}^{k} \left(4\left(\Delta - \frac{1}{2}\right)^{j-i} + 1\right) < 5^{k-t+1}\left(\Delta - \frac{1}{2}\right)^{\sum_{i=t}^{k} j-i} \\ &= 5^{k-t+1}\left(\Delta - \frac{1}{2}\right)^{(k-t+1)(j-(k+t)/2)} = \left(\frac{1}{\Delta - \frac{1}{2}}\right)^{O(\beta^2)}. \end{aligned}$$

Unter Verwendung von Ungleichung (4.11) erhalten wir den folgenden

Satz 4.9 *Bei Eingabe einer mit $\Delta \in (\frac{1}{2}, 1]$ LS–reduzierten Gitterbasis benötigt der Algorithmus ADFS für Blockgröße $\beta = k - j + 1$ höchstens $(1/(\Delta - \frac{1}{2}))^{O(\beta^2)}$ Iterationen von Schritt 2.*

4.2.2 Worst–Case–Analyse des Algorithmus ENUM

Analog zum Algorithmus ADFS wird Schritt 1 auch beim Algorithmus ENUM (siehe Seite 38) genau einmal durchlaufen. Hierfür werden $O(k - j + 1)$ arithmetische Operationen benötigt. Ein Durchlauf von Schritt 2 benötigt ebenfalls $O(k - j + 1)$ arithmetische

Operationen. Die Anzahl A der Durchläufe von Schritt 2 ist wiederum beschränkt durch

$$A \leq k - j + 1 + A(j, c_j) + 2 \sum_{t=j+1}^{k} A(t, c_j). \tag{4.14}$$

Dabei ist $A(t,x)$ die Anzahl der auf Stufe t aufgezählten Knoten $(\widetilde{u}_t, \ldots, \widetilde{u}_s)$ mit $\widetilde{c}_t < x$.

Für festes $(\widetilde{u}_{t+1}, \ldots, \widetilde{u}_s)$ sei $\widetilde{b}_{t+1} := \sum_{i=t+1}^{s} \widetilde{u}_i b_i$ und $\widetilde{c}_{t+1} := \|\pi_{t+1}(b_{t+1})\|_2^2 < c_j$.

Für jedes $\widetilde{u}_t \in \mathbb{Z}$ mit $\widetilde{c}_t := \|\pi_t(b_{t+1} + \widetilde{u}_t b_t)\|_2^2 < c_j$ gilt

$$\begin{aligned} c_j &> \widetilde{c}_t = \widetilde{c}_{t+1} + (\widetilde{u}_t + \textstyle\sum_{i=t+1}^{s} \widetilde{u}_i \mu_{i,t})^2 c_t \\ &\geq (\widetilde{u}_t + \textstyle\sum_{i=t+1}^{s} \widetilde{u}_i \mu_{i,t})^2 c_t \end{aligned}$$

und damit

$$|\widetilde{u}_t + \textstyle\sum_{i=t+1}^{s} \widetilde{u}_i \mu_{i,t}| < \sqrt{\frac{c_j}{c_t}}.$$

Es ergibt sich also

$$A(t, c_j) < \prod_{i=t}^{k} \Big(2\sqrt{\frac{c_j}{c_i}} + 1\Big). \tag{4.15}$$

Dieses Produkt kann, ebenso wie die tatsächliche Anzahl der Iterationen, für allgemeine Eingaben bei fester Blockgröße beliebig groß werden. Für L^3–reduzierte Gitterbasen erhalten wir jedoch eine obere Schranke, die nur noch von der Blockgröße und dem Parameter δ abhängt:

Für mit δ L^3–reduzierte Basen ist $c_j/c_t \leq (\delta - \frac{1}{4})^{j-t}$. Durch Einsetzen dieser Schranke in Ungleichung (4.15) ergibt sich

$$\begin{aligned} A(t, c_j) &< \prod_{i=t}^{k} \left(2\left(\delta - \frac{1}{4}\right)^{(j-i)/2} + 1\right) < 3^{k-t+1}\left(\delta - \frac{1}{4}\right)^{\frac{1}{2}\sum_{i=t}^{k}(j-i)} \\ &= 3^{k-t+1}\left(\delta - \frac{1}{4}\right)^{\frac{1}{2}(k-t+1)(j-(k+t)/2)} = \left(\frac{1}{\delta - \frac{1}{4}}\right)^{O(\beta^2)}. \end{aligned}$$

Unter Verwendung von Ungleichung (4.14) erhalten wir den folgenden

Satz 4.10 *Bei Eingabe einer mit $\delta \in (\frac{1}{4}, 1]$ L^3–reduzierten Gitterbasis benötigt der Algorithmus ENUM für Blockgröße $\beta = k - j + 1$ höchstens $(1/(\delta - \frac{1}{4}))^{O(\beta^2)}$ Iterationen von Schritt 2. Die Anzahl der arithmetischen Operationen ist beschränkt durch $O(\beta)(1/(\delta - \frac{1}{4}))^{O(\beta^2)}$.*

Bemerkung 4.11 Für $\delta = 0.99$ ergibt sich $A(t, c_j) < 1.0782^{\beta^2} 3^{\beta}$.
Bei Eingabe einer $(\beta - 1, 1)$–blockreduzierten Basis reduziert sich die obere Schranke für die Anzahl der Iterationen von Schritt 2 auf $\beta^{O(\beta)}$ (siehe [Sch87]).

4.3 Average–Case–Analyse der Aufzählungsverfahren

4.3.1 Average–Case–Analyse des Algorithmus ENUM

Wir approximieren die mittlere Anzahl der Iterationen von Schritt 2 im Algorithmus ENUM. Zur Vereinfachung betrachten wir den Fall, daß anstelle der Abfrage „IF $\tilde{c}_t < \bar{c}_j$“ die Abfrage „IF $\tilde{c}_t < c$“ für eine Konstante $c > 0$ verwendet wird, d.h. alle Vektoren $b = \sum_{i=j}^{s} \tilde{u}_i b_i$ mit $\tilde{u}_s > 0$ und Höhenquadrat $\tilde{c}_j < c$ werden aufgezählt.

Die Anzahl der Vektoren eines Gitters $L \subset \mathbb{R}^n$ vom Rang $m \leq n$ in einer geeigneten Teilmenge $S \subset \operatorname{span}(L)$ läßt sich durch $\operatorname{vol}_m(S)/\det(L)$ approximieren. Dieses generelle Prinzip geht auf Gauß zurück. Wir nennen es daher *Gaußsche Volumenheuristik*.

Wir wenden die Gaußsche Volumenheuristik auf die inneren Knoten des Suchbaums an:
Zur Vereinfachung der Darstellung betrachten wir den Fall $j = 1,\ k = \beta = m$.
Gegeben sei eine Basis $b_1, \ldots, b_m$ des Gitters L.
Seien ganze Zahlen $\tilde{u}_t, \ldots, \tilde{u}_m$ fixiert mit $(\tilde{u}_t, \ldots, \tilde{u}_m) \neq 0$ und $c_t(\tilde{u}_t, \ldots, \tilde{u}_m) < c$.
Gesucht sind ganze Zahlen $\tilde{u}_1, \ldots, \tilde{u}_{t-1}$ mit $c_1(\tilde{u}_1, \ldots, \tilde{u}_m) < c$, oder, anders formuliert, zu einem gegebenen Vektor $b = \sum_{i=t}^{m} \tilde{u}_i b_i$, ein Vektor $\bar{b} = \sum_{i=1}^{t-1} \tilde{u}_i b_i \in \bar{L} := L(b_1, \ldots, b_{t-1})$ mit $\|b + \bar{b}\|_2^2 < c$.
Wir zerlegen b in die orthogonalen Anteile bezüglich $\operatorname{span}(\bar{L})$:

$$b = y - z, \quad z = -\sum_{j=1}^{t-1} \sum_{i=t}^{m} \tilde{u}_i \mu_{i,j} \hat{b}_j \in \operatorname{span}(\bar{L}), \quad y \in \operatorname{span}(\bar{L})^{\perp}, \quad \tilde{c}_t = \|y\|_2^2.$$

Wir suchen also nach einem Punkt in

$$(b + \bar{L}) \cap S_{t-1}(\sqrt{c - \tilde{c}_t}, y) = \bar{L} \cap S_{t-1}(\sqrt{c - \tilde{c}_t}, z).$$

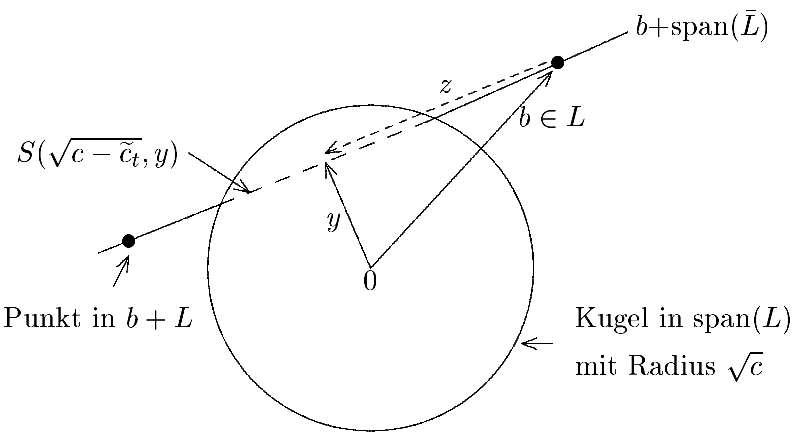

Abbildung 4.1 Gaußsche Volumenheuristik

Definition 4.12 *Für ein Gitter L mit Basis $b_1, \ldots, b_m$ heißt eine Wahrscheinlichkeitsverteilung von Punkten $\sum_{i=1}^{m} t_i b_i \in \operatorname{span}(L)$* gleichverteilt modulo L, *wenn der Vektor $(\{t_i\}\ i = 1, \ldots, m)$ der* reduzierten Koeffizienten *gleichverteilt ist in $[-\frac{1}{2}, \frac{1}{2})^m$. Hierbei ist $\{x\} := x - \lceil x \rfloor$ der Abstand von x zur nächsten ganzen Zahl.*

Diese Notation hängt nicht von der Wahl der Basis ab:
Seien $b_1, \ldots, b_m$ und $\bar{b}_1, \ldots, \bar{b}_m$ zwei Basen des Gitters L. Dann gibt es eine Matrix $U \in GL_m(\mathbb{Z})$ mit $[b_1, \ldots, b_m] = [b_1, \ldots, b_m]U$ und $|\det U| = 1$. U transformiert also die Gleichverteilung auf $\sum_{i=1}^{m} b_i[-\frac{1}{2}, \frac{1}{2})^m$ in die Gleichverteilung auf $\sum_{i=1}^{m} \bar{b}_i[-\frac{1}{2}, \frac{1}{2})^m$. Die Gleichverteilung der reduzierten Koeffizienten bleibt auch beim Übergang von einer Basis zum zugehörigen Orthogonalsystem $\hat{b}_1, \ldots, \hat{b}_m$ erhalten.

Lemma 4.13 *Sei L ein Gitter mit Basis $b_1, \ldots, b_m$, $S(z, r) \subset \operatorname{span}(L)$ eine m–dimensionale Kugel mit festem Radius r und Mittelpunkt z, der gleichverteilt ist modulo L. Dann gilt für die erwartete Anzahl $E_z\#(S(z, r) \cap L)$ von Gittervektoren in $S(z, r)$*

$$E_z\#(S(z, r) \cap L) = \frac{\operatorname{vol}_m S(z, r)}{\det L}.$$

Beweis von Lemma 4.13:
Für $z = \bar{z} \operatorname{mod} L$ gilt $\#(S(z, r) \cap L) = \#(S(\bar{z}, r) \cap L)$. Die mittlere Anzahl von Gitterpunkten in $S(z, r)$ ist gleich der mittleren Anzahl von Gitterpunkten pro Volumenanteil $\operatorname{vol}_m S(z, r)$. Damit ist der Erwartungswert für $\#(S(z, r) \cap L)$ gleich $\operatorname{vol}_m S(z, r) / \det L$. □

Satz 4.14 *Falls der Vektor $(\{\mu_{i,j}\},\ 1 \leq j < i \leq m)$ gleichverteilt ist in $[-\frac{1}{2}, \frac{1}{2})^{\binom{m}{2}}$, dann ist für jedes feste $(\widetilde{u}_t, \ldots, \widetilde{u}_m) \in \mathbb{Z}^{m-t+1} \setminus \{0\}$ die erwartete Anzahl von ganzzahligen Vektoren $(\widetilde{u}_1, \ldots, \widetilde{u}_{t-1})$ mit $c_1(\widetilde{u}_1, \ldots, \widetilde{u}_m) \leq c$ gleich* $\operatorname{vol}_{t-1} S(z, \sqrt{c - \widetilde{c}_t}) / \det(\bar{L})$.

Beweis von Satz 4.14:
Wegen $(\widetilde{u}_t, \ldots, \widetilde{u}_m) \neq 0$ existiert ein $s \in \{t, \ldots, m\}$ mit $\widetilde{u}_s \neq 0$. Damit sind die Vektoren $(\{\widetilde{u}_s \mu_{s,j}\},\ j = 1, \ldots, t-1)$ und $(\{\sum_{i=t}^{m} \widetilde{u}_i \mu_{i,j}\},\ j = 1, \ldots, t-1)$ gleichverteilt in $[-\frac{1}{2}, \frac{1}{2})^{t-1}$. Deshalb ist z gleichverteilt modulo $\bar{L}$ und die Behauptung folgt aus

$$\#\{(\widetilde{u}_1, \ldots, \widetilde{u}_{t-1}) \in \mathbb{Z}^{t-1} : c_1(\widetilde{u}_1, \ldots, \widetilde{u}_{t-1}) < c\} = \#\{S(z, \sqrt{c - \widetilde{c}_t}) \cap \bar{L}\}$$

mit Lemma 4.13. □

Die Gaußsche Volumenheuristik liefert eine gute Abschätzung für die Anzahl A der Durchläufe von Schritt 2.

Der Erwartungswert für die Anzahl $A(t, \bar{c}_1)$ der im Suchbaum auf Stufe t auftretenden Knoten $(\widetilde{u}_t, \ldots, \widetilde{u}_k)$ mit $\widetilde{c}_t < \bar{c}_1$ ist etwa der Quotient aus dem Volumen der $(m - t + 1)$–dimensionalen Kugel mit Radius $\sqrt{\bar{c}_1}$ und dem zweifachen der Determinante des Gitters

$L(\pi_t(b_t), \ldots, \pi_t(b_m))$. Der Faktor 2 entsteht durch die Beseitigung der Redundanzen (Beschränkung auf $\widetilde{u}_s > 0$) im Aufzählungsverfahren. Wir erhalten

$$A(t, \bar{c}_1) \quad \approx \quad \frac{\pi^{(m-t+1)/2}\, \bar{c}_1^{(m-t+1)/2}}{2(c_t \cdots c_m)^{1/2}\, \Gamma(\frac{m-t+1}{2} + 1)}. \tag{4.16}$$

und damit

$$A \quad \approx \quad m + A(1, \bar{c}_1) + 2 \sum_{t=2}^{m} A(t, \bar{c}_1).$$

Abbildung 4.2 zeigt eine Verteilung für die Höhenquadrate und die Anzahl der im Algorithmus ENUM auf den einzelnen Stufen des Suchbaums auftretenden Knoten, wie sie in den Anwendungen (siehe Kapitel 6) typischerweise auftritt.

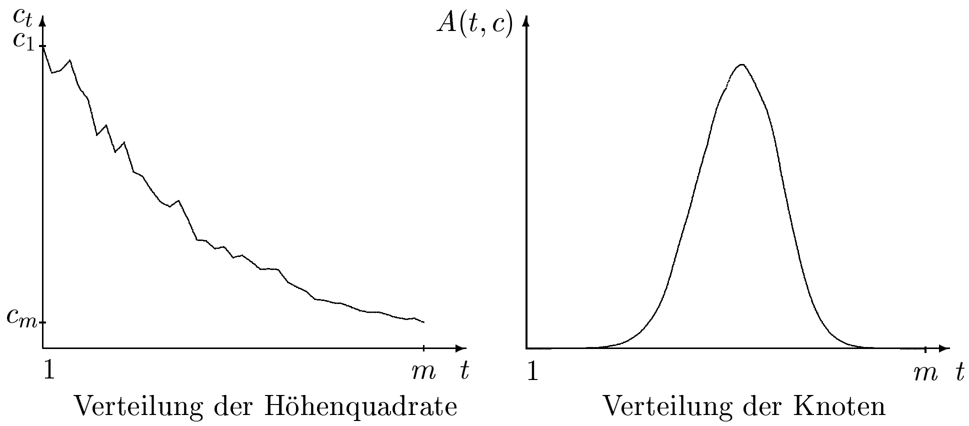

Abbildung 4.2 typisches Verhalten des Algorithmus ENUM

Die tatsächliche Zahl der auftretenden Knoten im Suchbaum stimmt sehr gut mit dem gemäß (4.16) berechneten Erwartungswert überein. Die Abweichungen liegen meist erheblich unter 1 Promille.

4.3.2 Analyse der effizienten Aufzählung bezüglich l_∞–Norm

Wie wir bereits in Kapitel 3.3 gesehen haben, werden bei der Aufzählung eines Vektors mit l_∞–Norm höchstens c und Verwendung des Abbruchtests **SCHNITT**$_\infty$ ausschließlich mit $(\lambda_t, \ldots, \lambda_s) = (1, 0, \ldots, 0)$ anstelle aller Vektoren in der Kugel um 0 mit Radius $c\sqrt{n}$ nur die Vektoren in der Vereinigung der 2^n Kugeln um $\{\pm\frac{c}{2}\}^n$ mit Radius $\frac{c}{2}\sqrt{n}$ aufgezählt. Das Volumen dieser Vereinigung ist etwa das $\left(\frac{2+\pi}{2\pi}\right)^{n-1}$–fache des Volumens der großen Kugel. Die direkte Übertragung dieser Volumenverhältnisse auf die Gaußsche Volumenheuristik liefert eine Abschätzung für die Gesamtzahl A_∞ der Durchläufe von Schritt 2, die bei praktischen Testläufen nur geringfügig von den tatsächlichen Werten abweicht.

$$A_\infty \approx \left(\frac{2+\pi}{2\pi}\right)^{k-j} A(j,n) + 2\sum_{t=j+1}^{k}\left(\frac{2+\pi}{2\pi}\right)^{k-t} A(t,n). \tag{4.17}$$

Die Anzahl der arithmetischen Operationen ist damit etwa

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

4.3.3 Analyse der effizienten Aufzählung bezüglich l_1–Norm

Bei der Aufzählung eines Vektors mit l_1–Norm höchstens c und Abbruchtest **SCHNITT$_1$** ausschließlich mit $(\lambda_t, \ldots, \lambda_s) = (1, 0, \ldots, 0)$ werden anstelle aller Vektoren in der Kugel um 0 mit Radius c nur die Vektoren in der Vereinigung der $2n$ Kugeln mit Radius $c/2$ aufgezählt (siehe Kapitel 3.4). Das Volumen dieser Vereinigung ist höchstens das $n2^{1-n}$–fache des Volumens der großen Kugel. Die direkte Übertragung dieser Volumenverhältnisse auf die Gaußsche Volumenheuristik liefert erneut eine Abschätzung für die Gesamtzahl A_1 der Durchläufe von Schritt 2.

$$A_1 \approx (k-j+1)2^{j-k}A(j,c^2) + 2\sum_{t=j+1}^{k}(k-t+1)2^{t-k}A(t,c^2). \tag{4.18}$$

Die Anzahl der arithmetischen Operationen ist damit etwa

$$O(nA_1) \approx O\left(n \cdot \left((k-j+1)2^{j-k}A(j,c^2) + 2\sum_{t=j+1}^{k}(k-t+1)2^{t-k}A(t,c^2)\right)\right).$$

Kapitel 5

Geschnittene Aufzählung für die euklidische Norm

Die Schranke für die Laufzeit der bisher betrachteten deterministischen Aufzählungsverfahren bei Eingabe einer mit $\delta \in (\frac{1}{4}, 1]$ L^3–reduzierten Gitterbasis wächst exponentiell im Quadrat der Blockgröße $\beta = k - j + 1$ (Satz 4.10). Für große β suchen wir daher nach einem probabilistischen Verfahren, das bei „erheblich geringerer“ Laufzeit „hinreichend oft“ einen kürzesten Gittervektor findet. Bei gegebener Verteilung für die Eingaben sind Äste des bei der vollständigen Aufzählung auftretenden Suchbaums so abzuschneiden, daß der Quotient aus dem Aufwand für eine geschnittene Aufzählung und der Wahrscheinlichkeit, einen kürzesten Gittervektor zu finden, möglichst klein wird. Zur Vereinfachung der Darstellung betrachten wir den Fall $j = 1,\ \beta = k = m$.

5.1 Die Gaußsche Volumenheuristik

Wir nehmen im folgenden an, daß das Aufzählungsverfahren als Eingabe eine mit δ L^3-reduzierte Gitterbasis erhält, deren zugehörige Gram–Schmidt–Koeffizienten unabhängig und gleichverteilt sind in $[-\frac{1}{2}, \frac{1}{2})$.

Wir setzen $\bar{c}_1 := c_1,\ \delta \in (\frac{1}{4}, 1]$ fest und betrachten die Knoten eines Aufzählungsbaums auf Stufe t bei Eingabe einer beliebigen, mit δ L^3–reduzierten Gitterbasis $b_1, \ldots, b_m$:

$$G_t := \left\{ (\widetilde{u}_t, \ldots, \widetilde{u}_m) \in \mathbb{Z}^{m-t+1} \;\middle|\; \begin{array}{l} \exists s \in \{t, \ldots, m\} : \widetilde{u}_s > 0, \widetilde{u}_j = 0 \text{ für } j = s+1, \ldots, m, \\ \exists \mu_{j,k} \in (-\frac{1}{2}, \frac{1}{2}], c_{j+1} \geq (\delta - \mu_{j+1,j}^2) c_j, 1 \leq k < j \leq m, \\ \widetilde{c}_t = c_t(\widetilde{u}_t, \ldots, \widetilde{u}_m) < \bar{c}_1 \end{array} \right\}.$$

G_t ist die Menge aller Knoten, die bei einer vollständigen Aufzählung auftreten können, wenn die Eingabe mit δ L^3–reduziert ist.

Sei $A_t \subset B_t \subset G_t$, $a_t := \min\{\| \sum_{i=1}^m \widetilde{u}_i b_i\|_2^2 \mid (\widetilde{u}_t, \ldots, \widetilde{u}_m) \in A_t, (\widetilde{u}_1, \ldots, \widetilde{u}_{t-1}) \in \mathbb{Z}^{t-1}\}$ und $b_t := \min\{\| \sum_{i=1}^m \widetilde{u}_i b_i\|_2^2 \mid (\widetilde{u}_t, \ldots, \widetilde{u}_m) \in B_t, (\widetilde{u}_1, \ldots, \widetilde{u}_{t-1}) \in \mathbb{Z}^{t-1}\}$.

Für zufällige, in $[-\frac{1}{2}, \frac{1}{2})$ gleichverteilte Gram–Schmidt–Koeffizienten $\mu_{i,j}, j < t$, gilt

$$
\begin{aligned}
\mathrm{Ws}(a_t = b_t) &= \frac{E(\#\{(\widetilde{u}_1, \ldots, \widetilde{u}_m) \in \mathbb{Z}^{t-1} \times A_t| \ \| \sum_{i=1}^m \widetilde{u}_i b_i\|_2^2 < \bar{c}_1\})}{E(\#\{(\widetilde{u}_1, \ldots, \widetilde{u}_m) \in \mathbb{Z}^{t-1} \times B_t| \ \| \sum_{i=1}^m \widetilde{u}_i b_i\|_2^2 < \bar{c}_1\})} \\
&= \frac{\sum\limits_{(\widetilde{u}_t, \ldots, \widetilde{u}_m) \in A_t} E(\#\{(\widetilde{u}_1, \ldots, \widetilde{u}_{t-1} \in \mathbb{Z}^{t-1}| \ \| \sum_{i=1}^m \widetilde{u}_i b_i\|_2^2 < \bar{c}_1\})}{\sum\limits_{(\widetilde{u}_t, \ldots, \widetilde{u}_m) \in B_t} E(\#\{(\widetilde{u}_1, \ldots, \widetilde{u}_{t-1} \in \mathbb{Z}^{t-1}| \ \| \sum_{i=1}^m \widetilde{u}_i b_i\|_2^2 < \bar{c}_1\})} \\
&= \frac{\sum\limits_{(\widetilde{u}_t, \ldots, \widetilde{u}_m) \in A_t} \mathrm{vol}_{t-1} S(0, \sqrt{\bar{c}_1 - \widetilde{c}_t}) / \det L(b_1, \ldots, b_{t-1})}{\sum\limits_{(\widetilde{u}_t, \ldots, \widetilde{u}_m) \in B_t} \mathrm{vol}_{t-1} S(0, \sqrt{\bar{c}_1 - \widetilde{c}_t}) / \det L(b_1, \ldots, b_{t-1})} \\
&= \frac{\sum\limits_{(\widetilde{u}_t, \ldots, \widetilde{u}_m) \in A_t} (\bar{c}_1 - \widetilde{c}_t)^{\frac{t-1}{2}}}{\sum\limits_{(\widetilde{u}_t, \ldots, \widetilde{u}_m) \in B_t} (\bar{c}_1 - \widetilde{c}_t)^{\frac{t-1}{2}}} \quad .
\end{aligned}
$$

Die Wahrscheinlichkeit auf Stufe t hängt nur von den Gram–Schmidt–Koeffizienten $\mu_{i,j}$ mit $j < t$ ab. Damit sind die Erfolgswahrscheinlichkeiten der Aufzählung auf den einzelnen Stufen unabhängig voneinander und wir erhalten als Gesamtwahrscheinlichkeit, einen kürzesten Gittervektor zu finden, das Produkt der Wahrscheinlichkeiten für die einzelnen Stufen.

Daraus ergeben sich unmittelbar Strategien zur geschnittenen Aufzählung mit garantierter Erfolgswahrscheinlichkeit $\geq p$:
Bei festem $s \leq m$ fixieren wir die Erfolgswahrscheinlichkeiten p_t auf den Stufen $t = 2, \ldots s$ so, daß die Gesamterfolgswahrscheinlichkeit $\prod_{t=2}^s p_t \geq p$ ist (z.B. durch $p_t := p^{\frac{1}{s-1}}$).

Geschnittene Aufzählung in Breadth–First–Search–Ordnung

Bestimme die Menge B_s aller Knoten auf Stufe s mit $\widetilde{c}_s < \bar{c}_1$ und sortiere nach dem „Gewicht“ $(\bar{c}_1 - \widetilde{c}_s)^{\frac{s-1}{2}}$. $A_s := B_s$.
Entferne den Knoten mit minimalem Gewicht aus A_s, solange
$\Big(\sum\limits_{\widetilde{u}_s \in A_s} (\bar{c}_1 - \widetilde{c}_s)^{\frac{s-1}{2}}\Big) \Big/ \Big(\sum\limits_{\widetilde{u}_s \in B_s} (\bar{c}_1 - \widetilde{c}_s)^{\frac{s-1}{2}}\Big)$ nicht kleiner wird als p_s.
Auf Stufe $t < s$ bestimme die Menge $B_t \subset \mathbb{Z} \times A_{t+1}$ aller Knoten $(\widetilde{u}_t, \ldots, \widetilde{u}_s)$ mit $\widetilde{c}_t < \bar{c}_1$. Sortiere B_t nach dem „Gewicht“ $(\bar{c}_1 - \widetilde{c}_t)^{\frac{t-1}{2}}$. $A_t := B_t$.
Entferne den Knoten mit minimalem Gewicht aus A_t, solange
$\Big(\sum\limits_{(\widetilde{u}_t, \ldots, \widetilde{u}_m) \in A_t} (\bar{c}_1 - \widetilde{c}_t)^{\frac{t-1}{2}}\Big) \Big/ \Big(\sum\limits_{(\widetilde{u}_t, \ldots, \widetilde{u}_m) \in B_t} (\bar{c}_1 - \widetilde{c}_t)^{\frac{t-1}{2}}\Big)$ nicht kleiner wird als p_t.

Diese Vorgehensweise ist insofern optimal, als die Anzahl der aufzuzählenden Knoten bei fest vorgegebener Verteilung der Erfolgswahrscheinlichkeiten auf die einzelnen Stufen minimal ist. Nachteilig wirkt sich zum einen der Aufwand für das Sortieren aller Knoten einer Stufe aus, zum anderen wächst der Hauptspeicherbedarf sehr schnell mit der Blockgröße. Das Verfahren ist daher auf sequentiellen Rechnern nur eingeschränkt verwendbar (bei 64 MB Hauptspeicher bis Blockgröße 50 bei Erfolgswahrscheinlichkeit ≤ 0.5). Für größere Blockgrößen und Erfolgswahrscheinlichkeiten empfiehlt sich die Verwendung eines Vektorrechners, der die implizite Parallelität ausnutzen kann, die bei der Breadth–First–Search Vorgehensweise auftritt. Auf sequentiellen Rechnern liefert die folgende approximative Vorgehensweise ähnlich kleine Aufzählungsbäume bei erheblich kleinerem Hauptspeicheraufwand und ohne Sortierung der Knoten.

Geschnittene Aufzählung in Depth–First–Search–Ordnung

Initialisiere die Variablen $A_t := B_t := 0$ für $t = 2, \ldots, s$.
Für $1 < t \leq s$ und festes $(\widetilde{u}_{t+1}, \ldots, \widetilde{u}_s)$ durchlaufe alle für $\widetilde{u}_t$ möglichen Werte $(\widetilde{c}_t < \bar{c}_1)$ in aufsteigender Ordung bezüglich $\widetilde{c}_t$.
Addiere jeweils das „Gewicht" $(\bar{c}_1 - \widetilde{c}_t)^{\frac{t-1}{2}}$ zu B_t hinzu.
Falls $A_t / B_t \geq p_t$, dann gehe zum nächsten $\widetilde{u}_t$ über (bzw. erhöhe t, falls alle $\widetilde{u}_t$ bereits abgearbeitet sind).
Sonst addiere das Gewicht zu A_t und erniedrige t.
Für $t = 1$ bestimme $\widetilde{u}_1$ mit minimalem $\widetilde{c}_1$.
Falls $\widetilde{c}_1 < \bar{c}_1$, setze $\bar{c}_1 = \widetilde{c}_1$, $(u_1, \ldots, u_m) = (\widetilde{u}_1, \ldots, \widetilde{u}_s, 0, \ldots, 0)$.

5.1.1 Statistische Verifikation der Heuristik

Wir verifizieren die Heuristik für $m = 20$. Hierzu erzeugen wir unabhängig gleichverteilte Gram–Schmidt–Koeffizienten $\mu_{i,j} \in [-0.5, 0.5)$ für $1 \leq j < i \leq 20$ und setzen $c_1 := 1, c_{i+1} := (1 - \mu_{i+1,i}^2) c_i$ für $i = 1, \ldots, 19$. Mit diesen Eingaben durchlaufen wir den (geschnittenen) Suchbaum mit $\bar{c}_1 = 1$ und betrachten die Größe des Suchbaums auf den einzelnen Stufen, die Quotienten der Summen der Gewichte sowie deren Produkt als Erwartungswert für den Anteil der gefundenen kurzen Vektoren. Wir betrachten nur den Teilbaum mit $\widetilde{u}_{20} > 0$ und mitteln die Ergebnisse jeweils über 1000 zufälligen Eingaben.

Schnitt nur auf einer Stufe

Es wird nur auf Stufe t geschnitten. In Tabelle 5.1 wird der mittlere Anteil der gefundenen kurzen Vektoren (kV) sowie der mittlere Anteil der vom ungeschnittenen Suchbaum aufgezählten Knoten (AK) für verschiedene p und t dargestellt.

Breadth–First–Search–Methode										
t	p=0.10		p=0.25		p=0.50		p=0.75		p=0.90	
	kV	AK	kV	AK	kV	AK	kV	AK	kV	AK
2	0.1020	0.9835	0.2517	0.9854	0.5012	0.9890	0.7519	0.9933	0.9014	0.9965
3	0.1020	0.9539	0.2513	0.9581	0.5009	0.9666	0.7507	0.9777	0.9001	0.9870
4	0.1017	0.9064	0.2511	0.9133	0.5002	0.9280	0.7510	0.9485	0.9007	0.9672
5	0.1019	0.8375	0.2513	0.8475	0.5010	0.8695	0.7510	0.9019	0.9003	0.9335
6	0.1019	0.7466	0.2519	0.7600	0.5015	0.7903	0.7503	0.8368	0.9001	0.8842
7	0.1021	0.6374	0.2515	0.6545	0.5007	0.6942	0.7501	0.7562	0.9005	0.8216
8	0.1022	0.5183	0.2522	0.5395	0.5007	0.5890	0.7509	0.6676	0.9008	0.7518
9	0.1029	0.3995	0.2516	0.4252	0.5010	0.4850	0.7509	0.5802	0.9007	0.6830
10	0.1038	0.2914	0.2527	0.3218	0.5023	0.3924	0.7518	0.5043	0.9002	0.6246
11	0.1060	0.2024	0.2529	0.2378	0.5020	0.3195	0.7515	0.4475	0.9010	0.5834
12	0.1080	0.1365	0.2547	0.1774	0.5031	0.2703	0.7515	0.4136	0.9014	0.5631
13	0.1118	0.0935	0.2569	0.1403	0.5040	0.2446	0.7520	0.4024	0.9011	0.5628
14	0.1174	0.0708	0.2607	0.1237	0.5057	0.2395	0.7511	0.4100	0.9012	0.5789
15	0.1261	0.0643	0.2668	0.1239	0.5094	0.2512	0.7530	0.4329	0.9010	0.6072
16	0.1518	0.0771	0.2855	0.1414	0.5187	0.2787	0.7577	0.4682	0.9022	0.6440
17	0.1843	0.1085	0.3093	0.1749	0.5405	0.3248	0.7676	0.5175	0.9066	0.6892
18	0.3002	0.2042	0.3652	0.2397	0.5893	0.4063	0.7935	0.5906	0.9164	0.7460
19	0.5222	0.4158	0.5397	0.4283	0.6708	0.5239	0.8657	0.7245	0.9459	0.8320
20	0.9557	0.8917	0.9557	0.8917	0.9557	0.8917	0.9557	0.8917	0.9675	0.9083

Depth–First–Search–Methode										
t	p=0.10		p=0.25		p=0.50		p=0.75		p=0.90	
	kV	AK	kV	AK	kV	AK	kV	AK	kV	AK
2	0.1014	0.9838	0.2516	0.9862	0.5006	0.9901	0.7507	0.9942	0.9012	0.9970
3	0.1007	0.9550	0.2511	0.9604	0.5015	0.9697	0.7508	0.9805	0.9003	0.9887
4	0.1006	0.9086	0.2512	0.9177	0.5011	0.9346	0.7502	0.9550	0.9008	0.9718
5	0.1012	0.8409	0.2502	0.8547	0.5005	0.8811	0.7514	0.9144	0.9002	0.9434
6	0.1010	0.7514	0.2502	0.7707	0.5015	0.8087	0.7504	0.8582	0.9001	0.9025
7	0.1007	0.6436	0.2504	0.6693	0.5005	0.7210	0.7510	0.7892	0.9006	0.8516
8	0.1009	0.5261	0.2516	0.5585	0.5017	0.6250	0.7497	0.7140	0.9002	0.7960
9	0.1016	0.4087	0.2497	0.4486	0.5006	0.5312	0.7497	0.6413	0.9001	0.7429
10	0.1026	0.3017	0.2510	0.3494	0.5014	0.4491	0.7509	0.5805	0.9005	0.7001
11	0.1023	0.2135	0.2519	0.2691	0.5010	0.3853	0.7508	0.5367	0.9005	0.6718
12	0.1029	0.1477	0.2519	0.2115	0.5014	0.3440	0.7515	0.5132	0.9006	0.6609
13	0.1037	0.1039	0.2512	0.1753	0.5010	0.3218	0.7518	0.5072	0.9010	0.6633
14	0.1066	0.0787	0.2520	0.1576	0.5007	0.3177	0.7502	0.5158	0.9003	0.6774
15	0.1115	0.0684	0.2536	0.1522	0.5003	0.3252	0.7501	0.5326	0.9003	0.6972
16	0.1281	0.0738	0.2579	0.1557	0.5028	0.3375	0.7509	0.5505	0.8998	0.7181
17	0.1763	0.1056	0.2718	0.1725	0.5069	0.3561	0.7524	0.5750	0.9010	0.7424
18	0.2991	0.2037	0.3169	0.2906	0.5234	0.3951	0.7615	0.6023	0.9048	0.7641
19	0.5222	0.4158	0.5222	0.4158	0.5560	0.4629	0.8216	0.6958	0.9337	0.8231
20	0.9557	0.8917	0.9557	0.8917	0.9557	0.8917	0.9557	0.8917	0.9675	0.8917

Tabelle 5.1 Schnitt nur auf einer Stufe

Der mittlere Anteil der gefundenen kurzen Vektoren ist stets größer als die geforderte Erfolgswahrscheinlichkeit p. Der Aufwand für die Aufzählung hängt entscheidend von der Position t des Schnitts ab. Das Verhältnis zwischen dem Aufwand und dem Anteil der gefundenen kurzen Vektorren ist im Bereich $15 \leq t \leq 17$ am günstigsten.

Gleicher Schnitt auf allen Stufen

Es wird auf allen Stufen $t > j$ mit der gleichen Erfolgswahrscheinlichkeit geschnitten, d.h. wir setzen $p_t := p^{1/(m-1)}$. In Tabelle 5.2 wird der mittlere Anteil der gefundenen kurzen Vektoren (kV) sowie der mittlere Anteil der vom ungeschnittenen Baum aufzuzählenden Knoten (AK) für verschiedene p dargestellt.

Breadth–First–Search–Methode								
p	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08
kV	0.02322	0.03724	0.05059	0.06366	0.07543	0.08717	0.09861	0.10998
AK	0.00831	0.01140	0.01417	0.01681	0.01917	0.02149	0.02382	0.02612
p	0.09	0.10	0.15	0.20	0.25	0.30	0.40	0.50
kV	0.12181	0.13317	0.18826	0.23921	0.29237	0.34377	0.44236	0.53946
AK	0.02855	0.03093	0.04267	0.05413	0.06696	0.08026	0.10890	0.14227
p	0.60	0.70	0.75	0.80	0.85	0.90	0.91	0.92
kV	0.63321	0.72773	0.77375	0.81948	0.86533	0.91063	0.91936	0.92839
AK	0.18135	0.23085	0.26086	0.29638	0.33986	0.39688	0.41094	0.42620
p	0.93	0.94	0.95	0.96	0.97	0.98	0.99	1.00
kV	0.93757	0.94656	0.95526	0.96415	0.97312	0.98191	0.99130	1.00000
AK	0.44340	0.46248	0.48388	0.50931	0.54042	0.58047	0.64125	1.00000

Depth–First–Search–Methode								
p	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08
kV	0.02288	0.03683	0.04954	0.06192	0.07363	0.08504	0.09654	0.10836
AK	0.00922	0.01273	0.01580	0.01870	0.02144	0.02400	0.02670	0.02937
p	0.09	0.10	0.15	0.20	0.25	0.30	0.40	0.50
kV	0.11954	0.13027	0.18334	0.23509	0.28689	0.33728	0.43654	0.53310
AK	0.03197	0.03455	0.04736	0.06052	0.07447	0.08908	0.12104	0.15797
p	0.60	0.70	0.75	0.80	0.85	0.90	0.91	0.92
kV	0.62820	0.72303	0.77109	0.81654	0.86344	0.90931	0.91801	0.92783
AK	0.20096	0.25444	0.28733	0.32510	0.37181	0.43223	0.44711	0.46288
p	0.93	0.94	0.95	0.96	0.97	0.98	0.99	1.00
kV	0.93632	0.94585	0.95425	0.96353	0.97296	0.98203	0.99106	1.00000
AK	0.48037	0.49998	0.52156	0.54747	0.57856	0.61897	0.67832	1.00000

Tabelle 5.2 gleicher Schnitt auf allen Stufen

Auch bei der gleichmäßigen Verteilung des Schnitts auf alle Stufen ist der mittlere Anteil der gefundenen kurzen Vektoren stets mindestens so groß wie die geforderte Erfolgswahrscheinlichkeit p. Der Aufwand ist deutlich geringer als beim Schnitt nur auf einer Stufe.

Optimierte Verteilung des Schnitts

Wir verwenden die Gaußsche Volumenheuristik, um a priori die erwartete Größe des ungeschnittenen Suchbaums auf den einzelnen Stufen zu bestimmen. Das erwartete Wachstum des Baums von Stufe t zu Stufe $t-1$ ergibt sich dann als Quotient der entsprechenden Erwartungswerte für die Größe. Die Verteilung der Mißerfolgswahrscheinlichkeiten auf die einzelnen Stufen erfolgt nun proportional zu diesem erwarteten Wachstum. In Tabelle 5.3 wird der mittlere Anteil der gefundenen kurzen Vektoren (kV) sowie der mittlere Anteil der vom ungeschnittenen Baum aufzuzählenden Knoten (AK) für verschiedene p dargestellt.

Breadth–First–Search–Methode								
p	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08
kV	0.02702	0.04289	0.05885	0.07354	0.08775	0.10126	0.11429	0.12673
AK	0.00826	0.01125	0.01406	0.01669	0.01920	0.02167	0.02403	0.02632
p	0.09	0.10	0.15	0.20	0.25	0.30	0.40	0.50
kV	0.13965	0.15403	0.21402	0.27075	0.32522	0.37664	0.47758	0.57214
AK	0.02873	0.03136	0.04329	0.05556	0.06866	0.08188	0.11167	0.14612
p	0.60	0.70	0.75	0.80	0.85	0.90	0.91	0.92
kV	0.66711	0.75478	0.79882	0.83982	0.88292	0.92295	0.93042	0.93824
AK	0.18837	0.23935	0.27120	0.30740	0.35309	0.41190	0.42575	0.44161
p	0.93	0.94	0.95	0.96	0.97	0.98	0.99	1.00
kV	0.94692	0.95454	0.96225	0.96957	0.97698	0.98498	0.99257	1.00000
AK	0.45934	0.47832	0.49987	0.52536	0.55575	0.59571	0.65540	1.00000

Depth–First–Search–Methode								
p	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08
kV	0.02671	0.04235	0.05609	0.06986	0.08430	0.09852	0.11190	0.12372
AK	0.00904	0.01247	0.01531	0.01811	0.02100	0.02382	0.02658	0.02895
p	0.09	0.10	0.15	0.20	0.25	0.30	0.40	0.50
kV	0.13557	0.14869	0.20769	0.26644	0.31859	0.37041	0.47021	0.56605
AK	0.03168	0.03438	0.04735	0.06103	0.07444	0.08878	0.12108	0.15815
p	0.60	0.70	0.75	0.80	0.85	0.90	0.91	0.92
kV	0.66088	0.74777	0.79225	0.83537	0.87929	0.91994	0.92750	0.93638
AK	0.20337	0.25724	0.29141	0.33000	0.37875	0.43984	0.45452	0.47174
p	0.93	0.94	0.95	0.96	0.97	0.98	0.99	1.00
kV	0.94486	0.95316	0.96068	0.96900	0.97669	0.98450	0.99230	1.00000
AK	0.48984	0.51052	0.53288	0.55828	0.58936	0.62912	0.68823	1.00000

Tabelle 5.3 optimierte Verteilung des Schnitts

Durch die optimierte Verteilung des Schnitts wird der Aufwand für die Aufzählung gegenüber der gleichmäßigen Verteilung des Schnitts nochmals verringert.

5.2 Die lokale Gaußsche Volumenheuristik

Schnorr und Hörner [SH95] verwenden in ihrem Aufzählungsalgorithmus eine lokale Variante der Volumenheuristik. Dabei wird ein Knoten auf Stufe t des Suchbaums abgeschnitten, sobald die Wahrscheinlichkeit ω_t, einen kürzeren Vektor durch Aufzählen des zugehörigen Teilbaums zu finden, den Wert 2^{-p} unterschreitet. Die Wahrscheinlichkeit bezieht sich auch hier auf unabhängige, in $[-\frac{1}{2}, \frac{1}{2})$ gleichverteilte Gram–Schmidt–Koeffizienten.

Der Nachteil dieser Variante gegenüber den im vorigen Abschnitt vorgestellten Methoden besteht darin, daß man keine exakte untere Schranke für die Wahrscheinlichkeit angeben kann, mit der ein kürzester Gittervektor gefunden wird. Der Ansatz, die Mißerfolgswahrscheinlichkeit durch die Summe $(\beta - 1)2^{-p}$ der Wahrscheinlichkeiten, in den einzelnen Knoten des Pfades zum absoluten Minimum fälschlicherweise abzuschneiden, scheitert daran, daß die Unabhängigkeit der betrachteten Ereignisse $[\omega_t < 2^{-p}]$ nicht gegeben ist. In praktischen Tests lagen die Erfolgsquoten zum Teil deutlich unter der Schranke $1 - (\beta - 1)2^{-p}$.

In Tabelle 5.4 wird der mittlere Anteil der gefundenen kurzen Vektoren (kV) sowie der mittlere Anteil der vom ungeschnittenen Baum aufzuzählenden Knoten (AK) für verschiedene Parameter p dargestellt.

p	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7
kV	0.09273	0.12467	0.15747	0.19136	0.22504	0.25936	0.29369	0.32801
AK	0.12831	0.13430	0.14046	0.14688	0.15347	0.16029	0.16732	0.17455
p	0.8	0.9	1.0	1.1	1.2	1.3	1.4	1.5
kV	0.36206	0.39531	0.42692	0.45854	0.48896	0.51810	0.54637	0.57330
AK	0.18193	0.18946	0.19704	0.20483	0.21271	0.22067	0.22873	0.23686
p	1.6	1.7	1.8	1.9	2.0	2.5	3.0	3.5
kV	0.59983	0.62434	0.64781	0.67014	0.69155	0.78065	0.84673	0.89401
AK	0.24509	0.25336	0.26173	0.27007	0.27851	0.32060	0.36263	0.40379
p	4.0	4.5	5.0	6.0	7.0	8.0	9.0	10.0
kV	0.92717	0.95051	0.96701	0.98527	0.99337	0.99708	0.99888	0.99954
AK	0.44354	0.48181	0.51826	0.58496	0.64383	0.69505	0.73933	0.77741

Tabelle 5.4 lokale Heuristik

In Abbildung 5.1 sind die Anteile am ungeschnittenen Aufzählungsbaum in Abhängigkeit von den Anteilen der gefundenen kurzen Vektoren für die gleichmäßige Verteilung des Schnitts bei der Depth–First–Search–Methode (durchgezogene Linie), bei der Breadth–First–Search–Methode (gestrichelte Linie) und der lokalen Heuristik (punktierte Linie) dargestellt.

Es wird deutlich, daß die lokale Heuristik für sehr kleine Erfolgswahrscheinlichkeiten (Anteile der gefundenen kurzen Vektoren) sehr schlecht ist. Für große Erfolgswahrscheinlichkeiten nähert sich der Anteil der aufzuzählenden Knoten immer mehr der Breadth–First–Search–Methode an, bei der wiederum etwa 10% weniger Knoten aufgezählt werden müssen als bei der Depth–First–Search–Methode. Für Erfolgswahrscheinlichkeiten nahe bei 1 ist der Aufwand der lokalen Methode sogar geringfügig kleiner als bei der Breadth–First–Search–Methode. Dies ist dadurch zu erklären, daß bei der lokalen Heuristik bereits nahezu alle Knoten mit Höhenquadrat $< c_1$ aufgezählt werden und in einigen Fällen zum Abbruch führen, während bei den Stufenmethoden stets auch noch der erste Knoten mit Höhenquadrat $\geq c_1$ aufgezählt wird.

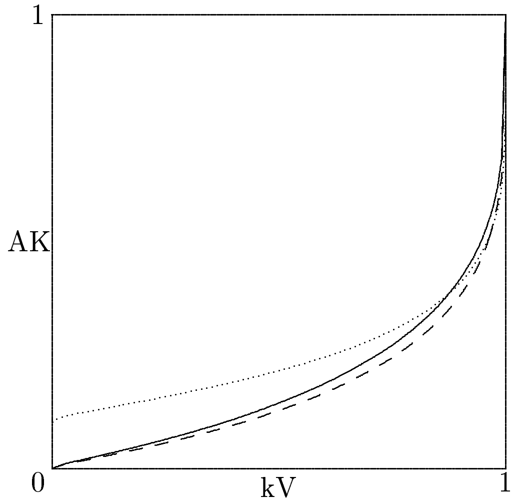

Abbildung 5.1 Stufenheuristik und lokale Heuristik

In Abbildung 5.2 sind die Anteile am ungeschnittenen Aufzählungsbaum in Abhängigkeit von den Anteilen der gefundenen kurzen Vektoren für die optimierte Verteilung des Schnitts bei der Depth–First–Search–Methode (durchgezogene Linie) und der Breadth–First–Search–Methode (gepunktete Linie) der Kurve für die gleichmäßige Verteilung des Schnitts bei der Breadth–First–Search–Methode (gestrichelte Linie) gegenübergestellt.

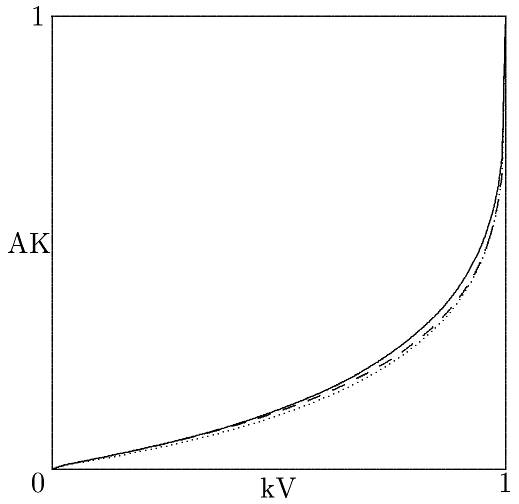

Abbildung 5.2 optimierte Stufenheuristik

Die optimierte Breadth–First–Search–Methode benötigt für feste Erfolgsquoten stets den geringsten Anteil am ungeschnittenen Suchbaum. Die Breadth–First–Search–Methode mit gleichmäßiger Verteilung des Schnitts ist für kleine Erfolgswahrscheinlichkeiten nahezu gleichwertig mit der optimierten Depth–First–Search–Methode. Für große Erfolgswahrscheinlichkeiten nähert sich der Aufwand der optimierten Breadth–First–Search–Methode immer mehr dem Aufwand für die Breadth–First–Search–Methode mit gleichmäßiger Verteilung an.

Kapitel 6

Anwendungen

Die in diesem Kapitel vorgestellten Algorithmen sind in der Programmiersprache C unter dem Betriebssystem HP–UX 9.05, einem UNIX–Derivat der Firma HP (Hewlett Packard) auf HP–Workstations 'Apollo 9000' der Serien 710, 715/50, 735/99 und 735/125 implementiert. Tabelle 6.1 gibt für die einzelnen Serien jeweils die Taktfrequenz (MHz) des Prozessors sowie die theoretisch erreichbare Anzahl der floating point Operationen (MFLOPs) und Instruktionen (MIPs) in Millionen pro Sekunde an.

Serie	(MHz)	MFLOPs	MIPs
710	50	12.2	57
715/50	50	13.2	62
735/99	99	40.8	124
735/125	125	57.4	154

Tabelle 6.1 Leistungsdaten der HP–Worksations

Für die Berechnungen in ganzzahliger Arithmetik wird die in der Arbeitsgruppe Mathematische Informatik der Universität Frankfurt entwickelte Programmbibliothek LARIFARI verwendet.

6.1 Lösung von Rucksackproblemen

Ein *Rucksackproblem* besteht darin, zu gegebenen natürlichen Zahlen $n,\ a_1, \dots, a_n$ und s einen 0–1–Vektor $(x_1, \dots, x_n)$ zu finden mit $\sum_{i=1}^n x_i a_i = s$ oder zu zeigen, daß kein solcher Vektor existiert. Wir nennen 0–1–Vektoren $x = (x_1, \dots, x_n)$ mit $\sum_{i=1}^n x_i a_i = s$ *Lösungen* des Rucksackproblems $(n, a_1, \dots, a_n, s)$. Das zugehörige Entscheidungsproblem, ob eine Lösung existiert oder nicht, ist NP–vollständig [EB81].

Einen Ansatz zur Lösung von Rucksack–Problemen liefert die Theorie der Gitterbasenreduktion in euklidischer Norm. Die *Dichte* d eines Rucksackproblems ist definiert als

$$d := \frac{n}{\log_2 \max\limits_{1 \le i \le n} a_i}.$$

Lagarias und Odlyzko [LO85] reduzieren ein gegebenes Rucksackproblem $(n, a_1, \dots, a_n, s)$ darauf, einen kurzen Gittervektor bezüglich der euklidischen Norm zu bestimmen. Sie zeigen, daß bei Dichte $d < 0.645$ „fast immer" ein kürzester Gittervektor einer Lösung des Rucksackproblems zugeordnet werden kann, sofern es überhaupt eine Lösung besitzt. Dabei bedeutet „fast immer", daß für hinreichend großes n und zufällige lösbare Rucksackprobleme $(n, a_1, \dots, a_n, s)$ die Wahrscheinlickeit der Existenz eines von 0 verschiedenen Gittervektors mit Norm kleiner der minimalen Norm eines Lösungsvektors exponentiell klein ist (in n). Durch Verwendung einer besser konditionierten Gitterbasis konnte die Schranke für die Dichte auf $d < 0.9408$ erhöht werden [CJL$^+$92].

Sowohl Schnorr und Euchner [SE94] als auch Hörner [Hör94] verwenden zur Lösung des Rucksackproblems Gitterbasenreduktionsalgorithmen bezüglich der euklidischen Norm. Schnorr und Euchner verwenden die Gitterbasis

$$B := (b_1, \dots, b_{n+1})^T := \begin{pmatrix} 2 & & 0 & na_1 & 0 \\ & \ddots & & \vdots & \vdots \\ 0 & & 2 & na_n & 0 \\ 1 & \cdots & 1 & ns & 1 \end{pmatrix}.$$

Jeder Gittervektor $z = (z_1, \dots, z_{n+2}) \in L(b_1, \dots, b_{n+1})$ mit $|z_{n+2}| = 1, z_{n+1} = 0$ und $z_i \in \{0, 1\}$ für $i = 1, \dots, n$ liefert eine Lösung $x = (x_1, \dots, x_n)$ des Rucksackproblems $(n, a_1, \dots, a_n, s)$. Dabei ist $x_i = \frac{1}{2}|z_i - z_{n+2}|$. Zum Auffinden dieser Vektoren wird der Algorithmus SCHNORR–EUCHNER (siehe unten) verwendet. Anstelle der Längenreduktionen in der (β, δ)–Blockreduktion werden L^3–Reduktionen mit δ und 5 tiefen Einfügungen durchgeführt. Nach jeder Längenreduktion (innerhalb der L^3–Reduktionen) wird der neue Basisvektor auf Lösung getestet. Für große Blockgrößen wird in der Blockreduktion eine geschnittene Aufzählung verwendet. Bei der Aufzählung wird der Teilbaum mit Wurzel $(\tilde{u}_t, \dots, \tilde{u}_k)$ abgeschnitten, sobald $\tilde{c}_t > c_j \min(1.05(k - t + 1)/(k - j), 1)$ gilt (Details siehe [SE94]).

Algorithmus SCHNORR–EUCHNER
EINGABE: $n, a_1, \dots, a_n, s$
1. Berechne die Basis $B = (b_1, \dots, b_{n+1})^T$
2. Permutiere $b_1, \dots, b_{n+1}$ zufällig so, daß die b_j mit $b_{j,n+2} \neq 0$ vorne stehen.
3. (β, δ)–blockreduziere $b_1, \dots, b_{n+1}$.
4. (Paare–Reduktion)
   F := TRUE
   WHILE F=TRUE
      F := FALSE
      Sortiere $b_1, \dots, b_{n+1}$ so, daß $\|b_1\|_2 \leq \|b_2\|_2 \leq \dots \leq \|b_{n+1}\|_2$
      FOR $j = 1, \dots, n+1$
         FOR $k = 1, \dots, j-1$
            IF $\|b_j \pm b_k\|_2 < \|b_j\|_2$
            THEN $b_j := b_j \pm b_k$
               F := TRUE

Wiederhole Schritte 2–4 15 mal.
Der Algorithmus stoppt, sobald eine Lösung gefunden wurde.

Hörner [Hör94] geht bei seinem Algorithmus davon aus, daß die Anzahl q der Einsen im Lösungsvektor bekannt ist. (Diese Zusatzinformation ist häufig verfügbar; ansonsten sind maximal alle $n+1$ Möglichkeiten für q zu testen.) Als Gitterbasis verwendet er

$$B := (b_1, \dots, b_{n+1})^T := \begin{pmatrix} 1 & q & \cdots & q & n^2 s & n^2 q \\ 0 & n & & 0 & n^2 a_1 & n^2 \\ \vdots & & \ddots & & \vdots & \vdots \\ 0 & 0 & & n & n^2 a_n & n^2 \end{pmatrix}.$$

Jeder Gittervektor $z = (z_1, \dots, z_{n+3})$ mit $z_{n+2} = z_{n+3} = 0$, $|z_1| = 1$ und $z_i \in \{z_1 q, z_1(q-n)\}$ für $i = 2, \dots, n+1$ liefert eine Lösung $x = (x_1, \dots, x_n)$ des Rucksackproblems. Dabei ist $x_i = \frac{z_1 q - z_{i+1}}{n}$. Im Algorithmus HÖRNER (siehe unten) werden anstelle der Längenreduktionen in der (β, δ)–Blockreduktion L^3–Reduktionen mit δ und 5 tiefen Einfügungen durchgeführt. Nach jeder Längenreduktion (innerhalb der L^3–Reduktionen) wird der neue Basisvektor auf Lösung getestet. Dabei entfällt ab Schritt 4 der Test auf $z_{n+2} = z_{n+3} = 0$. Der Algorithmus stoppt, sobald eine Lösung gefunden wurde.

Am Ende jeder Runde wird in Schritt 6 eine geschnittene Aufzählung über die gesamte Basis (d.h. mit $j = 1, k = n-1$) durchgeführt. Diese Aufzählung wird beendet, sobald ein Vektor gefunden wurde, dessen Normquadrat nicht größer ist als $n(n-q)q+1$ (dies ist das Normquadrat eines Lösungsvektors). Auf den einzelnen Stufen t wird abgeschnitten, sobald $\widetilde{c}_t > \alpha_t(n(n-q)q+1)$ ist. Die Werte für α_t werden a priori berechnet und fallen linear mit t (es gilt $\alpha_1 = 1$). Als weiteres Abbruchkriterium dienen die Zahl und die zulässigen Positionen der „Verzweigungen". Eine Verzweigung auf Stufe t liegt vor, wenn für festes $(\widetilde{u}_{t+1}, \dots, \widetilde{u}_k)$ mehr als ein Wert für $\widetilde{u}_t$ durchlaufen wird. Nähere Einzelheiten dieser Schnittmethode finden sich in [Hör94].

Algorithmus HÖRNER
EINGABE: $n, a_1, \dots, a_n, s, q$
1. Berechne die Basis $B = (b_1, \dots, b_{n+1})^T$
2. L^3–reduziere $b_1, \dots, b_{n+1}$ mit $\delta = 0.99$
3. Falls genau 2 Basisvektoren b_i, b_j existieren mit $b_{i,n+2}b_{j,n+3} - b_{i,n+3}b_{j,n+2} \neq 0$ und $b_{k,n+2} = b_{k,n+3} = 0$ für $k \neq i, j$, dann entferne die Vektoren b_i und b_j sowie die letzten beiden Spalten aus der Basis (die entfernten Vektoren können in keiner Linearkombination enthalten sein, die einer Lösung zugeordnet ist).
   Andernfalls stoppe mit einer Fehlermeldung.
   (Dieser Fall trat bei keinem der durchgeführten Tests auf.)
4. Permutiere $b_1, \dots, b_{n-1}$ zufällig so, daß die b_j mit $b_{j,1} \neq 0$ vorne stehen.
5. (β, δ)–blockreduziere $b_1, \dots, b_{n-1}$ mit $\delta = 0.99$.
6. führe eine geschnittene Aufzählung über die gesamte Basis durch.

Wiederhole Schritte 4–6 15 mal.

Der Nachteil der Algorithmen SCHNORR–EUCHNER und HÖRNER besteht darin, daß das Auffinden einer vorhandenen Lösung nicht garantiert werden kann. Die Erfolgswahrscheinlichkeit hängt entscheidend von der Dichte des Rucksackproblems ab (siehe Kapitel 6.1.1).

Das Entscheidungsproblem der Existenz einer Lösung ist nur dann mittels Gitterbasenreduktion vollständig zu lösen, wenn am Ende der Algorithmen eine vollständige Aufzählung aller Gittervektoren durchgeführt wird, deren Norm gleich der Norm eines Lösungsvektors ist. Eine solche Aufzählung bezüglich der euklidischen Norm ist jedoch für Dimension $n > 50$ nicht mehr in akzeptabler Rechenzeit möglich. Das Aufzählungsverfahren ENUM_∞ (siehe Seite 43) erlaubt uns nun, das Entscheidungsproblem zumindest bis zur Dimension $n = 66$ vollständig und effizient zu lösen.

Wir betrachten das durch die folgende Basis $B = (b_1, \dots, b_{n+1})^T$ aufgespannte Gitter L:

$$B := \begin{pmatrix} 1 & 1 & \cdots & 1 & 2s \\ 0 & 2 & & 0 & 2a_1 \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & & 2 & 2a_n \end{pmatrix}.$$

Offensichtlich gilt:

$$\lambda_{1,\|\cdot\|_\infty}(L) = 1 \Leftrightarrow \text{ das Rucksackproblem } (n, a_1, \dots, a_n, s) \text{ ist lösbar.} \tag{6.1}$$

Jeder Gittervektor $z = (z_1, \dots, z_{n+2})$ mit $\|z\|_\infty = 1$ liefert eine Lösung $x = (x_1, \dots, x_n)$ des Rucksackproblems. Dabei ist $x_i = \frac{1}{2}|z_1 - z_{i+1}|$. Wir verwenden das effiziente Aufzählungsverfahren ENUM_∞, um einen Gittervektor b mit $\|b\|_\infty = 1$ zu finden oder zu zeigen, daß $\lambda_{1,\|\cdot\|_\infty}(L) > 1$ gilt.

Die Laufzeitanalysen aus Kapitel 4 zeigen, daß eine „möglichst gute“ Vorreduktion bezüglich der euklidischen Norm erforderlich ist, um die abschließende Aufzählung effizient durchführen zu können. Wir verwenden hierzu das folgende deterministische Verfahren:

Algorithmus RUCKSACK
EINGABE: $n, a_1, \dots, a_n, s$
1. Berechne die Basis $B = (b_1, \dots, b_{n+1})^T$.
2. L^3–reduziere $b_1, \dots, b_{n+1}$ mit $\delta = 0.99$.
3. Tranformiere die Basis so, daß $b_{i,n+2} = 0$ für $i \leq n$ und $b_{n+1,n+2} \neq 0$. Entferne den Vektor b_{n+1} und die letzte Spalte aus der Basis. (Die Transformation erfolgt mittels Gauß–Elimination bezüglich der letzten Spalte. Der entfernte Vektor kann nicht in eine Linearkombination eingehen, die einer Lösung des Rucksackproblems zugeordnet ist.)
4. (β, δ)–blockreduziere $b_1, \dots, b_n$ mit $\delta = 0.99$.
5. rufe Algorithmus ENUM_∞ auf.

AUSGABE: Gittervektor b mit $\|b\|_\infty = 1$ oder FALSE, falls das Rucksackproblem nicht lösbar ist.

Anstelle der Längenreduktionen in der (β, δ)–Blockreduktion werden L^3–Reduktionen mit δ und 5 tiefen Einfügungen durchgeführt. Nach jeder Längenreduktion (innerhalb der L^3–Reduktionen) wird getestet, ob der neue Basisvektor l_∞–Norm 1 besitzt. In diesem Fall stoppt der Algorithmus.

Im Algorithmus ENUM_∞ wird c mit 1 initialisiert, die Funktion SCHNITT_∞ verwendet nur den Vektor $(\lambda_t, \dots, \lambda_s) = (1, 0, \dots, 0)$.

6.1.1 Versuchsergebnisse

In den folgenden Tabellen werden die erzielten Ergebnisse den Ergebnissen von Schnorr–Euchner [SE94] und Hörner [Hör94] gegenübergestellt. Für jede Dimension n und jede Bitlänge b (der Gewichte a_i) wurden zufällig 20 lösbare Rucksackprobleme erzeugt, indem jeweils $x \in_R \{z \in \{0,1\}^n | \sum_{i=1}^n x_i = q := \frac{n}{2}\}$ und die Gewichte $a_1, \dots, a_n \in_R [0, 2^b)$ unabhängig und gleichverteilt gewählt und $s := \sum_{i=1}^n a_i x_i$ gesetzt wurde. Für $n = 42, 50$ und 58 wurde stets als Blockgröße $\beta = 20$ verwendet. In Dimension $n = 66$ verwendet der Schnorr–Euchner–Algorithmus $\beta = 50$ mit geschnittener Aufzählung, während in den beiden anderen Algorithmen mit $\beta = 30$ blockreduziert wird. Für die Algorithmen von

Schnorr–Euchner und Hörner sind in den nachfolgenden Tabellen jeweils die Anzahl der gefundenen Lösungen in der ersten Runde (Erf. 1) und insgesamt (Erf. ges.), die Gesamtzahl der durchlaufenen Runden (Runden) sowie die mittlere Laufzeit (Zeit) des Algorithmus aufgeführt. Der Algorithmus RUCKSACK durchläuft stets nur eine Runde. Daher sind anstelle der Spalten für die Erfolge in der ersten Runde und die Rundenzahl Spalten für die Anzahl der Erfolge innerhalb der Blockreduktion (Erf. Block) sowie in der vollständigen Aufzählung ENUM_∞ (Erf. Enum) aufgelistet. Die letzte Zeile jeder Tabelle ($\sum$) enthält die Gesamtzahl der Erfolge bzw. Runden für alle Testläufe in der jeweiligen Dimension. Die Versuche wurden auf HP–Workstations der Serien 715/50 und 710 durchgeführt. Die Statistik von Schnorr–Euchner wurde auf anderen Rechnern erstellt; die angegebenen Laufzeiten sind entsprechend umgerechnet.

		SCHNORR–EUCHNER				HÖRNER				RUCKSACK			
n	b	Erf. 1	Erf. ges.	Run-den	Zeit m:ss	Erf. 1	Erf. ges.	Run-den	Zeit m:ss	Erf. Block	Erf. Enum	Erf. ges.	Zeit m:ss
42	24	20	20	20	0:03	20	20	20	0:02	20	0	20	0:02
42	28	18	20	22	0:10	19	20	21	0:04	19	1	20	0:04
42	32	20	20	20	0:12	20	20	20	0:04	20	0	20	0:04
42	36	15	19	45	0:30	19	20	21	0:08	18	2	20	0:09
42	40	18	20	30	0:28	16	20	39	0:15	18	2	20	0:08
42	44	19	20	25	0:19	20	20	20	0:08	19	1	20	0:07
42	48	20	20	20	0:11	20	20	20	0:08	20	0	20	0:06
42	52	20	20	20	0:09	20	20	20	0:07	20	0	20	0:06
42	56	20	20	20	0:08	20	20	20	0:07	20	0	20	0:06
42	60	20	20	20	0:08	20	20	20	0:08	20	0	20	0:07
Σ		190	199	242		194	200	221		194	6	200	

		SCHNORR–EUCHNER				HÖRNER				RUCKSACK			
n	b	Erf. 1	Erf. ges.	Run-den	Zeit m:ss	Erf. 1	Erf. ges.	Run-den	Zeit m:ss	Erf. Block	Erf. Enum	Erf. ges.	Zeit m:ss
50	26	20	20	20	0:09	20	20	20	0:04	20	0	20	0:03
50	30	20	20	20	0:18	17	20	23	0:07	19	1	20	0:09
50	34	18	20	22	0:32	19	20	21	0:11	20	0	20	0:08
50	38	17	20	25	1:02	16	20	50	0:30	18	2	20	0:21
50	42	14	19	53	2:03	19	20	21	0:20	16	4	20	0:25
50	46	10	20	77	3:13	13	18	61	0:55	12	8	20	0:37
50	50	16	19	41	1:46	20	20	20	0:32	14	6	20	0:27
50	54	19	20	22	1:08	17	20	28	0:26	20	0	20	0:21
50	58	20	20	20	0:50	20	20	20	0:21	20	0	20	0:15
50	62	20	20	20	0:35	20	20	20	0:19	20	0	20	0:15
50	66	20	20	20	0:29	20	20	20	0:18	20	0	20	0:16
50	70	20	20	20	0:24	20	20	20	0:18	20	0	20	0:15
Σ		214	238	360		221	238	324		219	21	240	

		SCHNORR–EUCHNER				HÖRNER				RUCKSACK			
n	b	Erf. 1	Erf. ges.	Runden	Zeit mm:ss	Erf. 1	Erf. ges.	Runden	Zeit mm:ss	Erf. Block	Erf. Enum	Erf. ges.	Zeit mm:ss
58	29	20	20	20	00:23	19	20	21	00:07	20	0	20	00:06
58	35	17	20	26	01:15	19	20	21	00:16	20	0	20	00:16
58	41	10	20	34	03:13	8	20	37	00:51	12	8	20	04:08
58	47	10	17	89	06:36	10	18	75	02:24	12	8	20	01:19
58	53	6	15	130	12:44	11	16	117	03:23	5	15	20	05:11
58	58	2	16	155	15:03	11	20	37	02:09	8	12	20	02:09
58	63	15	20	35	04:59	19	20	21	01:06	13	7	20	01:21
58	69	19	20	21	02:51	20	20	20	01:06	18	2	20	00:47
58	75	20	20	20	01:55	20	20	20	00:48	20	0	20	00:40
58	81	20	20	20	01:08	20	20	20	00:41	20	0	20	00:34
58	87	20	20	20	00:48	20	20	20	00:37	20	0	20	00:33
58	93	20	20	20	00:46	20	20	20	00:36	20	0	20	00:33
Σ		179	228	590		197	234	429		188	52	240	

		SCHNORR–EUCHNER				HÖRNER				RUCKSACK			
n	b	Erf. 1	Erf. ges.	Runden	Zeit h:mm:ss	Erf. 1	Erf. ges.	Runden	Zeit h:mm:ss	Erf. Block	Erf. Enum	Erf. ges.	Zeit h:mm:ss
66	26	20	20	20	0:00:14	20	20	20	0:00:05	20	0	20	0:00:05
66	34	20	20	20	0:01:34	20	20	20	0:00:24	20	0	20	0:00:29
66	42	20	20	20	0:06:22	20	20	20	0:01:59	17	3	20	0:13:39
66	50	10	19	78	0:36:08	10	18	76	0:10:16	11	9	20	1:50:54
66	58	9	14	119	1:28:10	7	19	83	0:26:51	13	7	20	1:11:45
66	66	10	19	70	1:14:17	16	20	26	0:22:26	12	8	20	1:06:17
66	72	18	20	26	0:31:21	18	20	22	0:10:08	17	3	20	0:18:42
66	80	20	20	20	0:15:16	20	20	20	0:04:45	19	1	20	0:04:05
66	88	20	20	20	0:14:28	20	20	20	0:02:13	20	0	20	0:02:20
66	96	20	20	20	0:11:28	20	20	20	0:01:37	20	0	20	0:01:30
Σ		167	192	413		171	197	327		169	31	200	

Beim Vergleich der Ergebnisse fällt auf, daß der Schnorr–Euchner–Algorithmus stets die wenigsten Lösungen findet und die größte Laufzeit benötigt. Bis einschließlich Dimension 50 ist der deterministische Algorithmus RUCKSACK der schnellste. Erst bei Dimension 66 ist der Hörner–Algorithmus deutlich schneller, dafür aber nicht immer erfolgreich.

Inzwischen wurden von Schnorr und Hörner [SH95] mit Hilfe der lokalen Gaußschen Volumenheuristik noch schnellere probabilistische Algorithmen entwickelt, die bei allen Experimenten bis Dimension 66 eine Lösung des Rucksackproblems gefunden haben. Diese Algorithmen könnten auch bei den deterministischen Algorithmen anstelle der Blockreduktion mit Blockweite 20 (30) zur besseren Vorreduktion verwendet werden. Damit würde auch die abschließende vollständige Aufzählung beschleunigt werden. (Entsprechende Versuche wurden aus Zeitgründen nicht mehr durchgeführt.)

6.2 Angriffe auf Rucksack–Kryptosysteme

Wegen der NP–Vollständigkeit des Rucksackproblems bestand die Hoffnung, sichere und effiziente Public–Key Kryptosysteme auf der Grundlage dieses Problems entwickeln zu können. Bei Public–Key Kryptosystemen übermittelt Alice die Nachricht m verschlüsselt an Bob, indem sie m mit dem öffentlichen Schlüssel K_B von Bob kodiert und das Ergebnis $E_{K_B}(m)$ an Bob übermittelt. Nur Bob ist in der Lage, mit Hilfe seines geheimen Schlüssels S_B die Nachricht m aus $E_{K_B}(m)$ zu rekonstruieren. Bei einem Rucksack–Kryptosystem besteht der öffentliche Schlüssel im wesentlichen aus einer Menge $a_1, \ldots, a_n$ von „schwierigen" Gewichten. Alice verschlüsselt eine Nachricht $m = m_1 \cdots m_n \in \{0,1\}^n$, indem sie $s := \sum_{i=1}^n m_i a_i$ berechnet und s an Bob sendet. Zur Berechnung der verschlüsselten Nachricht sind lediglich linear viele Additionen erforderlich. Damit ist das Verschlüsseln sehr effizient. Bob verwendet seinen geheimen Schlüssel, um das „schwierige" Rucksackproblem $(n, a_1, \ldots, a_n, s)$ in ein „einfaches" Rucksackproblem $(n, a'_1, \ldots, a'_n, s' = \sum_{i=1}^n m_i a'_i)$ zu transformieren und daraus die ursprüngliche Nachricht m zu rekonstruieren. Ein solches „einfaches" Rucksackproblem ist zum Beispiel eine Folge $a'_1, \ldots, a'_n$ mit $a'_1 = 1$ und $a'_i > \sum_{j=1}^{i-1} a'_j$. Für diese Gewichte ist die Lösung m stets in linearer Zeit zu bestimmen.

Eines der ersten Rucksack–Kryptosysteme wurde von Merkle und Hellman [MH78] vorgeschlagen. Der öffentliche Schlüssel des Merkle–Hellman–Kryptosystems besteht aus einem Rucksack mit kleiner Dichte. Das System wurde von Shamir [Sha82] und Brickell [Bri84] gebrochen.

Sowohl Chor und Rivest [CR88] als auch Orton [Ort94] schlagen ein Rucksack–Kryptosystem mit großer Dichte vor, um die „Low–density–Attacken" [Sha82, Bri84, LO85, CJL$^+$92] abzuwehren. In Kapitel 6.2.1 und 6.2.2 stellen wir erfolgreiche Angriffe gegen diese beiden Systeme vor.

6.2.1 Angriff auf das Kryptosystem von Chor und Rivest

Chor und Rivest [CR88] verwenden n speziell konstruierte Gewichte a_i und kodieren eine binäre Nachricht $x = x_1 \cdots x_n$ mit q Einsen durch $s := \sum_{i=1}^n x_i a_i$. Die Gewichte sind so konstruiert, daß die Kodierung der Nachricht injektiv ist, d.h. zwei verschiedenen Nachrichten werden verschiedene Summen s zugeordnet. Die Bitlänge der Gewichte ist so gewählt, daß insbesondere die Low–Density–Attacken von Lagarias–Odlyzko [LO85] scheitern.

Als Beispielparameter wurden $n = 103, q = 12$ gegeben. Dieses Beispiel wird ausführlich in der Diplomarbeit von Hörner [Hör94] behandelt und erfolgreich angegriffen.

Die Ausnutzung der speziellen Struktur des Rucksackproblems führt zu einer erheblichen Beschleunigung der erfolgreichsten Angriffe. Es ist auch erstmals gelungen, sämtliche 50 zufällig erzeugten Nachrichten mit durchschnittlicher Rechenzeit von weniger als 9 Stunden auf den HP–Workstations der Serien 715/50 und 710 zu entschlüsseln. Die maximale Rechenzeit für eine Nachricht lag bei 34.23 Stunden.

Wir betrachten das durch die folgende Basis aufgespannte Gitter:

$$B := (b_1, \ldots, b_{n+1})^T := \begin{pmatrix} 1 & q & q & \cdots & q & n^2 s & n^2 q \\ 0 & n & 0 & \cdots & 0 & n^2 a_1 & n^2 \\ 0 & 0 & n & & 0 & n^2 a_2 & n^2 \\ \vdots & & \ddots & \ddots & \vdots & \vdots & \vdots \\ 0 & \cdots & & & n & n^2 a_n & n^2 \end{pmatrix}.$$

Jeder Gittervektor $v = (v_0, \ldots, v_{n+2}) = \sum_{i=1}^{n+1} u_i b_i$ mit $v_0 = \pm 1, v_{n+1} = v_{n+2} = 0$ liefert die entschlüsselte Nachricht x, falls die Koeffizienten v_j genau q mal den Wert $v_0 * (q - n)$ und $n - q$ mal den Wert $v_0 * q$ haben. Es gilt dann

$$x_i = \begin{cases} 1, & \text{falls } v_i = v_0 * (q - n), \\ 0, & \text{falls } v_i = v_0 * q. \end{cases} \tag{6.2}$$

Die spezielle Form des Lösungsvektors wird nun in Verbindung mit den Techniken aus Kapitel 5 dazu ausgenutzt, den folgenden Angriff von Hörner [Hör94] zu verbessern.

Algorithmus Angriff auf das Chor–Rivest–Kryptosystem
EINGABE: $n, a_1, \ldots, a_n, s, q$
1. Berechne die Basis $B = (b_1, \ldots, b_{n+1})^T$
2. L^3–reduziere $b_1, \ldots, b_{n+1}$ mit $\delta = 0.99$
3. Transformiere die Basis so, daß gilt:
   $b_{i,n+1} = b_{i,n+2} = 0$ für $i = 1, \ldots, n-1$,
   $b_{n,n+1} \neq 0,\ b_{n,n+2} = 0$,
   $b_{n+1,n+2} \neq 0$.
   Entferne b_n und b_{n+1} aus der Basis.
   (Die entfernten Vektoren können in keine Linearkombination der Basisvektoren eingehen, die zur Lösung führt.)
4. Permutiere die Vektoren $b_1, \ldots, b_{n-1}$ zufällig, so daß die b_j mit $b_{j,0} \neq 0$ vorne stehen.
5. (β, δ)–blockreduziere $b_1, \ldots, b_{n-1}$ mit $\beta = 50$ und $\delta = 0.9$.
6. Führe eine geschnittene Aufzählung über die gesamte Basis durch.

AUSGABE: entschlüsselte Nachricht x oder Fehlermeldung

Bei der Blockreduktion wird anstelle einer Längenreduktion stets eine L^3–Reduktion mit $\delta = 0.99$ und 5 tiefen Einfügungen durchgeführt. Bei den Aufzählungen innerhalb der Blockreduktion wird mit der lokalen Volumenheuristik und $p = 13$ geschnitten (siehe Kapitel 5.2). Für $j \leq 5$ wird der aufgezählte Vektor b_j^{neu} bereits dann in die Basis eingefügt, wenn

$c_j^{\text{neu}} < 0.99 c_j$ gilt. Nach jedem Längenreduktionsschritt innerhalb der L^3–Reduktionen wird der neue Vektor auf Lösung getestet.

Die abschließende Aufzählung über die gesamte Basis (Schritt 6) verwendet die folgende adaptive Schnittmethode: Zunächst werden 50 zufällig und unabhängig gewählte Nachrichten mit $a_1, \ldots, a_n$ verschlüsselt und anschließend die ersten 5 Schritte des Angriffs durchgeführt. Da die Nachrichten bekannt sind, können für jeden Lösungsvektor v die Höhenquadrate $\widetilde{c}_t := \|\pi_t(v)\|_2^2$ bezüglich der Ausgabebasis $b_1, \ldots, b_{n-1}$ für $t = 1, \ldots, n-1$ bestimmt werden. Die Mittelwerte auf den einzelnen Stufen t werden in den Variablen α_t gespeichert. Bei der eigentlichen Aufzählung (zum Entschlüsseln einer unbekannten Nachricht) wird ein Teilbaum mit Wurzel $(\widetilde{u}_t, \ldots, \widetilde{u}_{n-1})$ abgeschnitten, sobald $\widetilde{c}_t \geq \zeta_t := (nq(n-q)+1)\min(1.00001, \alpha_t + add + 0.00001)$ gilt. Dabei ist add eine addidtive Konstante. In seiner Diplomarbeit verwendet Hörner die Werte $add = 0.08$ und $add = 0.13$. Für $add = 0.08$ liefert der Algorithmus eine Erfolgsquote von 52% bei einer durchschnittlichen Laufzeit von 1.90 Stunden. Bei $add = 0.13$ wurden 88% der Probleme in durchschnittlich 6.58 Stunden gelöst. Wir verwenden für den verbesserten Angriff die Konstanten $add = 0.13$ und $add = 0.20$. Die genaue Analyse der Struktur des Lösungsvektors liefert einen zusätzlichen deterministischen Schnitt.

Wir nehmen im folgenden an, daß $v := \sum_{i=1}^{n-1} u_i b_i$ die entschlüsselte Nachricht liefert.

Mit $v_t := \pi_t(v)$ gilt

$$< v_t, v_t > \;=\; < v_1, v_t > \;=\; v_{1,0} v_{t,0} + \sum_{i=1}^{n} v_{1,i} v_{t,i}. \tag{6.3}$$

Wir betrachten zunächst die q–elementige Menge $I := \{i : x_i = 1\}$ der Indizes, an denen die Nachrichtenbits 1 sind. Mit (6.2) folgt

$$\bar{c}_t := \; < v_t, v_t > \;=\; v_{1,0}\Big(v_{t,0} + q\sum_{i=1}^{n} v_{t,i} - n\sum_{i\in I} v_{t,i}\Big). \tag{6.4}$$

Nach Schritt 3 gilt stets

$$\sum_{i=1}^{n} b_{t,i} = \sum_{i=1}^{n} \hat{b}_{t,i} = 0 \tag{6.5}$$

wegen $0 = b_{t,n+2} = n\sum_{i=1}^{n} b_{t,i}$ für $t = 1, \ldots, n-1$. Wir erhalten

$$\bar{c}_t = v_{1,0}\Big(v_{t,0} - n\sum_{i\in I} v_{t,i}\Big). \tag{6.6}$$

Für die im Laufe der Aufzählung auftretenden Koeffizientenvektoren $(\widetilde{u}_t, \ldots, \widetilde{u}_{n-1})$ betrachten wir die orthogonalen Projektionen $w_t := \pi_t(\sum_{i=t}^{n-1} \widetilde{u}_i b_i)$. Im Teilbaum mit Wurzel $(\widetilde{u}_t, \ldots, \widetilde{u}_{n-1})$ kann nur dann der Lösungsvektor aufgezählt werden, wenn

$$\widetilde{c}_t := \|w_t\|_2^2 = v_{1,0}\Big(w_{t,0} - n\sum_{i\in I} w_{t,i}\Big). \tag{6.7}$$

Die Indexmenge I ist zum Zeitpunkt der Aufzählung nicht bekannt. Wir betrachten daher stattdessen die beiden Mengen $I_g := \{i_1, \ldots, i_q : 1 \leq i_j \leq n,\ w_{t,i_j} \geq w_{t,l}\ \forall l \notin I_g\}$ und $I_k := \{i_1, \ldots, i_q : 1 \leq i_j \leq n,\ w_{t,i_j} \leq w_{t,l}\ \forall l \notin I_k\}$ der Indizes mit den q größten bzw. kleinsten Werten.

Für $v_{1,0} = 1$ ergibt sich

$$\widetilde{c}_t \geq w_{t,0} - n \sum_{i \in I_g} w_{t,i}, \tag{6.8}$$

$$\widetilde{c}_t \leq w_{t,0} - n \sum_{i \in I_k} w_{t,i}, \tag{6.9}$$

für $v_{1,0} = -1$

$$\widetilde{c}_t \geq -w_{t,0} + n \sum_{i \in I_k} w_{t,i}, \tag{6.10}$$

$$\widetilde{c}_t \leq -w_{t,0} + n \sum_{i \in I_g} w_{t,i}. \tag{6.11}$$

Die Ungleichungen (6.8) und (6.10) sind i.a. trivial, da die rechten Seiten meist negativ sind. Für den Schnitt betrachten wir daher nur die Ungleichungen (6.9) und (6.11). Für jeden Knoten $(\widetilde{u}_t, \ldots, \widetilde{u}_{n-1})$ mit $\widetilde{c}_t < \zeta_t$ stellen wir die beiden folgenden Hypothesen auf:

H_t^+: $v_{1,0} = 1$ und $(\widetilde{u}_t, \ldots, \widetilde{u}_{n-1}) = (u_t, \ldots, u_{n-1})$,

H_t^-: $v_{1,0} = -1$ und $(\widetilde{u}_t, \ldots, \widetilde{u}_{n-1}) = (u_t, \ldots, u_{n-1}))$.

Die Hypothese H_t^+ (H_t^-) kann nur dann korrekt sein, wenn alle Hypothesen H_i^+ (H_i^-) korrekt sind für $i = t+1, \ldots, n-1$. Wir können daher H_t^+ (H_t^-) bereits für alle Koeffizienten $\widetilde{u}_t$ verwerfen, wenn H_{t+1}^+ (H_{t+1}^-) verworfen wurde. Außerdem verwerfen wir H_t^+ (H_t^-), falls Ungleichung (6.9) ((6.11)) verletzt ist. Analog zum deterministischen Schnitt aus Lemma 3.3 können wir die Suche in Richtung $w_t - w_{t+1}$ abbrechen, sobald beide Hypothesen für $(\widetilde{u}_t, \ldots, \widetilde{u}_{n-1})$ verworfen wurden. Zum Beweis unterscheiden wir zwei Fälle. Für $(\widetilde{u}_{t+1}, \ldots, \widetilde{u}_{n-1}) \neq (u_{t+1}, \ldots, u_{n-1})$ kann der Abbruch nicht zum „Verlust“ der Lösung führen. Für $(\widetilde{u}_{t+1}, \ldots, \widetilde{u}_{n-1}) = (u_{t+1}, \ldots, u_{n-1})$ gilt $\widetilde{c}_{t+1} = v_{1,0}(w_{t+1,0} - n \sum_{i \in I} w_{t+1,i})$. Alle wegen des Abbruchs nicht aufgezählten Koeffizienten $\widetilde{u}_t'$ würden einen Vektor w_t' liefern mit $w_t' = w_{t+1} + \lambda(w_t - w_{t+1})$ für ein $\lambda > 1$. Es folgt $\widetilde{c}_t' = \|\pi_t(w_t')\|_2^2 < v_{1,0}(w_{t,0}' - n \sum_{i \in I} w_{t,i}')$, d.h. H_t^+ und H_t^- würden verworfen.
Die Effizienz der Tests hängt entscheidend vom Algorithmus zur Bestimmung von I_g bzw. I_k ab. Hierzu wird das verbesserte Auswahlverfahren verwendet, wodurch der Test in $O(n) + O(q \log n)$ Schritten durchgeführt werden kann. Durch Mitführen der Indizes können wir zunächst die Ungleichung auf Stufe t mit den Indizes von Stufe $t+1$ testen. Die Indexmenge muß nur dann neu berechnet werden, wenn die Ungleichung mit der (möglicherweise nicht völlig korrekten) Indexmenge der höheren Stufe nicht erfüllt ist. Da sich die

Indexmengen beim Übergang zur nächstniedrigeren Stufe in der Regel kaum oder gar nicht ändern, ersparen wir uns dadurch sehr häufig deren explizite Bestimmung. Wir fixieren einzelne Indizes, sobald die zugehörigen Koeffizienten hinreichend groß (klein) sind und damit nicht mehr durch einen anderen Koeffizienten ersetzt werden können, ohne die Ungleichung zu verletzen (d.h. die fixierten Indizes müssen in I enthalten sein, falls H_t^+ (H_t^-) korrekt ist).

Wegen $v_t - v_{t+1} = \hat{b}_t(u_t + y_t)$ ist $(u_t + y_t)^2 \|\hat{b}_t\|_2^2 = v_{1,0}(u_t + y_t)(\hat{b}_{t,0} - n \sum_{i \in I} \hat{b}_{t,i})$. Wir können daher die Intervalle, in denen $\widetilde{u}_t + y_t$ liegen muß, a priori beschränken durch

$$\widetilde{u}_t + y_t \quad \leq \quad \begin{cases} \hat{b}_{t,0} - n \sum\limits_{i \in I_k} \hat{b}_{t,i} & \text{, falls } v_{1,0} = 1, \\ -\hat{b}_{t,0} + n \sum\limits_{i \in I_g} \hat{b}_{t,i} & \text{, falls } v_{1,0} = -1, \end{cases} \tag{6.12}$$

$$\widetilde{u}_t + y_t \quad \geq \quad \begin{cases} \hat{b}_{t,0} - n \sum\limits_{i \in I_g} \hat{b}_{t,i} & \text{, falls } v_{1,0} = 1, \\ -\hat{b}_{t,0} + n \sum\limits_{i \in I_k} \hat{b}_{t,i} & \text{, falls } v_{1,0} = -1. \end{cases} \tag{6.13}$$

Gegenüber einer vollständigen Aufzählung (in einer (n+1)–dimensionalen Kugel mit Radius $\sqrt{nq(n-q)+1}$) zählen wir durch diesen deterministischen Schnitt nur noch in der Vereinigung von $2\binom{n}{q}$ Kugeln mit Radius $\frac{1}{2}\sqrt{nq(n-q)+1}$ auf. Der Volumenanteil beträgt also (wegen der Überlappung der kleinen Kugeln) weniger als $\binom{n}{q} * 2^{-n}$. Für $n = 103, q = 12$ ergibt sich ein Anteil von $1.5 * 10^{-16}$ (für $n = 197, q = 24 \ : \ 2.17 * 10^{-29}$). Bei heuristisch geschnittener Aufzählung ist dieses Verhältnis weniger günstig, da die kleinen Kugeln die inneren Schichten der großen Kugel stärker überdecken als die äußeren. Dennoch lassen sich schon bei relativ starkem heuristischem Schnitt die Laufzeiten der Algorithmen durch zusätzliches deterministisches Schneiden reduzieren. Dadurch ist es erstmals möglich, auf HP–Workstations der Serie 715/50 alle 50 zufällig erzeugten Probleme der Dimension 103 mit durchschnittlicher Laufzeit von weniger als 9 Stunden zu lösen. Die maximale Rechenzeit liegt bei 34.23 Stunden. Außerdem wurden 88 % der Probleme mit durchschnittlicher Rechenzeit unter 4 Stunden gelöst. Ohne den deterministischen Schnitt wurden hierfür 6.5 Stunden benötigt. Die für die praktische Anwendung vorgeschlagenen Parameter $n = 197, q = 24$ wurden bisher aus Speicherplatz– und Rechenzeitgründen nicht verwendet. Stattdessen wurden von Hörner [Hör94] einige Experimente für $n = 151, q = 16$ durchgeführt. Hierbei wurden (ohne Verwendung des deterministischen Schnitts) 5 von 50 Nachrichten mit Laufzeiten zwischen einer und 249 Stunden auf HP–Workstations der Serie 735 entschlüsselt. Hochrechnungen für $n = 197, q = 24$ auf der Basis dieser Experimente lassen vermuten, daß mit den neuen Algorithmen eine Laufzeit von wenigen Wochen für eine Erfolgsquote von 5% ausreicht.

6.2.2 Angriff auf das Orton–Kryptosystem

Orton [Ort94] schlägt ein Public–Key–Kryptosystem vor, das auf dichten, kompakten und modularen Rucksackproblemen basiert. Ein Rucksackproblem heißt *kompakt*, wenn zu gegebenen natürlichen Zahlen $n, a_1, \ldots, a_n$ und y anstelle eines 0–1–Vektors $(x_1, \ldots, x_n)$ mit $\sum_{i=1}^{n} x_i a_i = y$ ein ganzzahliger Vektor $(x_1, \ldots, x_n) \in [0, 2^s - 1]^n$ zugelassen wird. Das Rucksackproblem heißt *modular*, wenn die Gleichheit $\sum_{i=1}^{n} x_i a_i = y$ nur modulo einer gegebenen Zahl q gelten muß. Im folgenden brechen wir das Orton–Kryptosystem mit Hilfe der Aufzählung in l_∞–Norm (siehe hierzu auch [Rit96]).

Wir beschreiben zunächst kurz das Orton–Schema; für eine detaillierte Darstellung verweisen wir auf [Ort94].

Öffentliche Parameter: natürliche Zahlen r, n, s. (Nachrichten bestehen aus n Blöcken mit jeweils s Bit; r ist die Anzahl der Runden für die Erzeugung der Schlüssel.)

Geheimer Schlüssel: ganze Zahlen $a_i^{(0)}$ mit $a_1^{(0)} = 1$, $a_i^{(0)} > (2^s - 1) \sum_{j=1}^{i-1} a_j^{(0)}$ für $i = 1, \ldots, n$ und natürliche Zahlen q_2, $p^{(k)}, w^{(k)}$ für $k = 1, \ldots, r$, wobei $q_1 := p^{(r)}/q_2 \in \mathbb{Z}$.

Der Teil $\{a_i^{(0)}\}$ des geheimen Schlüssels repräsentiert einen „einfachen" Rucksack. Durch die folgenden Operationen wird er in einen „schwierigen" Rucksack transformiert, der durch den öffentlichen Schlüssel repräsentiert wird.

$$\begin{aligned}
a_i^{(k)} &:= a_i^{(k-1)} w^{(k)} \bmod p^{(k)} \text{ für } i = 1, \ldots, n+k-1, \quad a_{n+k}^{(k)} := -p^{(k)}, \\
f_i^{(k)} &:= 2^{-\operatorname{prec}(k)} \lfloor a_i^{(k)} 2^{\operatorname{prec}(k)} / p^{(k)} \rfloor \text{ für } i = 1, \ldots, n+k-1, \quad k = 1, \ldots, r, \\
a_{i,j} &:= a_i^{(r)} \bmod q_j \text{ für } i = 1, \ldots, n+r-1, \ j = 1, 2.
\end{aligned}$$

Hierbei wird die geheime „Falltür" q_2, $p^{(k)}, w^{(k)}$ $(k = 1, \ldots, r)$ verwendet. $\operatorname{prec}(k)$ ist die Bitgenauigkeit für die Berechnung der Quotienten $f_i^{(k)}$ in der k–ten Runde, d.h. die Binärdarstellung der rationalen Zahlen $f_i^{(k)}$ wird nach $\operatorname{prec}(k)$ Nachkommabits abgeschnitten. Orton schlägt vor, $\operatorname{prec}(k) = s + \log_2 n + k + 2$ zu verwenden. Diese Wahl garantiert eine eindeutige Entschlüsselung und verhindert gleichzeitig bekannte Attacken von Brickell [Bri84] und Shamir [Sha79].

Öffentlicher Schlüssel: natürliche Zahlen q_1, $\operatorname{prec}(k)$ für $k = 1, \ldots, r-1$,
nichtnegative ganze Zahlen $a_{i,j}$ für $i = 1, \ldots, n+r-1$, $j = 1, 2$,
rationale Zahlen $f_i^{(k)} \in 2^{-\operatorname{prec}(k)}[0, 2^{\operatorname{prec}(k)})$ für $k = 1, \ldots, r-1$, $i = 1, \ldots, n+k-1$.

VERSCHLÜSSELUNG
EINGABE: öffentlicher Schlüssel, Nachricht $(x_1, \ldots, x_n) \in [0, 2^s)^n$
1. $x_{n+k} := \left\lfloor \sum_{i=1}^{n+k-1} x_i f_i^{(k)} \right\rfloor$ für $k = 1, \ldots, r-1$
2. $y_1 := \sum_{i=1}^{n+r-1} x_i a_{i,1} \bmod q_1$, $\quad y_2 := \sum_{i=1}^{n+r-1} x_i a_{i,2}$

AUSGABE: verschlüsselte Nachricht (y_1, y_2)

ENTSCHLÜSSELUNG
EINGABE: öffentlicher und geheimer Schlüssel, verschlüsselte Nachricht (y_1, y_2)
1. rekombiniere $y^{(r)} \equiv y_j \bmod q_j$ $(j = 1, 2)$ mit Chinesischem Restsatz:
   $y^{(r)} := q_2\Big((y_1 - y_2)q_2^{-1} \bmod q_1\Big) + y_2$
2. $y^{(k-1)} := y^{(k)}(w^{(k)})^{-1} \bmod p^{(k)}$ für $k = r, \ldots, 1$
3. löse $\sum_{i=1}^n x_i a_i^{(0)} = y^{(0)}$ mit $x_i \in [0, 2^s)$.
   (dies ist einfach wegen $a_i^{(0)} > (2^s - 1) \sum_{j=1}^{i-1} a_j^{(0)}$)

AUSGABE: entschlüsselte Nachricht $(x_1, \ldots, x_n)$

Der Angriff mit l_∞–Norm kürzesten Gittervektoren

Wir ordnen dem Entschlüsselungsproblem ein ganzzahliges Gitter zu, so daß aus jedem Gittervektor mit l_∞–Norm 1 die entschlüsselte Nachricht rekonstruiert werden kann. Aus der gegebenen Basis des Gitters berechnen wir mit dem L^3–Algorithmus eine reduzierte Basis, die es uns erlaubt, mittels vollständiger Aufzählung effizient einen Vektor mit l_∞–Norm 1 zu finden.

Wir beschreiben das Entschlüsselungsproblem zunächst als (nichtlineares) Gleichungssystem: Zu gegebenem öffentlichen Schlüssel und einer verschlüsselten Nachricht (y_1, y_2) sind ganze Zahlen $x_1, \ldots, x_n \in [0, 2^s)$, $x_{n+k} \in [0, 2^{s+k+\log_2 n-1})$ zu finden mit

$$\sum_{i=1}^{n+r-1} x_i a_{i,1} = y_1 \bmod q_1, \tag{6.14}$$

$$\sum_{i=1}^{n+r-1} x_i a_{i,2} = y_2, \tag{6.15}$$

$$x_{n+k} = \Big\lfloor \sum_{i=1}^{n+k-1} x_i f_i^{(k)} \Big\rfloor \quad \text{für } k = 1, \ldots, r-1. \tag{6.16}$$

Wir transformieren die Gleichungen (6.14)–(6.16) in ein System von $r+1$ ganzzahligen linearen Gleichungen mit $m := ns + (r-1)(r/2 + s + \lceil \log_2 n \rceil - 1) + \sum_{k=1}^{r-1} \mathrm{prec}(k)$ 0–1–Variablen (siehe (6.19) unten).

$f_i^{(k)} 2^{\mathrm{prec}(k)} \in [0, 2^{\mathrm{prec}(k)})$ ist ganzzahlig. Daher können wir (6.16) schreiben als

$$x_{n+k} 2^{\mathrm{prec}(k)} = \sum_{i=1}^{n+k-1} x_i f_i^{(k)} 2^{\mathrm{prec}(k)} - x_{n+r+k-1} \quad \text{für } k = 1, \ldots, r-1, \tag{6.17}$$

wobei die zusätzlichen Variablen $x_{n+r+k-1}$ ganze Zahlen in $[0, 2^{\mathrm{prec}(k)})$ sind.

Mit $a_{i,k+2} := f_i^{(k)} 2^{\mathrm{prec}(k)}$ für $i = 1, \ldots, n+k-1$, $a_{n+k,k+2} := -2^{\mathrm{prec}(k)}$, $a_{n+r+k-1,k+2} := -1$ und $a_{i,k+2} := 0$ sonst vereinfachen sich die Ungleichungen (6.17) zu

$$\begin{aligned} \sum_{i=1}^{n+2r-2} x_i a_{i,k+2} &= 0 \quad \text{für } k = 1, \ldots, r-1 \\ \text{mit} \qquad & x_{n+r+k-1} \in [0, 2^{\mathrm{prec}(k)}) \quad \text{für } k = 1, \ldots, r-1. \end{aligned} \tag{6.18}$$

Die eindeutige Lösung von (6.14),(6.15),(6.18) läßt sich direkt auf die eindeutige Lösung von (6.14)–(6.16) transformieren.

Wir erhalten 0–1–Variablen, indem wir die binäre Repräsentation der ganzzahligen Variablen verwenden. Hierzu definieren wir rekursiv

$$D_i := \sum_{j=1}^{i-1} d_j \text{ mit } d_i := \begin{cases} s & \text{für } 1 \leq i \leq n, \\ s + i + \lceil \log_2 n \rceil - n - 1 & \text{für } n+1 \leq i \leq n+r-1, \\ \mathrm{prec}(i - (n+r-1)) & \text{für } n+r \leq i \leq n+2r-2. \end{cases}$$

Sei $(t_{D_i+1}, \ldots, t_{D_i+d_i}) \in \{0,1\}^{d_i}$ die Binärdarstellung von x_i, d.h. $x_i = \sum_{l=0}^{d_i-1} t_{D_i+l+1} 2^l$. Wir setzen $A_{D_i+l+1,j} := a_{i,j} 2^l$ für $i = 1, \ldots, n+2r-2$, $j = 1, \ldots, r+1$, $l = 0, \ldots, d_i - 1$, wobei $a_{i,1} := a_{i,2} := 0$ für $i > n+r-1$.
Mit $y_3 := \ldots := y_{r+1} := 0$ vereinfachen sich die Ungleichungen (6.14),(6.15),(6.18) zu

$$\begin{aligned} \sum_{i=1}^{m} t_i A_{i,1} &= y_1 + z q_1, \\ \sum_{i=1}^{m} t_i A_{i,j} &= y_j \quad \text{für } j = 2, \ldots, r+1, \\ \text{mit} \qquad & t_i \in \{0,1\}, \quad z \in \mathbb{Z}. \end{aligned} \tag{6.19}$$

Wir betrachten die Zeilenvektoren $b_1, \ldots, b_{m+2} \in \mathbb{Z}^{m+r+2}$ der folgenden Matrix (6.20) als Basis des Gitters L.

$$B := \begin{pmatrix} 0 & 2 & 0 & \cdots & 0 & NA_{1,1} & NA_{1,2} & \cdots & NA_{1,r+1} \\ 0 & 0 & 2 & \ddots & 0 & NA_{2,1} & NA_{2,2} & \cdots & NA_{2,r+1} \\ \vdots & \vdots & \ddots & \ddots & \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 0 & 2 & NA_{m,1} & NA_{m,2} & \cdots & NA_{m,r+1} \\ 0 & 0 & \cdots & 0 & 0 & Nq_1 & 0 & \cdots & 0 \\ 1 & 1 & \cdots & 1 & 1 & Ny_1 & Ny_2 & \cdots & Ny_{r+1} \end{pmatrix} \tag{6.20}$$

Für jede ganze Zahl $N \geq 2$ gilt:

Jeder Gittervektor $v = (v_0, \ldots, v_{m+r+1}) = \sum_{i=1}^{m+2} c_i b_i \in L$ mit l_∞–norm 1 ist ein l_∞–Norm–kürzester von Null verschiedener Gittervektor und hat die Form $\{\pm 1\}^{m+1} \times 0^{r+1}$, wobei

$c_1, \ldots, c_m \in \{0, -c_{m+2}\}$. Die Nullen in den letzten $r+1$ Koeffizienten implizieren

$$\sum_{i=1}^{m} c_i A_{i,1} + c_{m+2} y_1 = 0 \bmod q_1 \text{ und} \tag{6.21}$$

$$\sum_{i=1}^{m} c_i A_{i,j} + c_{m+2} y_j = 0 \quad \text{für } j = 2, \ldots, r+1. \tag{6.22}$$

Mit $t_i := |c_i| = (|v_i - v_0|)/2$ für $i = 1, \ldots, m$ erhalten wir die eindeutige Lösung von (6.19), die wir direkt in die entschlüsselte Nachricht transformieren können.

Praktischer Algorithmus zum Brechen des Orton–Kryptosystems

Wir verwenden eine leicht modifizierte Version des Algorithmus ENUM$_\infty$ zur Bestimmung des Vektors v, der die Originalnachricht repräsentiert. Wegen $\|v\|_2^2 = m+1$ und $\|v\|_\infty = 1$ können wir die Abfrage „IF $\widetilde{c}_t < (R(\|.\|_\infty c)^2$“ durch „IF $\widetilde{c}_t \leq m+1$“ ersetzen und den Algorithmus abbrechen, sobald wir einen Gittervektor mit l_∞–Norm 1 gefunden haben. Zusätzlich zum Abbruchkriterium (3.6) beenden wir die Aufzählung für $\widetilde{u}_t$, sobald wir einen Index $j \in [0, m]$ gefunden haben mit $b_{i,j} = 0$ für $i = 1, \ldots, t-1$ und $b_{t,j} \neq 0$, $|w_{t,j}| \neq 1$. Dadurch verfehlen wir die Lösung nicht, denn für alle möglichen Wahlen von $\widetilde{u}_1, \ldots, \widetilde{u}_{t-1}$ ist $w_{1,j} = w_{t,j} \neq \pm 1$.

Algorithmus zum Angriff auf das Orton–Kryptosystem
EINGABE: öffentlicher Schlüssel, verschlüsselte Nachricht (y_1, y_2)
1. erzeuge die Gitterbasis $b_1, \ldots, b_{m+2}$ mit $N := n^2$ gemäß (6.20)
2. L^3–reduziere $b_1, \ldots, b_{m+2}$ mit $\delta = 0.99$
3. rufe ENUM$_\infty$ auf; wir erhalten einen Vektor v mit $\|v\|_\infty = 1$
4. $x_i := \sum_{l=0}^{s-1} |v_{s(i-1)+l+1} - v_0| 2^{l-1}$ für $i = 1, \ldots, n$

AUSGABE: entschlüsselte Nachricht $(x_1, \ldots, x_n)$

Die Vektoren $b_1, \ldots, b_{m+1}$ der Basis (6.20) hängen nur vom öffentlichen Schlüssel ab. Daher können wir die L^3–reduzierte Basis $b'_1, \ldots, b'_{m+1}$ von $b_1, \ldots, b_{m+1}$ a priori für jeden öffentlichen Schlüssel berechnen und zwischenspeichern. Für jede mit diesem öffentlichen Schlüssel kodierte Nachricht können wir dann die Vektoren $b'_1, \ldots, b'_{m+1}$ zusammen mit b_{m+2} anstelle der Basis (6.20) verwenden und ersparen dadurch einen Großteil der Rechenzeit.

Praktische Resultate: In Tabelle 6.2 sind die von Orton in [Ort94] vorgeschlagenen Parameter (r, n, s) zusammen mit den Größen der daraus resultierenden Gitterbasen B zusammengestellt. Die Spalte T gibt die Anzahl der Rechenschritte für den stärksten bekannten Angriff [Bri84, Sha79] an, wie sie in [Ort94] berechnet wurde.

r	n	s	T	Größe von B
3	200	1	2^{100}	246×249
4	3	150	2^{91}	1379×1383
4	4	170	2^{104}	1729×1733
5	2	150	2^{91}	1534×1539
5	2	170	2^{101}	1734×1739
5	3	170	2^{104}	1912×1917

Tabelle 6.2 Parameter für das Orton–Kryptosystem

Wir erzeugen zufällig 10 öffentliche Schlüssel für die Parameter $(r, n, s) = (3, 200, 1)$. Für jeden dieser Schlüssel verschlüsseln wir unabhängig 10 zufällige Nachrichten $(x_1, \dots, x_{200}) \in \{0,1\}^{200}$. Anschließend rekonstruieren wir die Nachrichten unter Verwendung des öffentlichen Schlüssels und der verschlüsselten Nachricht. In Tabelle 6.3 sind die durchschnittlichen, minimalen und maximalen Rechenzeiten des Angriffs, der Vorreduktion von $b_1, \dots, b_{m+1}$ und des Angriffs nach der Vorreduktion zusammengestellt. Alle Zeiten sind in Minuten auf einer HP–Workstation der Serie 735/99 angegeben. Die Algorithmen haben stets die Nachricht korrekt entschlüsselt.

Algorithmus	mittl. Laufzeit	min. Laufzeit	max. Laufzeit
Angriff	10.15	8.69	13.79
L^3–Reduktion von $b_1, \dots, b_{m+1}$	9.00	8.52	9.44
Angriff nach Vorreduktion	1.48	0.29	5.23

Tabelle 6.3 experimentelle Ergebnisse

Erste Experimente haben gezeigt, daß für die übrigen Parameter aus Tabelle 6.2 die Nachricht in weniger als 30 Minuten entschlüsselt werden kann, nachdem eine Vorreduktion durchgeführt wurde. Diese Vorreduktion benötigt weniger als 12 Stunden.

Wir konnten die 3 Herausforderungen von Orton [Ort96] mit den Parametern $(r, n, s) = (4, 2, 130)$, $(5, 2, 150)$ und $(5, 2, 170)$ erfolgreich bestehen. Die benötigten Rechenzeiten lagen ebenfalls unter 30 Minuten nach einer Vorreduktion von weniger als 12 Stunden.

Für alle bisher durchgeführten Experimente mit $s \geq 130$ reichte bereits der L^3–Algorithmus aus, um die Nachricht zu entschlüsseln. Demgegenüber war für $s = 1$ stets die Aufzählung nötig.

6.3 Angriff auf die Damgård–Hashfunktionen

Als *Hashfunktionen* bezeichnen wir Funktionen h, die Zahlen beliebiger Bitlänge auf Zahlen mit fester Bitlänge abbilden. Für solche Funktionen existieren offensichtlich *Kollisionen*, d.h. Eingaben $x \neq y$ mit $h(x) = h(y)$. In vielen kryptographischen Protokollen werden Hashfunktionen benötigt, für die keine Kollisionen konstruiert werden können. Damgård [Dam89] hat hierfür unter anderem Hashfunktionen auf der Basis von Rucksackproblemen vorgeschlagen. Es werden 256 zufällige Gewichte a_i der Bitlänge 120 und ein zufälliger 128–Bit–Startwert $(x_1, \ldots, x_{128})$ gewählt. Den 128–Bit–Hashwert eines k*128–Bit–Strings erhalten wir durch k–fache Anwendung der Vorschrift

$$x := \sum_{i=1}^{128} x_i a_i + \sum_{i=1}^{128} y_i a_{128+i} \bmod 2^{128}.$$

Dabei durchläuft y sukzessive alle k 128–Bit–Blöcke und x ist der Ausgabewert der vorherigen Runde. Zum Brechen der Hashfunktion genügt es, eine Kollision zu finden, d.h. eine 0–±1–Lösung x der Gleichung $\sum_{i=1}^{n} x_i a_i = 0$ für $n \leq 256$ zufällige Gewichte a_i der Bitlänge 120. Wir betrachten das durch die folgende Basis $B = (b_1, \ldots, b_n)^T$ aufgespannte Gitter:

$$B := \begin{pmatrix} 1 & 0 & \cdots & 0 & 2a_1 \\ 0 & 1 & \ddots & \vdots & 2a_2 \\ \vdots & \ddots & \ddots & 0 & \vdots \\ 0 & \cdots & 0 & 1 & 2a_n \end{pmatrix}.$$

Jeder Gittervektor $z = (z_1, \ldots, z_{n+1})$ mit $\|z\|_\infty = 1$ liefert unmittelbar eine Kollision der Hashfunktion.

Joux und Stern [JS94] haben gezeigt, daß bereits für $n = 80$ mit sehr großer Wahrscheinlichkeit Kollisionen der Hashfunktion existieren. Die erste Kollision für Damgårds Hashfunktion wurde von Joux und Granboulan [JG94] gefunden. Mit Hilfe der lokalen Gaußschen Volumenheuristik konnten Schnorr und Hörner [SH95] den Angriff erheblich beschleunigen. Erste Experimente zeigen, daß die Kombination des Schnorr–Hörner–Angriffs mit dem deterministischen Schnitt für die Suche nach Gittervektoren mit l_∞–Norm 1 den Angriff nochmals verbessert.

6.4 Konstruktion von t–Designs

Ein *t–Design*, oder genauer ein *t–(v,k,λ)–Design*, ist ein Paar $(V,\mathcal{B})$, wobei $\mathcal{B}$ ein System von k–elementigen Teilmengen (den sogenannten *Blöcken*) einer v–elementigen Menge V ist, so daß jede t–elementige Teilmenge von V in genau λ Blöcken aus $\mathcal{B}$ enthalten ist. Das t–Design heißt *einfach*, wenn kein Block mehrfach in $\mathcal{B}$ enthalten ist. Wenn jede k–elementige Teilmenge von V gleich oft in $\mathcal{B}$ enthalten ist, dann heißt das t–Design *trivial.* Ein t–(v,k,λ)–Design mit $\lambda = 1$ heißt *Steiner–System*.

Aus einem einfachen t–(v,k,λ)–Design erhalten wir einen nichtlinearen binären Code C mit konstantem Gewicht k und Wortlänge v, indem wir jedem Block den zugehörigen Adjazenzvektor zuordnen:

$$\begin{aligned} \mathcal{B} \ni b &= (b_{i_1},\ldots,b_{i_k}) \mapsto c = (c_1,\ldots,c_v) \in C, \\ c_i &:= \begin{cases} 1, & \text{falls } i \in \{i_1,\ldots,i_k\}, \\ 0, & \text{sonst.} \end{cases} \end{aligned}$$

Die Minimaldistanz zweier Codewörter ist damit mindestens 2, die Anzahl der Codewörter ist $\lambda\binom{v}{k}$.

Für allgemeine t ist die Konstruktion von einfachen, nichttrivialen t–(v,k,λ)–Designs schwierig. Eine notwendige Bedingung für die Existenz eines einfachen t–(v,k,λ)–Designs ist

$$\lambda\binom{v-i}{t-i} \equiv 0 \left(\mathrm{mod}\binom{k-i}{t-i}\right) \qquad \text{für} \quad i = 0,\ldots,t. \tag{6.23}$$

(Für einen Beweis siehe z.B. [KM76].)

Einen direkten Ansatz zur Konstruktion aller einfachen t–(v,k,λ)–Designs liefert die Matrix $M^v_{t,k} := \left(m^v_{T,K}\right)$ mit $m^v_{T,K} := \begin{cases} 1, & \text{falls } T \subset K, \\ 0, & \text{sonst.} \end{cases}$

Dabei durchläuft T (K) alle t– $(k$–$)$ elementigen Teilmengen der v–elementigen Menge V.

Jedem einfachen t–(v,k,λ)–Design entspricht eineindeutig eine 0–1–Lösung x des linearen Gleichungssystems

$$M^v_{t,k}x = (\lambda,\ldots,\lambda)^T. \tag{6.24}$$

Die Matrix $M^v_{t,k}$ besitzt $\binom{v}{t}$ Zeilen und $\binom{v}{k}$ Spalten. Damit besteht das zu lösende Gleichungssystem aus $\binom{v}{t}$ Gleichungen mit $\binom{v}{k}$ Variablen und ist damit bereits für kleine Parameter v,k,t zu groß, um durch systematisches Aufzählen Lösungen zu finden. (Für $v = 33, k = 8$ und $t = 7$ erhalten wir z.B. ein Gleichungssystem mit $\binom{33}{7} = 4272048$ Gleichungen und $\binom{33}{8} = 13884156$ Variablen.)

Dennoch lassen sich t–(v,k,λ)–Designs durch Lösen von linearen Gleichungssystemen finden, indem wir zusätzliche Bedingungen an das gesuchte t–Design stellen und dadurch die

Dimension des Gleichungssystems erheblich erniedrigen. Dadurch kann es zwar passieren, daß existierende t–(v,k,λ)–Designs nicht gefunden werden, aber in einigen Fällen führt dieses Vorgehen zum Ziel. Wir betrachten die zusätzliche Bedingung, daß eine gegebene Untergruppe A der symmetrischen Gruppe S_V in der Automorphismengruppe $Aut(V,\mathcal{B})$ des gesuchten Designs enthalten ist. (S_V bezeichnet die Gruppe der Permutationen der Elemente von V. Eine Permutation π ist in $Aut(V,\mathcal{B})$ enthalten, wenn jedes Element von $\mathcal{B}$ durch die Anwendung von π auf ein Element von $\mathcal{B}$ abgebildet wird. Wir sagen „A operiert auf V und induziert eine Operation auf $\mathcal{B}$".)

Sei $X \subset V$. Dann heißt die Menge $O_A(X) := \{Y \subset V : \exists \pi \in A \text{ mit } Y = \pi(X)\}$ *Bahn* oder *Orbit* von X unter der Operation von A auf V. Die Menge V_i aller i–elementigen Teilmengen von V ist die disjunkte Vereinigung von $\rho(i)$ Bahnen $O_i^j := O_A(I_j)$ mit i–elementigen Teilmengen $I_1,\dots,I_{\rho(i)}$ von V, $V_i = \bigcup\limits_{j=1}^{\rho(i)} O_i^j$.

Offensichtlich bewirkt die zusätzlich geforderte Eigenschaft des gesuchten t–Designs, daß mit jedem Block $b \in \mathcal{B}$ auch die gesamte Bahn $O_A(b)$ in $\mathcal{B}$ liegt.

Wir ersetzen die $\binom{v}{t} \times \binom{v}{k}$–Matrix $M_{t,k}^v$ durch die $\rho(t) \times \rho(k)$–Matrix $M_{t,k}^A := \left(m_{i,j}^A\right)$ mit

$$m_{i,j}^A := \left|\{K \in O_k^j : T_i \subset K\}\right|, \quad T_i \in O_t^i \text{ beliebig}, \quad i = 1,\dots,\rho(t), \quad j = 1,\dots,\rho(k).$$

Kramer und Mesner [KM76] haben gezeigt, daß genau dann ein einfaches t–(v,k,λ)–Design mit Automorphismengruppe A existiert, wenn es eine 0–1–Lösung x des linearen Gleichungssystems

$$M_{t,k}^A x = (\lambda,\dots,\lambda)^T \tag{6.25}$$

gibt. Die Matrix $M_{t,k}^A$ heißt daher auch *Kramer–Mesner–Matrix*.

Magliveras und Leavitt [ML84] konstruieren mit Hilfe der Kramer–Mesner–Matrix einfache, nichttriviale 6–(33,8,36)–Designs mit Automorphismengruppe $\mathbf{P\Gamma L_2(32)}$. Die Gruppe $\mathbf{P\Gamma L_2(32)}$ wird erzeugt von den beiden Permutationen

$$\begin{aligned} \alpha &:= (1\ 2\ 4\ 8\ 16)(3\ 6\ 12\ 24\ 17)(5\ 10\ 20\ 9\ 18)(7\ 14\ 28\ 25\ 19) \\ &\ (11\ 22\ 13\ 26\ 21)(15\ 30\ 29\ 27\ 23)(31)(32)(33) \qquad \text{und} \\ \beta &:= (1\ 18\ 30)(2\ 21\ 12)(3\ 10\ 28)(4\ 31\ 32)(5\ 24\ 14)(6\ 7\ 17) \\ &\ (8\ 25\ 27)(9\ 19\ 20)(11\ 15\ 13)(16\ 23\ 29)(22\ 33\ 26). \end{aligned}$$

Betten, Kerber, Kohnert, Laue und Wassermann [BKK$^+$95] verwenden Gitterbasenreduktionsalgorithmen zur Lösung des Gleichungssystems und können so einfache, nichttriviale 7–(33,8,10)– und 7–(33,8,16)–Designs mit Automorphismengruppe $\mathbf{P\Gamma L_2(32)}$ konstruieren. Das zu lösende Gleichungssystem (6.25) besteht aus $\rho(7) = 32$ Gleichungen und $\rho(8) = 97$ Variablen. (Demgegenüber bestand das ursprüngliche Gleichungssystem

(6.24) aus 4272048 Gleichungen und 13884156 Variablen.) Die dabei verwendete Gitterbasis $B = (b_1, \ldots, b_{\rho(k)+2})^T$ hat die Form

$$B := \left(\begin{array}{c|ccc|c|ccc}
1 & c_1 & \cdots & c_1 & 0 & 0 & \cdots & 0 \\ \hline
0 & 2c_1 & & & 0 & & & \\
 & & & 0 & & & & \\
\vdots & & \ddots & & \vdots & & c_0 {M_{t,k}^A}^T & \\
 & 0 & & & & & & \\
0 & & & 2c_1 & 0 & & & \\ \hline
0 & 0 & \cdots & 0 & 1 & c_0 & \cdots & c_0
\end{array}\right)$$

Jeder Gittervektor $y = (y_1, \ldots, y_{\rho(k)+\rho(t)+2})$ mit

$$|y_1| = 1,\ |y_2| = \cdots = |y_{\rho(k)+1}| = c_1,\ |y_{\rho(k)+2}| = \lambda,\ y_{\rho(k)+3} = \cdots = y_{\rho(k)+\rho(t)+2} = 0$$

liefert eine 0–1–Lösung des linearen Gleichungssystems und damit ein t–(k,v,λ)–Design.

Die Basis wird (β, δ)–blockreduziert mit $\beta = 40, \delta = 0.999999999, c_0 = 30, c_1 = 10$. Damit wird nach ca. 9 Minuten Rechenzeit auf einem PC 486 mit 66 MHz und 16 MB RAM jeweils eine Lösung für $\lambda = 10$ und $\lambda = 16$ gefunden. Der Versuch, mittels dieses Algorithmus ein einfaches, nichttriviales 8–(9,33,λ)–Design zu konstruieren, führt nicht zum Ziel. Der Algorithmus kann auch nicht zum Nachweis dafür verwendet werden, daß es kein solches Design mit Automorphismengruppe $\mathbf{P\Gamma L_2(32)}$ gibt. Dieser Nachweis gelingt jedoch mit einer modifizierten Version des Verfahrens zur Aufzählung von Gittervektoren mit l_∞–Norm 1.

ALGORITHMUS T–DESIGN
EINGABE: $t, k, \rho(t), \rho(k), B$

1. transformiere $b_2, \ldots, b_{\rho(k)+2}$ (z.B. durch Gauß–Elimination) so, daß
   $b_{i,j} = 0$ für $2 \leq i \leq \rho(k) - \rho(t) + 2, \quad \rho(k) + 3 \leq j \leq \rho(k) + \rho(t) + 2$
2. entferne $b_{\rho(k)-\rho(t)+3}, \ldots, b_{\rho(k)+2}$ aus der Basis
3. L^3–reduziere $b_2, \ldots, b_{\rho(k)-\rho(t)+2}$ mit $\delta = 0.99$.
4. verschiebe b_1 ans Ende der Basis und längenreduziere
5. definiere für $1 \leq i \leq \rho(k) - \rho(t) + 2,\ 1 \leq j \leq i$
   $b'_i := (b_{i,1}, \ldots, b_{i,\rho(k)+1})^T$
   $\mu'_{i,j} := <b'_i, \hat{b}'_j> / <\hat{b}'_j, \hat{b}'_j>$
   $c'_i := <\hat{b}'_i, \hat{b}'_i>$
6. rufe ENUM_∞ auf mit Eingaben $c'_i, b'_i, \hat{b}'_i, \mu'_{i,j}$

Die in Schritt 2 entfernten Vektoren können nicht zu einer Lösung des Gleichungssystems (6.25) beitragen, denn die Matrix $M^A_{t,k}$ besitzt vollen Rang $\rho(t)$, d.h. die letzten $\rho(t)$ Spalten von B sind linear unabhängig. Dadurch ist für jeden Gittervektor y, der von einem der entfernten Vektoren abhängt, mindestens einer der letzten $\rho(t)$ Koeffizienten von Null verschieden.

Im Algorithmus ENUM$_\infty$ werden folgende Modifikationen verwendet:
Wir beschränken uns auf den Fall $\widetilde{u}_{\rho(k)-\rho(t)+2} = 1$, denn wir sind nur an Gittervektoren y mit $y_1 = 1$ interessiert.
Die Abfrage „IF $\widetilde{c}_t < (R(\|.\|_\infty)c)^2$" wird ersetzt durch „IF $\widetilde{c}_t \leq \rho(k) + 1$".
Bei SCHNITT$_\infty$ beschränken wir uns auf $(\lambda_t, \ldots, \lambda_{\rho(k)-\rho(t)+2}) = (1, 0, \ldots, 0)$.
Falls ein Index $i \in \{2, \ldots, \rho(k) + 1\}$ existiert mit $\hat{b}'_{t,i} \neq 0,\ \hat{b}'_{1,i} = \cdots = \hat{b}'_{t-1,i} = 0,\ |w_{t,i}| \neq 1$, werden Teilbäume mit Wurzel $(\widetilde{u}_t, \ldots, \widetilde{u}_{\rho(k)-\rho(t)+2})$ abgeschnitten.
Anstelle eines updates von $(u_1, \ldots, u_{\rho(k)-\rho(t)+2})$ und c geben wir $(\widetilde{u}_1, \ldots, \widetilde{u}_{\rho(k)-\rho(t)+2})$ aus und setzen die Aufzählung mit dem nächsten $\widetilde{u}_1$ fort.
Jeder aufgezählte Vektor $v = \sum_{i=1}^{\rho(k)-\rho(t)+2} \widetilde{u}_i b'_i$ mit $\|v\|_\infty = 1$ liefert eine 0–1–Lösung des Gleichungssystems und damit ein t–(k,v,λ)–Design, wobei $\lambda = |\sum_{i=1}^{\rho(k)-\rho(t)+2} \widetilde{u}_i b_{i,\rho(k)+2}|$.

Für $t = 7,\ k = 8,\ v = 33$ und Automorphismengruppe A :=$\mathbf{P\Gamma L_2(32)}$ gilt $\rho(7) = 32$ und $\rho(8) = 97$, d.h. wir starten den Algorithmus T–DESIGN mit einer 99×131–Matrix B. In Schritt 6 zählen wir Vektoren in einem Gitter $L' \subset \mathbb{Z}^{98}$ vom Rang 67 auf. Der Algorithmus liefert auf einer HP–Workstation der Serie 735/99 circa 1000 7–(33,8,10)– und 7–(33,8,16)–Designs pro Minute.

Für $t = 8,\ k = 9,\ v = 33$ und A :=$\mathbf{P\Gamma L_2(32)}$ gilt $\rho(8) = 97$ und $\rho(9) = 248$, d.h. wir erhalten zu Beginn eine 250×347–Matrix B, die in Schritt 2 zu einer 153×250–Matrix verkleinert wird. Der Algorithmus benötigt auf einer HP–Workstation der Serie 735/99 insgesamt nur wenige Sekunden. Damit gelingt der Nachweis, daß keine einfachen, nichttrivialen 8–(33,9,λ)–Designs mit Automorphismengruppe $\mathbf{P\Gamma L_2(32)}$ existieren.

6.5 Faktorisierung von ganzen Zahlen

Viele Public–Key–Kryptosysteme basieren auf der Annahme, daß es nicht möglich ist, große natürliche Zahlen effizient in ihre Primfaktoren zu zerlegen. Der momentan schnellste Faktorisierungsalgorithmus ist das *Number Field Sieve* [LL93], das zur Faktorisierung einer Zahl N asymptotisch $O(e^{(1.92+o(1))(\ln N)^{1/3}(\ln\ln N)^{2/3}})$ arithmetische Operationen benötigt. Der Algorithmus konstruiert eine Kongruenz $x^2 \equiv y^2 \bmod N$ mit $x \not\equiv \pm y \bmod N$. Aus einer solchen Kongruenz erhalten wir unmittelbar zwei nichttriviale Faktoren von N.

Schnorr [Sch93] schlägt vor, eine solche Kongruenz mit Hilfe von Algorithmen zur Gitterbasenreduktion zu bestimmen:

EINGABE: N (eine zusammengesetzte Zahl mit mindestens 2 verschiedenen Primfaktoren)
$\alpha, c \in \mathbb{Q}$ mit $\alpha, c > 1$

1. Berechne die Liste $p_1, \ldots, p_t$ der ersten t Primzahlen, $p_t = (\ln N)^\alpha$.
2. Erzeuge mittels Gitterbasenreduktion eine Liste von $m \geq t+2$ Paaren $(u_i, v_i) \in \mathbb{N}^2$ mit den Eigenschaften
$$u_i = \prod_{j=1}^{t} p_j^{a_{i,j}} \quad \text{mit } a_{i,j} \in \mathbb{N} \tag{6.26}$$
$$|u_i - v_i N| \text{ ist über den Primzahlen } p_1, \ldots, p_t \text{ faktorisierbar.} \tag{6.27}$$
3. Faktorisiere $u_i - v_i N$ über den Primzahlen $p_1, \ldots, p_t$ und $p_0 = -1$.
Sei $u_i - v_i N = \prod_{j=0}^{t} p_j^{b_{i,j}}$, $\mathbf{b}_i = (b_{i,0}, \ldots, b_{i,t})$ und $\mathbf{a}_i = (a_{i,0}, \ldots, a_{i,t})$ mit $a_{i,0} = 0$.
4. Finde eine 0–1–Lösung $(c_1, \ldots, c_m) \neq (0, \ldots, 0)$ der Gleichung
$$\sum_{i=1}^{m} c_i(a_i + b_i) = 0(\mathrm{mod} 2)$$
5. $x := \prod_{j=0}^{t} p_j^{\sum_{i=1}^m c_i(a_{i,j}+b_{i,j})/2} (\mathrm{mod} N)$,
$y := \prod_{j=0}^{t} p_j^{\sum_{i=1}^m c_i b_{i,j}} (\mathrm{mod} N) = \prod_{j=0}^{t} p_j^{\sum_{i=1}^m c_i a_{i,j}} (\mathrm{mod} N)$.
(Die Konstruktion impliziert $x^2 = y^2 (\mathrm{mod} N)$.)
6. Falls $x \neq \pm y (\mathrm{mod} N)$, dann gebe $\mathrm{ggT}(x+y, N)$ aus und stoppe.
Sonst gehe nach 4 und erzeuge eine andere Lösung $(c_1, \ldots, c_m)$.

Die Paare (u_i, v_i) lassen sich aus Vektoren des Gitters $L_{\alpha,c} := L(b_1, \ldots, b_t)$ gewinnen, deren l_1–Norm–Abstand von $\mathbf{N} := (0, \ldots, 0, N^c \ln N)$ klein ist.

$$B = (b_1, \ldots, b_t)^T = \begin{pmatrix} \ln 2 & 0 & \cdots & 0 & N^c \ln 2 \\ 0 & \ln 3 & \cdots & 0 & N^c \ln 3 \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & \cdots & \ln p_t & N^c \ln p_t \end{pmatrix}.$$

Zu jedem Gittervektor $z = \sum_{i=1}^t e_i b_i$ definieren wir das Paar $(u(z), v(z))$ durch

$$u(z) := \prod_{e_i > 0} p_i^{e_i}, \qquad v(z) := \prod_{e_i < 0} p_i^{|e_i|}.$$

Schnorr zeigt, daß bei festem $\alpha, c > 1, \sigma > 0$ und $(\ln N)^\alpha = p_t < N$ für jeden Gittervektor z mit

$$\|z - \mathbf{N}\|_1 \leq (2c - 1) \ln N + 2\sigma \ln p_t \tag{6.28}$$

$|u(z) - v(z)N| \leq p_t^{1/\alpha + \sigma + o(1)}$ gilt.

Wir verfeinern dieses Resultat, um die Paare (u_i, v_i) effizienter bestimmen zu können. Insbesondere geben wir eine notwendige und eine hinreichende Bedingung dafür an, daß $|u - vN| \leq p_t^\sigma$ gilt. Hierzu betrachten wir die Eigenschaften der Logarithmusfunktion $f(x) := \ln x$. Für die Ableitungen gilt $f'(x) = \frac{1}{x} > 0$, $f''(x) = -\frac{1}{x^2} < 0$ und wir erhalten für jedes Paar $(a, b) \in \mathbb{R}^2$ mit $0 < a < b$

$$\begin{aligned} \ln a &< \ln b, \\ \ln b &< \ln a + \frac{b - a}{a}, \\ \ln b &> \ln a + \frac{b - a}{b}. \end{aligned}$$

Insgesamt folgt daraus

$$a(\ln b - \ln a) < b - a < b(\ln b - \ln a). \tag{6.29}$$

Wir setzen speziell $a := \min(u, vN)$, $b := \max(u, vN)$ und erhalten als notwendige Bedingung für $|u - vN| \leq p_t^\sigma$

$$\min(u, vN) |\ln u - \ln(vN)| < p_t^\sigma.$$

Dies ist äquivalent zu

$$\min(\ln u, \ln(vN)) + \ln(|N^c(\ln u - \ln(vN))|) < c \ln N + \sigma \ln p_t. \tag{6.30}$$

Als hinreichende Bedingung für $|u - vN| \leq p_t^\sigma$ erhalten wir analog

$$\max(\ln u, \ln(vN)) + \ln(|N^c(\ln u - \ln(vN))|) < c \ln N + \sigma \ln p_t. \tag{6.31}$$

Satz 6.1 *Seien $\alpha, c > 1, \sigma > 0$ fest und $(\ln N)^\alpha = p_t < N$. Dann gilt für jeden Vektor $z = \sum_{i=1}^t e_i b_i \in L_{\alpha,c}$:*

Eine notwendige Bedingung für $|u(z) - v(z)N| \leq p_t^\sigma$ ist

$$\frac{1}{2}\|z - \boldsymbol{N}\|_1 - \left(\frac{N^c + 1}{2N^c}\right)|(z - \boldsymbol{N})_{t+1}| + \ln(|(z - \boldsymbol{N})_{t+1}|) < (c - \frac{1}{2})\ln N + \sigma \ln p_t. \quad (6.32)$$

Eine hinreichende Bedingung für $|u(z) - v(z)N| \leq p_t^\sigma$ ist

$$\frac{1}{2}\|z - \boldsymbol{N}\|_1 - \left(\frac{N^c - 1}{2N^c}\right)|(z - \boldsymbol{N})_{t+1}| + \ln(|(z - \boldsymbol{N})_{t+1}|) < (c - \frac{1}{2})\ln N + \sigma \ln p_t. \quad (6.33)$$

Beweis von Satz 6.1:
Mit $u := u(z) = \prod_{e_i > 0} p_i^{e_i}$ und $v := v(z) = \prod_{e_i < 0} p_i^{|e_i|}$ gilt

$$\begin{aligned}
\min(\ln u, \ln(vN)) &= \frac{1}{2}(\ln u + \ln v + \ln N) - \frac{1}{2}|\ln u - \ln v - \ln N| \\
&= \frac{1}{2}\|z - \mathbf{N}\|_1 - \frac{N^c + 1}{2N^c}|(z - \mathbf{N})_{t+1}| + \frac{1}{2}\ln N, \\
\max(\ln u, \ln(vN)) &= \frac{1}{2}(\ln u + \ln v + \ln N) + \frac{1}{2}|\ln u - \ln v - \ln N| \\
&= \frac{1}{2}\|z - \mathbf{N}\|_1 - \frac{N^c - 1}{2N^c}|(z - \mathbf{N})_{t+1}| + \frac{1}{2}\ln N, \\
\ln(|N^c(\ln u - \ln(vN))|) &= \ln(|(z - \mathbf{N})_{t+1}|).
\end{aligned}$$

Durch Einsetzen dieser Äquivalenzen in (6.32) und (6.33) folgt die Behauptung. □

6.5.1 Praktische Experimente

Für praktische Experimente transformieren wir zunächst das „near vector" Problem (6.32) auf ein „short vector" Problem und approximieren anschließend die reellen durch ganzzahlige Vektoren. Die Approximation muß hinreichend gut sein, um die Rundungsfehler vernachlässigen zu können. Wir betrachten das durch die folgende Basis $\widetilde{B} := (\widetilde{b}_0, \widetilde{b}_1, \ldots, \widetilde{b}_t)^T$ aufgespannte Gitter:

$$\widetilde{B} := \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 & 10^{c_1}\ln N \\ 0 & 10^{c_2}\ln 2 & 0 & & 0 & 10^{c_1}\ln 2 \\ 0 & 0 & 10^{c_2}\ln 3 & & 0 & 10^{c_1}\ln 3 \\ \vdots & \vdots & & \ddots & & \vdots \\ 0 & 0 & 0 & \cdots & 10^{c_2}\ln p_t & 10^{c_1}\ln p_t \end{pmatrix}.$$

Für jeden Gittervektor $\widetilde{z} = (\widetilde{z}_0, \ldots, \widetilde{z}_{t+1}) = \sum_{i=0}^t e_i \widetilde{b}_i$ mit $|\widetilde{z}_0| = 1$ definieren wir

$$u(\widetilde{z}) := \prod_{\substack{i=1 \\ e_0 e_i < 0}}^t p_i^{|e_i|} \quad \text{und} \quad v(\widetilde{z}) := \prod_{\substack{i=1 \\ e_0 e_i > 0}}^t p_i^{|e_i|}.$$

Die notwendige Bedingung (6.32) für $|u - vN| < p_t^\sigma$ lautet mit den neuen Bezeichnungen

$$|\widetilde{z}_0| = 1, \qquad \frac{1}{2}10^{-c_2}\sum_{i=1}^{t}|\widetilde{z}_i| - 10^{-c_1}|\widetilde{z}_{t+1}| + \ln|\widetilde{z}_{t+1}| < c_1 \ln 10 + \sigma \ln p_t - \frac{1}{2}\ln N, \tag{6.34}$$

die hinreichende Bedingung (6.33) lautet

$$|\widetilde{z}_0| = 1, \qquad \frac{1}{2}10^{-c_2}\sum_{i=1}^{t}|\widetilde{z}_i| + 10^{-c_1}|\widetilde{z}_{t+1}| + \ln|\widetilde{z}_{t+1}| < c_1 \ln 10 + \sigma \ln p_t - \frac{1}{2}\ln N. \tag{6.35}$$

Bei den praktischen Implementationen verwenden wir stets ganzzahlige Gitter. Wir approximieren die reellen Vektoren $\widetilde{b}_0, \widetilde{b}_1, \ldots, \widetilde{b}_t$ durch ganzzahlige Vektoren und betrachten das durch die folgende Basis $B := (b_0, b_1, \ldots, b_t)^T$ aufgespannte Gitter.

$$B := \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 & \lceil 10^{c_1}\ln N\rfloor \\ 0 & \lceil 10^{c_2}\ln 2\rfloor & 0 & & 0 & \lceil 10^{c_1}\ln 2\rfloor \\ 0 & 0 & \lceil 10^{c_2}\ln 3\rfloor & & 0 & \lceil 10^{c_1}\ln 3\rfloor \\ \vdots & \vdots & & \ddots & & \vdots \\ 0 & 0 & 0 & \cdots & \lceil 10^{c_2}\ln p_t\rfloor & \lceil 10^{c_1}\ln p_t\rfloor \end{pmatrix}.$$

Algorithmus zur Bestimmung der Paare (u_i, v_i)
EINGABE: $N, t, c_1, c_2, \beta, \sigma, p, R$
1. Berechne die Basis $B = (b_0, b_1, \ldots, b_t)^T$
2. L^3–reduziere $b_0, \ldots, b_t$ mit $\delta = 0.99$
3. Permutiere die Vektoren $b_0, \ldots, b_t$ zufällig.
4. (β, δ)–blockreduziere $b_0, \ldots, b_t$ mit $\delta = 0.99$.

Wiederhole die Schritte 3 und 4 bis zu R mal.

Bei den Aufzählungen innerhalb der Blockreduktion wird die Gaußsche Volumenheuristik mit garantierter Erfolgswahrscheinlichkeit p verwendet. Nach jeder Längenreduktion innerhalb der L^3–Reduktionen testen wir, ob für den neuen (mit Rundungsfehlern behafteten) Vektor $z := (z_0, \ldots, z_{t+1})$ die Bedingungen (6.34) erfüllt sind. Falls (6.34) gilt, werden u, v und $u - vN$ berechnet und versucht, $u - vN$ über den ersten t Primzahlen zu faktorisieren. Der Algorithmus wird abgebrochen, sobald $t + 2$ Paare (u, v) gefunden wurden, für die dies möglich ist.

Der durch die Rundung entstehende Fehler auf den linken Seiten von (6.34) und (6.35) ist für jeden Gittervektor $z = \sum_{i=0}^{t} e_i b_i$ absolut beschränkt durch

$$\frac{1}{4}10^{-c_2}\sum_{i=1}^{t}|e_i| + \frac{1}{2}10^{-c_1}\sum_{i=0}^{t}|e_i| + \frac{\frac{1}{2}\sum_{i=0}^{t}|e_i|}{|z_{t+1}|}. \tag{6.36}$$

Nach Schritt 2 ist $|z_{t+1}|$ für hinreichend große c_1 und c_2 bei nahezu allen Basisvektoren von der Größenordnung 10^{c_2}. Damit ist dieser Fehler vernachlässigbar.

Schnorr verwendet für seine Versuche zur Faktorisierung der Zahl $N = 2131438662079$ die Parameter $t = 125,\ c_1 = 25,\ c_2 = 1,\ \beta = 32$ und $\sigma = 1$. Die Aufzählugen innerhalb der Blockreduktion sind nicht geschnitten. Außerdem wird anstelle von (6.34) getestet, ob

$$|z_0| = 1 \quad \text{und} \quad \|z\|_1 < 2c_1 \ln 10 + 2\sigma \ln p_t - \ln N \tag{6.37}$$

gilt. Dies ist bis auf Rundungsfehler äquivalent zum Test, ob Ungleichung (6.28) erfüllt ist. Das Auffinden eines einzelnen Paares (u, v) mit dem Schnorr–Algorithmus benötigt auf einer SUN–Sparcstation 1+ mehrere Stunden. Umgerechnet auf die Leistungsdaten einer HP–Workstation der Serie 735/125 bedeutet dies eine Laufzeit von mehreren Minuten.

Der neue Algorithmus liefert auf einer HP–Workstation der Serie 735/125 für die gleichen Parameter und Erfolgswahrscheinlichkeit $p = 0.05$ bereits einige Paare (u_i, v_i) pro Minute. Für $t = 125,\ c_1 = 15,\ c_2 = 4,\ \beta = 30,\ \sigma = 3.0$ und $p = 0.05$ findet er die benötigten $t+2 = 127$ Paare (u_i, v_i) in 2 Minuten. Die Schritte 3 und 4 des Algorithmus werden hierfür 3 mal durchlaufen. Durch die Erhöhung von c_2 auf 4 treten in den reduzierten Gitterbasen größere Zahlenwerte in der letzten Spalte auf. Diese Werte stimmen zumindest von der Größenordnung her mit den nicht gerundeten Werten überein.

Für $N = 250518388711599163$ liefert der Algorithmus auf einer HP–Workstation der Serie 735/125 mit den Parametern $t = 160,\ c_1 = 18,\ c_2 = 5,\ \beta = 100$ und $p = 0.001$ nach 27 Runden und Laufzeit 2.99 Stunden die zur Faktorisierung benötigten $t + 2 = 162$ Paare (u_i, v_i).

Eine mögliche Verbesserung des Algorithmus besteht darin, nach jeder Runde das effiziente Aufzählungsverfahren bezüglich der l_1–Norm zu verwenden. Hierzu transformieren wir die reduzierte Gitterbasis so, daß $b_{i,0} = 0$ für $i < t$ und $|b_{t,0}| = 1$ gilt. Bei der Aufzählung betrachten wir nur den Teilbaum mit Wurzel $\widetilde{u}_t = 1$. Anstelle von (6.34) zählen wir Vektoren auf, die (6.37) erfüllen. Erste Versuche zeigen, daß mit Hilfe dieses Verfahrens weitere Paare (u_i, v_i) gefunden werden. Allerdings sind die Laufzeiten hierfür noch deutlich höher als beim Verfahren ohne die abschließende Aufzählung. Erste Experimente zeigen, daß sich dieser Nachteil durch probabilistische Schnittmethoden ausgleichen läßt. Eine Optimierung dieser Schnittmethoden wurde aus Zeitgründen nicht mehr durchgeführt.

Literaturverzeichnis

[BKK+95] A. Betten, A. Kerber, A. Kohnert, R. Laue, and A. Wassermann. The discovery of simple 7-designs with automorphism group PΓL(2, 32). In *Applied Algebra, Algebraic Algorithms and Error–Correcting Codes, 11th International Symposium*, pages 131–145. Springer Lecture Notes in Computer Science, No. 948, 1995.

[Bli14] H.F. Blichfeldt. A new principle in the geometry of numbers, with some applications. *Transactions of the American Mathematical Society*, 15:227–235, 1914.

[Bli34] H.F. Blichfeldt. The minimum values of positive quadratic forms in six, seven and eight variables. *Mathematische Zeitschrift*, 39:1–15, 1934.

[Bri84] E.F. Brickell. Breaking iterated knapsacks. In *Proc. CRYPTO 84*, pp. 342–358, 1984.

[BS91] I.N. Bronstein and K.A. Semendjajew. *Taschenbuch der Mathematik*. Teubner Verlag, 1991. 25. Auflage.

[CJL+92] M. Coster, A. Joux, B. LaMacchia, A.M. Odlyzko, C.P. Schnorr, and J. Stern. Improved low–density subset sum algorithms. *Computational Complexity*, 2:111–128, 1992.

[Cop96a] D. Coppersmith. Finding a small root of a univariate modular equation. In *Proc. EUROCRYPT 96*, pp. 155–165, 1996.

[Cop96b] D. Coppersmith. Finding a small root of a bivariate integer equation; factoring with high bits known. In *Proc. EUROCRYPT 96*, pp. 178–189, 1996.

[CR88] B. Chor and R. L. Rivest. A knapsack type public-key cryptosystem based on arithmetic in finite fields. *IEEE Transactions on Information Theory*, 34(5):901–909, September 1988.

[CS97] D. Coppersmith and A. Shamir. Lattice attacks on NTRU. In *Proc. EUROCRYPT 97*, pp. 52–61, 1997.

[Dam89] I.B. Damgård. A design principle for hash functions. In *Proc. CRYPTO 89*, pp. 416–427. Springer Lecture Notes in Computer Science, No. 435, 1989.

[Dir50] G.L. Dirichlet. Über die Reduction der positiven quadratischen Formen mit drei unbestimmten ganzen Zahlen. *Journal reine angewandte Mathematik*, 40:228–232, 1850.

[EB81] P. van Emde Boas. Another NP–complete partition problem and the complexity of computing short vectors in a lattice. Technical report, Math. Department, 1981.

[Gau01] C.F. Gauß. Disquisitiones arithmeticae. Leipzig, 1801. Deutsche Übersetzung: Untersuchungen über die höhere Arithmetik; Springer, Berlin 1889. (reprint: Chelsea, New York, 1981.).

[Hel85] B. Helfrich. Algorithms to construct Minkowski reduced and Hermite reduced lattice bases. *Theoretical Computer Science*, 41:125–139, 1985.

[Her50] C. Hermite. Extraits de lettres de M.Ch. Hermite à M. Jacobi sur differents objets de la théorie des nombres, deuxième lettre. *Journal reine angewandte Mathematik*, 40:279–290, 1850.

[Hör94] H.H. Hörner. *Verbesserte Gitterbasenreduktion; getestet am Chor–Rivest Kryptosystem und an allgemeinen Rucksack–Problemen.* Diplomarbeit an der Universität Frankfurt, 1994.

[JG94] A. Joux and L. Granboulan. A practical attack against knapsack based hash functions. In *Proc. EUROCRYPT 94*, pp. 58–66, 1994.

[JS94] A. Joux and J. Stern. Lattice reduction: a toolbox for the cryptanalyst. Technical Report DGA/CELAR, ENS, 1994.

[Kai94] M. Kaib. *Gitterbasenreduktion für beliebige Normen.* Dissertation an der Universität Frankfurt, 1994.

[Kan87] R Kannan. Minkowski's convex body theorem and integer programming. *Mathematics of Operations Research*, 12:415–440, 1987.

[Kar84] N. Karmarkar. A new polynomial–time algorithm for linear programming. *Combinatorica*, 4(4):373–395, 1984.

[KL78] G.A. Kabatiansky and V.I. Levenshtein. Bounds for packings on a sphere and in space. *Problems of Information Transmission*, 14(1):1–17, 1978.

[KM76] E.S. Kramer and D.M. Mesner. t–designs on hypergraphs. *Discrete Math.*, 15:263–296, 1976.

[KZ72] A. Korkine and G. Zolotareff. Sur les formes quadratiques positives quaternaires. *Math. Annalen*, 5:581–583, 1872.

[KZ73] A. Korkine and G. Zolotareff. Sur les formes quadratiques. *Math. Annalen*, 6:366–389, 1873.

[KZ77] A. Korkine and G. Zolotareff. Sur les formes quadratiques positives. *Math. Annalen*, 11:242–292, 1877.

[Lag73] L. Lagrange. Recherches d'arithmetique. *Nouv. Mém. Acad. Berlin*, pp. 265–312, 1773. (= Œuvres 3, pp. 693–758).

[Len83] H.W. Jr. Lenstra. Integer programming with a fixed number of variables. *Mathematics of Operations Research*, 8(4):538–548, November 1983.

[LL93] A.K. Lenstra and H.W. Lenstra, Jr., editors. *The Development of the Number Field Sieve*, volume 1554 of *Lecture Notes in Mathematics*. Springer–Verlag, 1993.

[LLL82] A.K. Lenstra, H.W. Lenstra, Jr., and L. Lovász. Factoring polynomials with rational coefficients. *Math. Annalen*, 261:515–534, 1982.

[LLS90] J.C. Lagarias, H.W. Jr. Lenstra, and C.P. Schnorr. Korkin–Zolotarev bases and successive minima of a lattice and its reciprocal lattice. *Combinatorica*, 10:333–348, 1990.

[LO85] J.C. Lagarias and A.M. Odlyzko. Solving low–density subset sum problems. *Journal of the Association for Computing Machinery*, 32(1):229–246, 1985.

[LS92] L. Lovász and H. Scarf. The generalized basis reduction algorithm. *Mathematics of Operations Research*, 17(3):754–764, 1992.

[Mah38] K. Mahler. On Minkowski's theory of reduction of positive definite quadratic forms. *Quarterly J. Math.*, 9:259–262, 1938.

[MH78] R.C. Merkle and M.E. Hellman. Hiding information and signatures in trapdoor knapsacks. *IEEE Transactions on Information Theory*, IT–24(5):525–530, September 1978.

[Min91] H. Minkowski. Über die positiven quadratischen Formen und über kettenbruchähnliche Algorithmen. *Journal reine angewandte Mathematik*, 107:278–297, 1891.

[ML84] S. Magliveras and D.W. Leavitt. Designs from $\mathrm{P}\Gamma\mathrm{L}_2(32)$; computational group theory. *Computational Group Theory*, pp. 337–352, 1984.

[Ort94] G. Orton. A multiple–iterated trapdoor for dense compact knapsacks. In *Proc. EUROCRYPT 94*, pp. 112–130. Springer Lecture Notes in Computer Science, No. 950, 1994.

[Ort96] G. Orton. private Korrespondenz, 1996.

[PS82] C.H. Papadimitriou and K. Steiglitz. *Combinatorical Optimization: Algorithms and Complexity.* Prentice Hall, 1982.

[Rit96] H. Ritter. Breaking knapsack cryptosystems by l_∞–norm enumeration. In J. Pribyl, editor, *Proceedings of the 1st International Conference on the Theory and Applications of Cryptology, PRAGOCRYPT '96, Prague, Czech Republic, 30 September – 3 October, 1996, CTU Publishing House*, pp. 480–492, 1996.

[Sch87] C.P. Schnorr. A hierarchy of polynomial time lattice basis reduction algorithms. *Theoretical Computer Science*, 53:201–224, 1987.

[Sch93] C.P. Schnorr. Factoring integers and computing discrete logarithms via diophantine approximation. In *Advances in Computational Compexity.* DIMACS Series in Discrete Mathematics and Theoretical Science, AMS, 1993.

[Sch94] C.P. Schnorr. Block reduced lattice bases and successive minima. *Combinatorics, Probability and Computing*, 3:507–522, 1994.

[SE94] C.P. Schnorr and M. Euchner. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. *Mathematical Programming*, 66:181–199, 1994.

[SH95] C.P. Schnorr and H.H. Hörner. Attacking the Chor–Rivest cryptosystem by improved lattice reduction. In *Proc. EUROCRYPT 95*, pp. 1–12. Springer Lecture Notes in Computer Science, No. 921, 1995.

[Sha79] A. Shamir. On the cryptocomplexity of knapsack systems. In *Proc. 11th ACM Symp. on Theory of Computing*, pages 118–129, 1979.

[Sha82] A. Shamir. A polynomial time algorithm for breaking the basic Merkle–Hellman cryptosystem. In *Proc. 23rd IEEE Symp. on Foundations of Comp. Science*, pp. 145–152, 1982.

Symbolverzeichnis

$\|.\|$	Norm
$< a, b >$	Skalarprodukt der Vektoren a, b
$[a, b]$	abgeschlossenes Intervall
(a, b)	offenes Intervall
$[a, b), (a, b]$	halboffenes Intervall
$\lceil x \rceil = \min\{z \in \mathbb{Z} : z \geq x\}$	kleinste ganze Zahl $\geq x$
$\lfloor x \rfloor = \max\{z \in \mathbb{Z} : z \leq x\}$	größte ganze Zahl $\leq x$
$\lceil x \rfloor = \lceil x - 0.5 \rceil$	zu x nächste ganze Zahl
$\{x\} = x - \lceil x \rfloor$	Abstand von x zur nächsten ganzen Zahl
β	Blockweite, **13**
$\hat{b}_i$	Anteil des Vektors b_i orthogonal zu $\mathrm{span}(b_1, \ldots, b_{i-1})$, **8**
$c_i = \|\hat{b}_i\|_2^2$	Höhenquadrat des Vektors b_i, **19**
$c_j(\tilde{u}_j, \ldots, \tilde{u}_k) = \|\pi_j(\sum_{i=j}^{k} \tilde{u}_i b_i)\|_2^2$	Höhenquadrat des Vektors $\sum_{i=j}^{k} \tilde{u}_i b_i$, **25**
$\det(L)$	Determinante des Gitters L, **9**
F_i	Höhenfunktion, **11**
γ_β	Hermite–Konstanten, **15**
κ_β	κ–Konstanten, **19**
$L = L(b_1, \ldots, b_m)$	Gitter mit Basis $b_1, \ldots, b_m$, **7**
$\lambda_{i,\|.\|}(L)$	i–tes sukzessives Minimum des Gitters L, **7**
$\mu_{i,j}$	Gram–Schmidt–Koeffizienten, **8**
π_i	orthogonale Projektion auf $\mathrm{span}(b_1, \ldots, b_{i-1})^\perp$, **8**
$R(\|.\|), r(\|.\|)$	es gilt $r(\|.\|)\|x\| \leq \|x\|_2 \leq R(\|.\|)\|x\|$ für alle $x \in \mathbb{R}^n$, **22**
$S_n(M, r, \|.\|)$	n–dimensionale $\|.\|$–Kugel um M mit Radius r, **7**
$S(M, r)$	Kugel in euklidischer Norm um M mit Radius r, **7**
$\mathrm{sign}(x) = \begin{cases} -1, & x < 0 \\ 0, & x = 0 \\ 1, & x > 0 \end{cases}$	Vorzeichen der reellen Zahl x
$\mathrm{span}(L)$	lineare Hülle: der kleinste Vektorraum, der L enthält

Index

Lebenslauf

Name:	Harald Kilian Ritter	
Geburtstag und –ort:	25. August 1966, Aschaffenburg	
Eltern:	Walter und Hildegard Ritter (geb. Schnatz)	
Familienstand:	ledig	
Schulausbildung:	1972	Grundschule Großostheim
	1973–1976	Grundschule Karlstein–Dettingen
	1976–1985	Hanns–Seidel–Gymnasium Hösbach
	Juni 1985	Abitur
Grundwehrdienst:	Juli 1985–Sept. 1986	

Hochschulausbildung:

Okt. 1986 – Sept. 1992	Studium der Mathematik mit Nebenfach Informatik an der J.W. Goethe–Universität Frankfurt
Okt. 1988	Diplom–Vorprüfung Mathematik
Sept. 1992	Diplom–Hauptprüfung Mathematik, Thema der Diplomarbeit: *Zufallsbits basierend auf dem diskreten Logarithmus* Betreuer: Prof. Dr. C. P. Schnorr
seit Okt. 1992	Promotionsstudium Mathematik an der J.W. Goethe–Universität Frankfurt Betreuer der Dissertation: Prof. Dr. C. P. Schnorr

Berufserfahrung:

März 1991 – Nov. 1992	Wissenschaftliche Hilfskraft am Fachbereich Mathematik, Arbeitsgruppe Angewandte Mathematik/Theoretische Informatik
Dez. 1992 – März 1997	Wissenschaftlicher Mitarbeiter am Fachbereich Mathematik, Arbeitsgruppe Angewandte Mathematik/Theoretische Informatik
seit April 1997	Softwareentwickler bei *Gesellschaft für Finanzmarketing mbH*

Fremdsprachen:	Englisch, Französisch
Hobbies:	Chorgesang, Tischtennis, Radfahren, Fotografieren