UNIVERSITÄT DER BUNDESWEHR MÜNCHEN
Fakultät für Elektrotechnik und Informationstechnik

Das Shor-Verfahren als stochastischer Algorithmus

Felix Kugel

Vorsitzender des Promotionsausschusses: Prof. Dr. K. Pilzweger
1. Berichterstatter: Prof. Dr.Dr. S. Schäffler
2. Berichterstatter: Priv.-Doz. Dr. J.Schulze

Tag der Prüfung: 10. März 2006

Mit der Promotion erlangter akademischer Grad:
Doktor-Ingenieur
(Dr.-Ing.)

Neubiberg, der 19. März 2006

Meinen lieben Eltern

Elisabeth und Wolfgang Kugel

Inhaltsverzeichnis

1 Einleitung

1.1 Einführende Gedanken zur Faktorisierung heute

Public-Key-Kryptographie

Primzahlen wurde bis vor kurzer Zeit kein besonders hoher praktischer Stellenwert zugestanden. Doch seit der Entwicklung moderner Verschlüsselungsverfahren hat sich das entschieden geändert. Die meisten heutigen Verfahren zur Verschlüsselung und Entschlüsselung von Information bei Verwendung z.B. im

- Internet (Homebanking)
- Auto (SMS-Telematik, Zertifikate)

basieren auf der Schwierigkeit,

- große Primzahlen zu finden,
- eine große (öffentliche) Zahl in wenige aber große (private) Primfaktoren zu zerlegen.

RSA-Verfahren

Das bekannteste und erfolgreichste Verfahren in der Public-Key-Kryptographie ist das RSA-Verfahren. In diesem Verfahren muß u.a. eine große (öffentliche) natürliche Zahl n in zwei großen (private) Primzahlen zerlegt werden. Anwendung und Idee des Verfahrens werden in Kapitel 8.2 detailliert beschrieben.

Shor-Verfahren

Im Jahr 1994 demonstrierte der amerikanische Mathematiker Peter W. Shor, daß das Problem des Faktorisierens einer natürlichen Zahl n *effizient* auf einem Quanten-Rechner gelöst werden kann.

Was war mit *effizient* gemeint?

Teilt man eine natürliche Zahl n auf L Bits derart auf, daß

$$2^{L-1} \leq n \leq 2^L - 1$$

gilt, so benötigen klassische Rechner mindestens einen exponentiellen Aufwand von $O(L \cdot 2^{\frac{L}{2}})$ mathematischen Operationen, um für n einen Faktor zu ermitteln.

Effizient bedeutet im Zusammenhang mit dem Faktorisierungs-Problem, daß die Anzahl sogenannter „Gate-Operationen“ auf einem Quanten-Rechner ein einfaches Polynom darstellt. Für dieses Polynom definierte Shor die Zahl m in Abhängigkeit von n:

$$n^2 \leq 2^m < 2n^2$$

Mit Hilfe dieser Zahl m konnte Shor zeigen, daß die Anzahl der Gate-Operationen des sogenannten „Shor-Verfahrens“ auf dem Quanten-Rechner auf $O(m^3)$ beschränkt ist. Dieses Shor-Verfahren ist Teil eines Faktorisierungs-Algorithmus, der wegen der linearen Beziehung

$$L = \lfloor \frac{m+1}{2} \rfloor$$

damit insgesamt nur $O(L^3)$ Gate-Operationen benötigt. (In dieser Gleichung bezeichnet $\lfloor\ \rfloor$ die nächste nach unten gerundete natürliche Zahl. Die lineare Beziehung gilt nicht für die Zweierpotenzen $n = 2^k$.)

Bis heute gibt es noch keinen derart *effizienten* Quanten-Rechner. Die vorliegende Arbeit macht daher einen Ansatz, die stochastischen Eigenschaften der Quanten-Mathematik auszunutzen, um zu untersuchen, ob es für das Problem des Faktorisierens einer natürlichen Zahl n mit Hilfe des Shor-Verfahrens möglich ist, die Anzahl der mathematischen Operationen auf einem klassischen Rechner zu „polynomialisieren", was bedeutet, daß

- die Anzahl der mathematischen Operationen auf eine Größe der Form $O(m^p)$ reduziert werden kann. Die Potenz p soll hierbei eine kleine natürliche Zahl darstellen.
- für alle n ein maximaler Schwellwert p_{max} für den Potenzwert p, also ein $p \leq p_{max}$, existiert. Dies ist die wichtige Eigenschaft für die Darstellung der Anzahl als Polynom.

Diese Arbeit bevorzugt die Darstellung des Shor-Verfahrens mit Hilfe des Parameters m gegenüber L, da sich die Dimensionen dieses Verfahrens auf einem klassischen Rechner hierdurch einfacher darstellen lassen.

1.2 Eine persönliche Anmerkung

Für das Gelingen dieser Dissertation bedanke ich mich sehr sehr herzlich bei folgenden Personen:

- Herr Professor Dr. Dr. Stefan Schäffler betreute diese Dissertation. Er konnte mich durch seine stetige und unkomplizierte Hilfsbereitschaft immer auf das Neueste begeistern und motivieren. Besonders seine Bereitschaft, einen 40-jährigen Doktoranden zu betreuen, hat mich wirklich gefreut.
- Herr Privatdozent Dr. Jörg Schulze betreute diese Dissertation als Zweit-Korrektor. Er gab mir für die Fertigstellung dieser Arbeit einige sehr wichtige Impulse mit.
- Herr Gerhard Gries, Herr Holger Hingst und Herr Horst Schefts waren während der Dauer dieser Dissertation meine Vorgesetzten bei der Siemens VDO Automotive AG. Sie schufen die Rahmenbedingungen, die es mir ermöglichten, neben meiner beruflichen Tätigkeit diese Dissertation zu erstellen.
- Herr Norbert Burghardt und Herr Jürgen Grebe waren während der Dauer dieser Dissertation meine Projekt-Leiter in einem sehr zeitintensiven aber auch interessanten Automotive-Projekt. Sie schufen mir in der „heißen" Phase der Dissertation zeitliche Freiräume und hielten Projekt-Streß von mir fern.

Neubiberg, der 19. März 2006 Felix Kugel

1.3 Stichwortverzeichnis

Mathematische Koeffizienten

a	Basiswert der Modulo-Funktion $f(x) = a^x \pmod n$
c	Koeffizient innerhalb der vektoriellen Dimension des Argument-Hilbertraums: $0 \leq c \leq q$. „Relevante“ Koeffizienten werden über die Kettenbruchzerlegung $\lvert\frac{c}{q} - \frac{d}{r}\rvert \leq \frac{1}{2q}$ ermittelt.
f	Modulo-Funktion: $f(x) = a^x \pmod n$
L	Anzahl der Bits im Werte-Hilbertraum
m	Anzahl der Bits im Argument-Hilbertraum
n	Zahl, die faktorisiert wird: $n = n_1 \cdot n_2$
q	Vektorielle Dimension des Argument-Hilbertraums: $q = 2^m \geq n^2$
r	$r = \mathrm{ord}_n a =$ Ordnung von a modulo n. Für teilerfremde a und n ist r die kleinste natürliche Zahl, für die $a^r = 1 \pmod n$ gilt.
s	Reduzierte Anzahl im stochastischen Algorithmus: $s = s(m) = km^p$

Tabelle 1: Mathematische Koeffizienten

Quantenbit-Spezifische Ausdrucksformen

$\lvert k >$	Dirac-Notation für einen Basisvektor eines Multi-q-Bit (Physikalische Darstellung)
$\lvert x > \lvert y >$	Dirac-Notation für einen Basisvektor eines Multi-q-Bit in Tensorprodukt-Darstellung zweier Hilbert-Räume $\mathcal{H}_1 \otimes \mathcal{H}_2$: $\lvert x >$: Argument-Hilbertraum ($\mathcal{H}_1$) $\lvert y >$: Werte-Hilbertraum ($\mathcal{H}_2$)
$P(\lvert x > \lvert y >)$	Basis-Wahrscheinlichkeit: Wahrscheinlichkeit von Basisvektor $\lvert x > \lvert y >$
$P(\lvert x >)$	Argument-Wahrscheinlichkeit: Summe aller Basis-Wahrscheinlichkeiten für ein $\lvert x >$ im Argument-Hilbertraum: $P(\lvert x >) = \sum_y P(\lvert x > \lvert y >)$

Tabelle 2: Quantenbit-Spezifische Ausdrucksformen

Blockschaltbild des Shor-Verfahrens

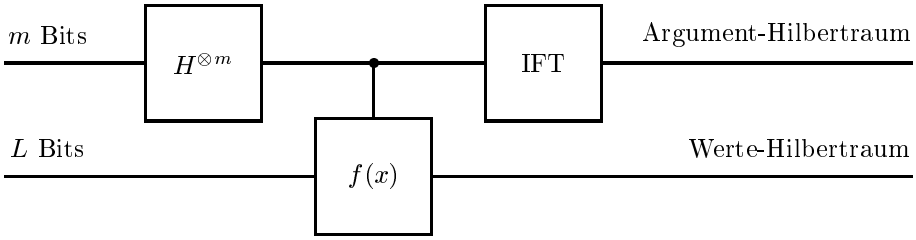

Abbildung 1: Blockschaltbild des Shor-Verfahrens

Begriffe des stochastischen Algorithmus

EGP	Einzelner gemessener Pfad: siehe Kapitel 4.5.4.
KMA	Kompletter mathematischer Algorithmus: siehe Kapitel 4.5.4.
	Relevante und nicht-relevante c-Werte: Relevante c-Werte erfüllen Gl.(78). Siehe Kapitel 4.2.4.
TSA	Teilsummen-Algorithmus: siehe Kapitel 4.5.4.

Tabelle 3: Begriffe des stochastischen Algorithmus

2 Verschiedene Faktorisierungs-Algorithmen

Primzahlen. Primzahlen sind Zahlen, die nur unechte Teiler haben, d.h. die nur durch 1 oder durch sich selbst teilbar sind. Z.B. ist 5 nur durch 1 und 5 teilbar und 13 ist nur durch 1 und 13 teilbar. Somit sind 5 und 13 Primzahlen. Die Zahl 1 selbst zählt nicht zu den Primzahlen, so daß die Primzahlfolge mit der Zahl 2 beginnt:

$$2, 3, 5, 7, 11, 13, 17, 19, 23, \ldots$$

Primfaktorzerlegung (Faktorisierung). Jede natürlich Zahl ist entweder selbst eine Primzahl oder läßt sich als Produkt von Primzahlen schreiben und somit in Primfaktoren zerlegen. Z.B. ist

$$120 = 4 \cdot 30 = 2^3 \cdot 3 \cdot 5.$$

Zur selben Zerlegung gelangt man, wenn man z.B. mit $120 = 10 \cdot 12$ beginnt. Mittels des *Euklidischen Algorithmus* läßt sich beweisen, daß es bis auf die Reihenfolge nur eine einzige Möglichkeit gibt, eine natürliche Zahl in Primzahlen aufzuspalten. Diese Formulierung des Satzes wäre unrichtig, würde man auch die 1 zu den Primzahlen rechnen.

Das Anliegen dieser Arbeit ist es, von einer natürlichen Zahl n einen echten Teiler (Faktor) zu ermitteln, so daß n als Produkt zweier Zahlen dargestellt werden kann (z.B. $120 = 10 \cdot 12$). Sukzessive Fortsetzung des Verfahrens mit den gefundenen Faktoren würde dann zur Primzahlzerlegung von n führen. Damit die Aufgabe nicht trivial wird, werden nur ungerade n untersucht.

2.1 Einfaches Faktorisieren

2.1.1 Einfaches Faktorisieren ohne Einschränkung der Divisoren

Den einfachsten Algorithmus zum Bestimmen eines Faktors von n erhält man, indem man diese Zahl einer sog. „Division mit Rest“ unterzieht. Hierbei ist der Divisor der Laufindex k, der bei 2 beginnt und solange um jeweils 1 erhöht wird, bis ein k erreicht ist, für das

$$n \pmod{k} = 0$$

Übersteigt k den Wert von $\lfloor\sqrt{n}\rfloor$, dann wurde kein Faktor gefunden, d.h. n ist Primzahl. Der folgende Algorithmus dividiert n solange, bis ein echter Faktor k gefunden wird.

1. [Einfacher Algorithmus] Falls $n = 1$, beende den Algorithmus.
2. [Initialisiere] Setze $k \leftarrow 2$.
3. [Division mit Rest] Ermittle den ganzzahligen Quotienten $q \leftarrow \lfloor \frac{n}{k} \rfloor$ und den Divisionsrest $r \leftarrow n \pmod{k}$.
4. [Faktor gefunden?] Falls $r = 0$, dann erkläre k als den gesuchten Faktor und beende den Algorithmus.

5. [Ende der Schleife?] Falls $k \geq q$, dann erkläre n als Primzahl, da kein Faktor gefunden wurde, und beende den Algorithmus.

6. [Erhöhe] Setze $k \leftarrow k + 1$ und gehe nach 3.

Ist n tatsächlich Primzahl, so führt keine der „Divisionen mit Rest“ zum gewünschten Ergebnis. In diesem Fall hat der Algorithmus die Maximalzahl von insgesamt $\lfloor\sqrt{n}\rfloor - 1$ „Divisionen mit Rest“ benötigt.
Besteht n allerdings aus dem Produkt zweier sog. „Zwillingsprimzahlen“, d.h. aus dem Produkt zweier Primzahlen, die nur um den Wert 2 differieren ($35 = 5 \cdot 7$ oder $143 = 11 \cdot 13$ oder $323 = 17 \cdot 19$ usw.), dann wird diese Maximalanzahl an „Divisionen mit Rest“ auch benötigt.
Diese Maximalanzahl läßt sich in der L-Bit-Arithmetik [siehe Gl.(21) in Kapitel 3.1] mit $O(2^{\frac{L}{2}})$ angeben.

Auf einem klassischen Rechner werden gemäß [KN] und [NI] (siehe auch Kapitel 3.5) pro „Divisionen mit Rest“ $O(L^2)$ Operationen benötigt. (Schnelle Algorithmen benötigen gemäß [KN] sogar nur $O(L^{1.58})$ Operationen. Auf diese wird hier nicht eingegangen.) Für den gesamten Algorithmus ergibt sich damit eine Anzahl von $O(L^2 2^{\frac{L}{2}})$ Operationen.

2.1.2 Einfaches Faktorisieren mit Quasi-Primzahlen

Ist eine Zahl ungerade, d.h. sie ist nicht durch 2 teilbar, dann ist sie selbstverständlich nicht durch 4 oder 6 oder 8 usw. teilbar.
Ist eine Zahl nicht durch 3 teilbar, dann ist sie selbstverständlich nicht durch 6 oder 9 oder 12 usw. teilbar.
Im oben genannten Algorithmus werden somit überflüssige „Divisionen mit Rest“ durchgeführt. Diese können vermieden werden, indem man als Divisoren die Menge aller Primzahlen hernimmt:

$$2, 3, 5, 7, 11, 13, 17, \ldots$$

Ist allerdings n sehr groß, so kostet es einigen Rechenaufwand, um alle Primzahlen $p \leq \lfloor\sqrt{n}\rfloor$ zu ermitteln.
Um diesen Aufwand einzusparen, kann man zur Ermittlung von Divisoren eine Zahlenfolge von „Quasi-Primzahlen“ entwickeln, die neben der 2 und 3 alle ungeraden Zahlen enthält, die nicht durch 3 teilbar sind:

$$2, 3, 5, 7, 11, 13, 17, 19, 23, 25, 29, 31, 35, 37, \ldots$$

Diese Zahlenfolge ergibt sich durch Addition nach dem dritten Term alternierden um 2 und um 4 und läßt sich auch folgendermaßen darstellen:

$$p_k = \begin{cases} 2 & \text{für } k = 1 \\ 3 & \text{für } k = 2 \\ 5 & \text{für } k = 3 \\ 7 & \text{für } k = 4 \\ p_{k-2} + 6 & \text{für } k \geq 5 \end{cases}$$

Der Faktorisierungs-Algorithmus lautet nun unter Zuhilfenahme von p_k folgendermaßen:

1. [Einfacher Algorithmus] Falls $n = 1$, beende den Algorithmus.
2. [Initialisiere] Setze $k \leftarrow 1$.
3. [Dividiere] Ermittle den ganzzahligen Quotienten $q \leftarrow \lfloor n/p_k \rfloor$ und den Divisionsrest $r \leftarrow n \pmod{p_k}$.
4. [Faktor gefunden?] Falls $r = 0$, dann erkläre p_k als den gesuchten Faktor und beende den Algorithmus.
5. [Ende der Schleife?] Falls $p_k \geq q$, dann erkläre n als Primzahl, da kein Faktor gefunden wurde, und beende den Algorithmus.
6. [Erhöhe] Setze $k \leftarrow k + 1$ und gehe nach 3.

Für $k \geq 5$ liegen in jedem Intervall von 6 natürlichen Zahlen immer nur zwei Zahlen der Zahlenfolge p_k. Hierdurch ergibt sich eine Reduktionen der Anzahl der „Divisionen mit Rest“ gegenüber dem oben beschriebenen Algorithmus um den Faktor 3.
Gemäß [KN] kann die Anzahl der Divisoren noch einmal um 20% verringert werden, wenn aus der Zahlenfolge alle Elemente vom Typ $30v \pm 5$ gestrichen werden. Weitere 14% werden eingespart durch Weglassen aller Vielfachen der Zahl 7.
Insgesamt verringert sich mit Hilfe von p_k die gesamte Anzahl der Divisionen um einen konstanten Faktor, der aber nicht von L abhängt, so daß sich die Gesamtzahl der Operationen weiterhin im Bereich von $O(L^2 2^{\frac{L}{2}})$ bewegt.

2.1.3 Einfaches Faktorisieren mit echten Primzahlen

Dieses Kapitel untersucht einen Faktorisierungs-Algorithmus mit reine Primzahlen p als Divisoren. Gemäß [KN] existieren zu einer natürlichen Zahl m ungefähr

$$\frac{m}{\ln(m)}$$

Primzahlen p mit $p < m$. Dieser Zahlenwert nähert sich allerdings nur asymptotisch dem wahren Zahlenwert, der im Allgemeinen höher ist, wie man anhand der folgenden Tabelle 4 erkennen kann:

m	Anzahl$(p) < m$	$m/\ln(m)$
10^3	168	145
10^6	78498	72382
10^9	50847534	48254942

Tabelle 4: Anzahl der Primzahlen kleiner m

Teilt man n auf L Bits auf [siehe Gl.(21) in Kapitel 3.1] und geht davon aus, daß für die Ermittlung der Divisoren nur Primzahlen p im Bereich von

$$2 \leq p \leq \lfloor \sqrt{n} \rfloor$$

benötigt werden, dann ergibt sich als Obergrenze für die Divisoren der Wert

$$m = \lfloor \sqrt{n} \rfloor \ < \ 2^{\frac{L}{2}}.$$

Setzt man nun $2^{\frac{L}{2}}$ in $\frac{m}{\ln(m)}$ ein, dann erhält man

$$\frac{2^{\frac{L}{2}}}{\ln(2^{\frac{L}{2}})} = \frac{2^{\frac{L}{2}}}{\frac{L}{2}\ln(2)} = \frac{2 \cdot 2^{\frac{L}{2}}}{L \cdot \ln(2)},$$

also ungefähr $O(\frac{1}{L} 2^{\frac{L}{2}})$ Divisoren für den Faktorisierungs-Algorithmus.
Multipliziert man diese Anzahl mit der Anzahl $O(L^2)$ der Operationen für jede „Division mit Rest“ (siehe Kapitel 3.5), so ergibt sich eine Gesamtanzahl von höchstens

$$O(L\ 2^{\frac{L}{2}})$$

Operationen. Dies bedeutet ein Einsparen um den Faktor L gegenüber den beiden bisher genannten Algorithmen. Allerdings ist hierbei noch nicht der Aufwand für die Ermittlung aller Primzahlen, die als Divisoren für den Faktorisierungs-Algorithmus benötigt werden, berücksichtigt. Wie man diese Primzahlen schnell und effizient ermitteln kann, zeigt folgendes Verfahren, das sogenannte *Sieb des Eratosthenes*.

Sieb des Eratosthenes: Der griechische Mathematiker *Erathostenes* (etwa 275 - 194 v. Chr.) erfand ein Verfahren, wie man sämtliche Primzahlen innerhalb eines Intervalls von 2 bis m findet. Dieses Verfahren wird im folgenden beschrieben:

- Man startet mit der kleinsten in diesem Intervall befindlichen Zahl, der 2, und streicht alle Vielfachen dieser Zahl (4, 6, 8, ...) aus dem Intervall heraus.
- Von den verbliebenen ungeraden Zahlen geht man zur nächstkleinsten Zahl, hier die 3, über und streicht dementsprechend alle Vielfachen dieser Zahl heraus (9, 15, 21, ..., die Zahlen 6, 12, 18 ... sind schon in der Streichsequenz der Zahl 2 herausgefallen).
- Dieses Verfahren läßt sich nun beliebig fortfahren: die nächstkleinsten Zahlen sind 5 und 7. Am Ende bleiben nur die Primzahlen des Intervals übrig.

Für die Zahlen bis $m = 100$ werden nur 4 Streichsequenzen benötigt. Die letzte Streichsequenz beinhaltet alle durch 7 teilbaren Zahlen. Das liegt daran, daß zwar $7 \cdot 7 = 49 < m$ ist, aber die nächste Primzahl 11 mit $11 \cdot 11 = 121 > m$ schon zu groß für weitere Streichungen ist.

Um dieses Verfahren zu konkretisieren, reicht es aus, zunächst einmal nur die ungeraden Zahlen im Intervall von 3 bis m zu betrachten. Daraus ergibt sich eine Menge P mit

$$P = \{x_j = 2j + 1 : 1 \leq j \leq \tfrac{m-1}{2}\} = \{3, 5, 7, \ldots\}.$$

Mit P läßt sich das Verfahren folgendermaßen eingrenzen:

- Beginnend mit der niedrigsten ungeraden Zahl x_j, also mit $p = x_1 = 3$, streicht man aus P die Zahlen p^2, $p(p+2)$, $p(p+4)$, $p(p+2v)$.
- Aus der dezimierten Restmenge von P sucht man nach x_1 die nächstkleine Zahl, in diesem Fall $x_2 = 5$, setzt sie in p ein und streicht hierzu die Zahlen p^2, $p(p+2)$, $p(p+4)$, $p(p+2v)$.
- Das Verfahren wiederholt man solange, bis eine Primzahl $p = x_j$ erreicht wird, für die $p^2 > m$ gilt.
- Am Ende muß man die Zahl 2 zur Restmenge hinzuaddieren, so daß nur Primzahlen übrigbleiben.

Der folgende Algorithmus in Anlehnung an [KN] hat den besonderen Vorteil, daß er auf multiplikative Operationen verzichtet. Ausgehend von der falschen Behauptung, daß alle ungeraden Zahlen $x_j = 2j + 1$ in P Primzahlen sind, definiert er zu jedem x_j einen korrespondieren boolschen Wert b_j, der mit $b_j = \mathrm{T}$ vorinitialisiert wird, was bedeutet, daß die Zahl x_j Primzahl ist.

j	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
x_j	3	5	7	9	11	13	15	17	19	21	23	25	27	29	31
b_j	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T

Tabelle 5: Vor Beginn des Siebens: Alle x-Werte sind Quasi-Primzahlen

Erst im Verlauf des Algorithmus werden für alle diejenigen ungeraden Zahlen x_j, die tatsächlich keine Primzahlen sind, deren boolsche Werte $b_j = \mathrm{F}$ gesetzt. Diese Zahlen sind die gestrichenen oder „ausgesiebten“ Zahlen.
Die erste ungerade Zahl, die nicht Primzahl ist, ist $x_4 = 9$. Folglich wird im weiteren Verlauf $b_4 = \mathrm{F}$ gesetzt werden. Die Zahl 9 wird damit „ausgesiebt“.

j	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
x_j	3	5	7	9	11	13	15	17	19	21	23	25	27	29	31
b_j	T	T	T	**F**	T	T	**F**	T	T	**F**	T	T	**F**	T	T

Tabelle 6: Erste Streichsequenz bei Vielfachen von $p = x_1 = 3$

In diesem Algorithmus werden den Lauf-Parametern j, k, p und q folgende Bedeutung zuteil:

$$p = x_j = 2j + 1$$

$$p^2 = x_j^2 = (2j+1)^2 = 4j^2 + 4j + 1 = 2(2j^2 + 2j) + 1 = 2q + 1 = x_q$$

$$\Longrightarrow p^2 = x_q \text{ mit } q = 2j^2 + 2j$$

$$p^2 + 2vp = x_j(x_j + 2v) = (2q+1) + 2vp = 2(q + vp) + 1 = x_k$$

$$\Longrightarrow p(p+2v) = x_k \text{ mit } k = q + vp$$

Beim Inkrementieren von $j \leftarrow j+1$ ergeben sich für p und q folgende Zuweisungsformeln:

$$p \leftarrow 2(j+1)+1 = 2j+3$$

$$\Longrightarrow p \leftarrow p+2$$

$$q \leftarrow 2(j+1)^2 + 2(j+1) = 2j^2 + 6j + 4 = (2j^2+2j) + 2(2j+3) - 2$$

$$\Longrightarrow q \leftarrow q + 2p - 2$$

Mit diesen additiven Zuweisungen, mit deren Hilfe Multiplikationen umschrieben werden, ergibt sich nun folgender Algorithmus zum Ermitteln aller Primzahlen innerhalb des Intervalls von 2 bis m:

1. [Initialisiere] Sei $m = 2M$ oder $m = 2M+1$. Initialisiere $b_k \leftarrow True$ für alle $1 \leq k \leq M$. Setze auch $j \leftarrow 1$, $p \leftarrow 3$ und $q \leftarrow 4$. (p und q ergeben sich aus j, wie oben beschrieben.) Schreibe die Zahl 2 in den Ergebnis-Vektor aller Primzahlen.

2. [Primzahl?] Ist $b_j = True$, dann setze $k \leftarrow q$. Ist $b_j = False$, dann gehe nach 4.

3. [Streiche Vielfache] Falls $k \leq M$, dann setze $b_k \leftarrow False$, $k \leftarrow k+p$ und wiederhole diesen Schritt 3.

4. [Inkremetiere] Setze $j \leftarrow j+1$, $p \leftarrow p+2$, $q \leftarrow q+2p-2$. Falls $q \leq M$, dann gehe nach Schritt 2.

5. [Ergebnis] Schreibe alle diejenigen x_k, für die (immer noch) $b_k = True$ gilt, in den Ergebnis-Vektor aller Primzahlen und beende den Algorithmus.

Beispiel 1:
Für $n = 50$ ergibt sich zunächst eine Menge $P = \{3, 5, 7, \ldots, 49\}$. Aus dieser Menge werden die Elemente

$$3\cdot 3,\ 3\cdot 5,\ 3\cdot 7,\ \ldots,\ 3\cdot 15,\ 5\cdot 5,\ 5\cdot 7,\ 5\cdot 9,\ 7\cdot 7$$

ausgesiebt, wobei die Zahl $x_{22} = 45 = 3\cdot 15 = 5\cdot 9$ zweimal, also einmal zuviel, ausgesiebt wird.

Beispiel 2:
Für $m = 100$ werden 49 Zahlen in die Menge P geschrieben. Aus dieser Menge wird 28 mal gesiebt, wobei die drei Zahlen 45, 63 und 75 zweimal, also einmal zu oft gesiebt werden. Damit ergeben sich mit der Zahl 2 zusammen insgesamt $49 - 28 + 3 + 1 = 25$ Primzahlen.

Beispiel 2:
Für $m = 200$ werden 99 Zahlen in die Menge P geschrieben. Aus dieser Menge wird 67 mal gesiebt. Hier wird insgesamt 13 mal zu oft gesiebt, wobei die

Zahlen 105, 165 und 195 sogar dreimal, also zweimal zu oft gesiebt werden. Damit ergeben sich mit der Zahl 2 zusammen insgesamt $99 - 67 + 13 + 1 = 46$ Primzahlen.

Die folgende Tabelle 7 gibt eine Übersicht über das Siebverfahren für große Werte von m.

m	Anzahl in Menge P	Anzahl aller gesiebten Zahlen	Anzahl zu oft gesiebter Zahlen	Anzahl der Primzahlen
10^2	49	28	3	25
10^3	499	457	125	168
10^4	4999	5995	2224	1229
10^5	49999	71556	31148	9592
10^6	499999	811068	389566	78498
10^7	4999999	8925144	4589723	664579

Tabelle 7: Siebkriterien für das Sieb des Eratosthenes

Wie man aus der Tabelle 7 erkennen kann, steigt mit größer werdendem m das Verhältnis

$$\frac{\text{Anzahl aller gesiebten Zahlen}}{\text{Anzahl in Menge } P}.$$

Interessant ist zu beobachten, daß für $m = 10^4$ die Anzahl der gesiebten Elemente größer ist als die Anzahl der Elemente in P. Für $m = 10^9$ übersteigt die Anzahl der gesiebten Zahlen mit $1.024 \cdot 10^9$ sogar die Zahl m.

Der oben beschriebene Algorithmus besteht im wesentlichen aus Additionen, wie bereits weiter oben geschildert wurde. Jede Addition führt zu einer Zahl, die „ausgesiebt“ wird. Aus diesem Grund gibt die Anzahl der gesiebten Elemente Auskunft über die Computational Resources des Siebverfahrens, sie stellt im wesentlichen die Gesamtzahl der benötigten Additionen. Da $m = \sqrt{n}$ ist, liegt bis zum Wert $n = 10^{18}$ die Obergrenze der benötigten Additionen in der L-Bit-Arithmetik bei $O(2^{\frac{L}{2}})$, d.h. die Anzahl der benötigten Operationen bei $O(L \cdot 2^{\frac{L}{2}})$.

Fazit: Addiert man die Berechnung der Primzahlen mit Hilfe des Siebs des Erathostenes zum Aufwand des Faktorisierungs-Algorithmus, deren Divisoren nur aus Primzahlen besteht, hinzu, dann wird der operative Aufwand im Bereich von $O(L \cdot 2^{\frac{L}{2}})$ nur für sehr große Zahlen (größer $n = 10^{18}$) überschritten.
Hinzu kommt allerdings noch als reiner Speicher eine Menge P der Mächtigkeit $\#P \leq \frac{m}{2}$, der die boolschen Werte b_k enthält und der linear mit m, also $O(2^{\frac{L}{2}})$, ansteigt.
Sieht man allerdings die Berechnung einer Auswahl an Primzahlen als einen einmaligen und unabhängigen Schritt an, dessen Ergebnismenge beliebig oft in einem Faktorisierungs-Algorithmus eingesetzt werden kann, dann ist dieser Faktorisierungs-Algorithmus im operativen Bereich den beiden anderen genannten Verfahren um den Faktor L überlegen.

2.2 Faktorisieren mit dem Verfahren von Gary Lee Miller

2.2.1 Der Algorithmus

In diesem Kapitel wird das Faktorisierungsverfahren erläutert, welches P. Shor für seinen Quantenalgorithmus verwendet hat. Hierbei wird neben der Zahl $n \in \mathbb{N}$, die faktorisiert werden soll, eine weitere natürliche Zahl $a \in \mathbb{N}$ benötigt, die folgende Eigenschaft haben soll:

$$a \in \mathbb{Z}_n^* \tag{1}$$

Die Menge $\mathbb{Z}_n^*$ beinhaltet alle positiven natürlichen Zahlen, die kleiner als n sind und zusätzlich teilerfremd zu n. (ggT: größter gemeinsamer Teiler.)

$$\mathbb{Z}_n^* = \{b \in \mathbb{N} : 1 \leq b \leq n-1 \ \wedge \ \mathrm{ggT}(b,n) = 1\} \tag{2}$$

Die Anzahl der Elemente einer Menge heißt *Mächtigkeit* einer Menge. Die Mächtigkeit von $\mathbb{Z}_n^*$ wird durch die *Euler*-Funktion

$$\varphi(n) = \#[\mathbb{Z}_n^*] \tag{3}$$

dargestellt. Mit der Hilfszahl a wird die Funktion

$$f(x) = a^x \pmod{n} \tag{4}$$

gebildet, für die gezeigt werden kann, daß ihre Funktionswerte Elemente von $\mathbb{Z}_n^*$ sind. Da demnach die Bildmenge $f(\mathbb{N})$ endlich ist, ist $f(x)$ periodisch, d.h. es existiert eine kleinste positive und natürliche Zahl r, die die Gleichung

$$f(x+r) = f(x)$$

erfüllt. Wegen

$$f(x+r) = f(x) \cdot f(r)$$

gilt

$$f(r) = a^r = 1 \pmod{n} \tag{5}$$

Die Zahl r bezeichnet man als „Ordnung von a modulo n“ und in mathematischer Darstellung schreibt sie sich:

$$r = \mathrm{ord}_n a \tag{6}$$

Der erste Schritt des Verfahrens von G. L. Miller beinhaltet somit das Ermitteln der Ordnung r für gegebene Werte von a und n. Für die weiteren Schritte werden zwei Bedingungen formuliert, die erfüllt sein müssen, damit das Verfahren zu einem erfolgreichen Abschluß geführt wird.

Die erste Bedingung verlangt, daß r gerade ist, damit diese Zahl halbiert werden kann. Ist diese Bedingung erfüllt, dann wird mit Hilfe von $f(r)$ folgende Umformung durchgeführt:

$$f(r) - 1 = a^r - 1 = (a^{\frac{r}{2}} - 1)(a^{\frac{r}{2}} + 1) = 0 \pmod{n} \tag{7}$$

Gl.(7) kann nun unter Zuhilfenahme einer natürlichen Zahl $c = c_1 \cdot c_2$ und mit Hilfe der noch unbekannten Faktorisierung $n = n_1 \cdot n_2$ durch

$$(a^{\frac{r}{2}} - 1)(a^{\frac{r}{2}} + 1) = cn = c_1 n_1 \cdot c_2 n_2 \tag{8}$$

dargestellt werden. Um eine Lösung für das Faktorisierungsproblem zu finden, ist für die Formulierung einer zweiten Bedingung notwendig, daß die Ungleichungen

$$n_1 < n \ \wedge \ n_2 < n \tag{9}$$

erfüllt sind, so daß

$$a^{\frac{r}{2}} - 1 = c_1 n_1 \ \wedge \ a^{\frac{r}{2}} + 1 = c_2 n_2 \tag{10}$$

gilt. Nur in diesem Fall können Faktoren von n durch

$$\mathrm{ggT}(n, a^{\frac{r}{2}} - 1) \ \wedge \ \mathrm{ggT}(n, a^{\frac{r}{2}} + 1)$$

gefunden werden. Diese Faktoren haben die Eigenschaft:

$$n_1 \mid \mathrm{ggT}(n, a^{\frac{r}{2}} - 1) \mid n \ \wedge \ n_2 \mid \mathrm{ggT}(n, a^{\frac{r}{2}} + 1) \mid n$$

Hierbei ist „|“ das Teilungsymbol. „$A \mid B$“ bedeutet „A teilt B“.

Wird die Bedingung in Gl.(9) nicht erfüllt, d.h.

$$n_1 = n \ \vee \ n_2 = n,$$

dann wäre

$$\begin{array}{lll} & a^{\frac{r}{2}} - 1 = c_1 n & \vee \quad a^{\frac{r}{2}} + 1 = c_2 n \\ \Longleftrightarrow & a^{\frac{r}{2}} = 1 + c_1 n & \vee \quad a^{\frac{r}{2}} = -1 + c_2 n \\ \Longleftrightarrow & a^{\frac{r}{2}} = 1 \ (\mathrm{mod}\ n) & \vee \quad a^{\frac{r}{2}} = -1 \ (\mathrm{mod}\ n) \end{array}$$

$$\Longleftrightarrow f(\frac{r}{2}) = 1 \ (\mathrm{mod}\ n) \ \vee \ f(\frac{r}{2}) = -1 \ (\mathrm{mod}\ n) \tag{11}$$

und es ergäbe sich damit keine weitere Möglichkeit, n weiter zu zerlegen.

Von den beiden Alternativen in Gl.(11) kann aber die linke nicht zutreffen, da gemäß Gl.(5) die Ordnung von a modulo n durch r definiert ist. Folglich wird als zweite Bedingung die rechte Seite von Gl.(11) hergenommen:

$$f(\frac{r}{2}) \neq -1 \ (\mathrm{mod}\ n) \tag{12}$$

Faßt man nun beide Bedingungen zusammen zu

$$\{r \text{ ist gerade}\} \ \wedge \ \{f(\frac{r}{2}) \neq -1 \ (\mathrm{mod}\ n)\} \tag{13}$$

dann gilt gemäß [NI] für jede natürliche Zahl n, die sich in ihre Primfaktoren p_k

$$n = p_1^{\alpha_1} \cdot p_2^{\alpha_2} \cdot \ldots \cdot p_K^{\alpha_K} \tag{14}$$

zerlegen läßt, folgende Aussage:

Die beiden Bedingungen sind für $K \geq 2$ erfüllt mit der Wahrscheinlichkeit (Siehe [1])

$$P[\{r \text{ ist gerade}\} \wedge \{f(\frac{r}{2}) \neq -1 \pmod{n}\}] \geq 1 - \frac{1}{2^{K-1}} \qquad (15)$$

D.h. für $K \geq 2$ ist die Wahrscheinlichkeit immer größer gleich 50%.

Für die praktische Anwendung des Verfahrens von G. L. Miller ist es hilfreich, bei der Berechnung des ggT auf den Satz von *Euclid* zurückzugreifen:

$$\text{ggT}(x, n) = \text{ggT}[x \pmod{n}, n] \qquad (16)$$

Angewendet auf das hier beschriebene Verfahren bedeutet dies:

$$\text{ggT}(a^{\frac{r}{2}} \pm 1, n) = \text{ggT}[(a^{\frac{r}{2}} \pm 1) (\text{mod } n), n] = \text{ggT}[f(\frac{r}{2}) \pm 1, n] \qquad (17)$$

2.2.2 Beispiele

Zum Verfahren von G. L. Miller nun ein paar Beispiele.

Beispiel 1: Sei $n = 15$ und $a = 7$. Dann gilt:

$a^4 = 1 \pmod{n} \Rightarrow r = 4$
r gerade $\wedge$ $f(\frac{r}{2}) = 4 \pmod{n}$
$\text{ggT}[f(\frac{r}{2}) - 1, n] = \text{ggT}(3, 15) = 3$
$\text{ggT}[f(\frac{r}{2}) + 1, n] = \text{ggT}(5, 15) = 5$
$\Rightarrow 15 = 3 \cdot 5$

Beispiel 2: Sei $n = 225$ und $a = 2$. Dann gilt:

$a^{60} = 1 \pmod{n} \Rightarrow r = 60$
r gerade $\wedge$ $f(\frac{r}{2}) = 199 \pmod{n}$
$\text{ggT}[f(\frac{r}{2}) - 1, n] = \text{ggT}(198, 225) = 9$
$\text{ggT}[f(\frac{r}{2}) + 1, n] = \text{ggT}(200, 225) = 25$
$\Rightarrow 225 = 9 \cdot 25$

Beispiel 3: Sei $n = 697$ und $a = 23$. Dann gilt:

$a^{80} = 1 \pmod{n} \Rightarrow r = 80$
r gerade $\wedge$ $f(\frac{r}{2}) = 288 \pmod{n}$
$\text{ggT}[f(\frac{r}{2}) - 1, n] = \text{ggT}(287, 697) = 41$
$\text{ggT}[f(\frac{r}{2}) + 1, n] = \text{ggT}(289, 697) = 17$
$\Rightarrow 697 = 41 \cdot 17$

[1] In [NI] und in [SH](1) wird fälschlicherweise $(1 - \frac{1}{2^K})$ gesetzt. Der Exponent K ist falsch und muß durch $(K - 1)$ ersetzt werden. Mehr hierzu in Kapitel 8.1.

Beispiel 4: Sei $n = 35$ und $a = 11$. Dann gilt:

$$a^3 = 1 \pmod{n} \;\Rightarrow\; r = 3$$

Da mit ungeradem r die erste Bedingung von Gl.(15) nicht erfüllt ist, kann das Verfahren hier nicht weiter angewendet werden.

Beispiel 5: Sei $n = 85$ und $a = 13$. Dann gilt:

$$a^4 = 1 \pmod{n} \;\Rightarrow\; r = 4$$
$$r \text{ gerade } \wedge\; f(\tfrac{r}{2}) = -1 \pmod{n}$$

Da mit $f(\frac{r}{2}) = -1 \pmod{n}$ die zweite Bedingung von Gl.(15) nicht erfüllt ist, kann das Verfahren hier nicht weiter angewendet werden.

Beispiel 6: Sei $n = 21$ und $a \in \mathbb{Z}_{21}^*$. $\mathbb{Z}_{21}^*$ ist gemäß Gl.(3) definiert. Dann gilt:

- Für $a = 2, 10, 11, 19$ ergibt $r = 6$. Diese Ordnung führt zu einer erfolgreichen Zerlegung $21 = 3 \cdot 7$.
- Für $a = 8, 13$ ergibt $r = 2$. Diese Ordnung führt zu einer erfolgreichen Zerlegung $21 = 3 \cdot 7$.
- Für $a = 1, 4, 16$ wird das Verfahren aufgrund einer ungeraden Ordnung r erfolglos abgebrochen.
- Für $a = 5, 17, 20$ wird das Verfahren wegen $a^{\frac{r}{2}} = -1 \pmod{n}$ erfolglos abgebrochen.
- Zusammenfassung: Die Menge $\mathbb{Z}_{21}^*$ enthält $\varphi(21) = 12$ Elemente für die Wahl von a. Von insgesamt 12 Berechnungsmöglichkeiten für das Verfahren führen 6 Versuche zum Erfolg. Das ergibt eine Erfolgsrate von 50%.

Aus Beispiel 6 lassen sich zwei Aussagen ableiten, die zur Verallgemeinerung der Wahl von a herangezogen werden können:

- Die Zahl $a = 1$ führt immer zu einem ungeraden $r = 1$ und daher zu einem erfolglosen Testfall.
- Die Zahl $a = n - 1$ führt immer zu $r = 2$ und wegen $a^{\frac{r}{2}} = n - 1 = -1 \pmod{n}$ zu einem erfolglosen Testfall.

Beläßt man $a = 1$ und $a = n - 1$ trotzdem im Kreis der möglichen Kandidaten für die Wahl von a, der durch die Menge $\mathbb{Z}_n^*$ definiert ist, und berechnet die Erfolgswahrscheinlichkeit für ein Gelingen des Verfahrens wie in Beispiel 6, so ist diese Erfolgswahrscheinlichkeit durch Gl.(15) definiert, wie einige Beispiele in der Tabelle 8 zeigen.

n	m	$\varphi(n)$	Erfolgreich	Erfolgsrate
$15 = 3 \cdot 5$	2	8	6	3/4
$35 = 5 \cdot 7$	2	24	18	3/4
$45 = 3^2 \cdot 5$	2	24	18	3/4
$51 = 3 \cdot 17$	2	32	30	15/16
$105 = 3 \cdot 5 \cdot 7$	3	48	42	7/8
$165 = 3 \cdot 5 \cdot 11$	3	80	70	7/8
$195 = 3 \cdot 5 \cdot 13$	3	96	90	15/16
$225 = 3^2 \cdot 5^2$	2	120	90	3/4

Tabelle 8: Hohe Erfolgsrate des Verfahrens von G. L. Miller

Tabelle 9 zeigt einige Beispiele, die das Gleichheitszeichen innerhalb des „$\geq$“ in Gl.(15) belegen. Interessant hierbei ist, daß es sich ausschließlich um Zahlenwerte von n handelt, deren Faktoren ausschließlich Primzahlen der Form $p = 3 + 4 \cdot k$ (3, 7, 11, 19, 23, 31, 43, ...) sind. Die Untersuchung dieser Eigenschaft ist nicht Gegenstand dieser Arbeit, wird aber im Anhang (Kapitel 8.1) näher beschrieben.

n	m	$\varphi(n)$	Erfolgreich	Erfolgsrate
$21 = 3 \cdot 7$	2	12	6	1/2
$33 = 3 \cdot 11$	2	20	10	1/2
$57 = 3 \cdot 19$	2	36	18	1/2
$63 = 3^2 \cdot 19$	2	36	18	1/2
$77 = 7 \cdot 11$	2	60	30	1/2
$133 = 7 \cdot 19$	2	108	54	1/2
$209 = 11 \cdot 19$	2	180	90	1/2
$231 = 3 \cdot 7 \cdot 11$	3	120	90	3/4
$363 = 3 \cdot 11^2$	2	110	220	1/2
$399 = 3 \cdot 7 \cdot 19$	3	216	162	3/4
$627 = 3 \cdot 11 \cdot 19$	3	360	270	3/4
$1463 = 7 \cdot 11 \cdot 19$	3	1080	810	3/4
$4389 = 3 \cdot 7 \cdot 11 \cdot 19$	4	2160	1890	7/8

Tabelle 9: Grenzwertige Erfolgsrate des Verfahrens von G. L. Miller

Tabelle 10 zeigt einige Beispiele für das Verfahren von G. L. Miller. Für große Werte von n werden aus der Vielzahl der möglichen Startwerte von a nur einige wenige dargestellt.

n	a	r	Ergebnis
15	1	1	r ungerade
15	$2, 7, 8, 13$	4	$3 \cdot 5$
15	$4, 11$	2	$3 \cdot 5$
15	14	2	$a^{\frac{r}{2}} = -1 \pmod{n}$
21	1	1	r ungerade
21	$2, 10, 11, 19$	6	$3 \cdot 7$
21	$4, 16$	3	r ungerade
21	$5, 17$	6	$a^{\frac{r}{2}} = -1 \pmod{n}$
21	$8, 13$	2	$3 \cdot 7$
21	20	2	$a^{\frac{r}{2}} = -1 \pmod{n}$
33	1	1	r ungerade
33	$2, 8, 17, 29$	10	$a^{\frac{r}{2}} = -1 \pmod{n}$
33	$4, 16, 25, 31$	5	r ungerade
33	$5, 7, 13, 14, 19, 20, 26, 28$	10	$3 \cdot 11$
33	$10, 23$	2	$3 \cdot 11$
33	32	2	$a^{\frac{r}{2}} = -1 \pmod{n}$
35	1	1	r ungerade
35	$2, 3, 12, 17, 18, 23, 32, 33$	12	$5 \cdot 7$
35	$4, 9, 26, 31$	6	$5 \cdot 7$
35	$6, 29$	2	$5 \cdot 7$
35	$8, 13, 22, 27$	4	$5 \cdot 7$
35	$11, 16$	3	r ungerade
35	$19, 24$	6	$a^{\frac{r}{2}} = -1 \pmod{n}$
35	34	2	$a^{\frac{r}{2}} = -1 \pmod{n}$
39	1	1	r ungerade
39	$2, 7, 11, 19, 20, 28, 32, 37$	12	$3 \cdot 13$
39	$4, 10, 29, 35$	6	$3 \cdot 13$
39	$5, 8, 31, 34$	4	$3 \cdot 13$
39	$14, 25$	2	$3 \cdot 13$
39	$16, 22$	3	r ungerade
39	$17, 23$	6	$a^{\frac{r}{2}} = -1 \pmod{n}$
39	38	2	$a^{\frac{r}{2}} = -1 \pmod{n}$

Tabelle 10: Beispiele des Verfahrens von G. L. Miller

2.2.3 Einschränkungen

Das Verfahren von G. L. Miller kann nicht angewendet werden für reine Potenzen von Primzahlen, d.h. für $n = p^k$, wobei p ungerade Primzahl und k eine natürliche Zahl ist.
Mit anderen Worten: für $n = p^k$ ist entweder r ungerade oder $a^{\frac{r}{2}} = -1 (\mod\ n)$. Damit existiert für n kein a, das die Voraussetzungen von Gl.(15) erfüllt. Warum das so ist, wird im folgenden gezeigt.

Es wird zunächst das Gegenteil behauptet. D.h. es wird angenommen, daß für $n = p^k$ ein a existiert, das Gl.(15) erfüllt, und daß das Ergebnis des Shor-Verfahrens in einer Zerlegung von $n = n_1 n_2$ in $n_1 = p^{k_1}$ und $n_2 = p^{k_2}$ mündet. Dabei gilt $k = k_1 + k_2$ und $1 \leq k_1 < k$ und $1 \leq k_2 < k$.

Setzt man n_1 und n_2 in Gl.(10) ein, dann erhält man

$$a^{\frac{r}{2}} - 1 = c_1 p^{k_1} \ \wedge \ a^{\frac{r}{2}} + 1 = c_2 p^{k_2}$$

Subtrahiert man die linke von der rechten Gleichung, dann erhält man

$$c_2 p^{k_2} - c_1 p^{k_1} \ = \ 2$$

Hierbei ergeben sich zwei Fälle.

Fall 1: Es gilt $k_2 \geq k_1$. Dann enthält die Gleichung

$$c_2 p^{k_2} - c_1 p^{k_1} - 2 \ = \ 0$$

eine ganzzahlige Nullstelle p, wenn $k_2 = 2k_1$ und p Teiler von 2 ist. Dies wird deutlich erkennbar, wenn man die Nullgleichung in

$$(ap^{b_1} + \tfrac{2}{x})(p^{b_2} - x) \ = \ ap^{b_1+b_2} - (axp^{b_1} - \tfrac{2}{x}p^{b_2}) - 2 \ = \ 0$$

umformt. Hierbei sind b_1, b_2 und x natürliche Zahlen. Damit das Produkt zweier Summen nicht insgesamt vier Terme ergibt, muß $b_1 = b_2 = b$ gelten. Damit gilt:

$$(ap^b + \tfrac{2}{x})(p^b - x) \ = \ ap^{2b} - (ax - \tfrac{2}{x})p^b - 2 \ = \ 0$$

Damit $(ax - \frac{2}{x})$ eine natürliche Zahl bleibt, muß x Teiler von 2 sein. Unter dieser Voraussetzung ergibt sich für $(p^b - x)$, daß $p = 1$ oder $p = 2$ gelten muß. Der erste Wert ist uninteressant für das Verfahren, und der zweite Wert ist ein Widerspruch, da $n = p^k$ nicht gerade ist.

Fall 2: Es gilt $k_1 > k_2$. Dann kann die Gleichung

$$c_1 p^{k_1} - c_2 p^{k_2} + 2 \ = \ 0$$

analog zum Fall 1 in

$$(ap^b - \tfrac{2}{x})(p^b - x) \ = \ ap^{2b} - (ax + \tfrac{2}{x})p^b - 2 \ = \ 0$$

umgeformt werden. Auch hier muß, damit $(ax + \frac{2}{x})$ eine natürliche Zahl bleibt, x Teiler von 2 sein. Damit ergibt sich auch hier der Widerspruch für $p = 2$, daß $n = p^k$ nicht gerade ist.

Fazit: Damit ist bewiesen, daß das Verfahren von G. L. Miller für $n = p^k$ nicht geeignet ist, da es die Voraussetzungen von Gl.(15) nicht erfüllt (q.e.d.). Allerdings kann das Verfahren sehr wohl für Potenzen von Nicht-Primzahlen angewendet werden. Hierzu sei Beispiel 2 aus Kapitel 2.2.2 erwähnt, in dem $n = 225 = 15^2$ in das Produkt $9 \cdot 25$ zerlegt wird.

2.2.4 Computational Resources

Um die Computational Resources für das Verfahren von G. L. Miller zu ermitteln, ist es notwendig, unter Zuhilfenahme der Euler-Funktion $\varphi(n)$ noch zusätzliche Eigenschaften darzustellen.

Jede Primzahl p ist teilerfremd zu allen Zahlen von 1 bis $p - 1$. Hier gilt:

$$\varphi(p) = p - 1 \tag{18}$$

Für zwei teilerfremde Zahlen n_1 und n_2 und deren Produkt $n = n_1 n_2$ gilt:

$$\varphi(n) = \varphi(n_1)\varphi(n_2) \tag{19}$$

Diese Gleichung kann mit Hilfe des „Chinesischen Resttheorems“ (siehe [NI]) nachgewiesen werden. Zum Beispiel gilt für die Zahl 15:

$$\mathbb{Z}_{15}^* = \{1, 2, 4, 7, 8, 11, 13, 14\}: \ \varphi(15) = \varphi(3) \cdot \varphi(5) = 2 \cdot 4 = 8$$

Der Schweizer Mathematiker *Leonhard Euler* (1707 - 1783) hat nachgewiesen, daß für alle a mit $a \in \mathbb{Z}_n^*$ gilt:

$$a^{\varphi(n)} = 1 \pmod{n} \tag{20}$$

Für das oben beschriebene Verfahren von G. L. Miller bedeutet dies, daß für die Ordnung r von a modulo n eine obere Grenze besteht. Ist n Primzahl, dann nimmt $r = \mathrm{ord}_n a$ den maximalen Wert $n - 1$ an oder ist Teiler davon (Siehe [NI]: die Ordnung r teilt $\varphi(n)$).

Das heißt, das Verfahren verwendet für die Berechnung von a^r höchstens eine Anzahl von $n-1$ Multiplikationen und für die Berechnung der Divisionen modulo n eine gleiche Anzahl an Operationen.

Teilt man die Zahl n auf L Bits auf [siehe Gl.(21) in Kapitel 3.1], dann handelt es sich um $O(2^L)$ Multiplikationen und $O(2^L)$ Divisionen mit Rest (siehe Kapitel 3.5). Sowohl die Multiplikationen als auch die Divisionen benötigen jeweils $O(L^2)$ Operationen, so daß die Gesamtzahl der verbrauchten Resourcen zur Ermittlung der Ordnung r eine Anzahl von $O(L^2 2^L)$ ergibt.

Die Berechnung der größten gemeinsamen Teiler (ggT) zur Ermittlung der Faktoren von n verbraucht $O(L^3)$ Operationen (siehe [NI]) und fällt bei der Gesamtbetrachtung nicht ins Gewicht.

3 Mathematische Grundlagen

In diesem Kapitel werden einige mathematischen Grundlagen erläutert, die für das Verständnis des Shor-Verfahrens von Bedeutung sind.

3.1 L-Bit-Arithmetik: Die Zahlen n, m und L

In diesem Kapitel wird der Zusammenhang zwischen der Zahl n, der Bit-Dimension L von n und der Bit-Dimension m, die im Shor-Verfahren eine wichtige Rolle spielt, hergestellt.

Zusammenhang zwischen n und L

Die folgende Tabelle 11 gibt Auskunft, welche natürlichen Zahlen n sich durch wieviel Bits L darstellen lassen.

L	n	
1	$0 \leq n \leq 1$	
2	$2 \leq n \leq 3$	
3	$4 \leq n \leq 7$	
4	$8 \leq n \leq 15$	
$L \geq 2$	$2^{L-1} \leq n \leq 2^L - 1$	$n \geq 2$

Tabelle 11: Bitdarstellung der natürlichen Zahlen

Aufrundung: Mit dem Klammerpaar $\lceil\ \rceil$, das die nächste nach oben gerundete natürliche Zahl ermittelt, läßt sich für $n \geq 2$ und $L \geq 2$ der Zusammenhang zwischen n und L folgendermaßen herleiten:

$$2^{L-1} \leq n \leq 2^L - 1$$

$$\Longrightarrow 2^{L-1} < 2^{L-1} + 1 \leq n + 1 \leq 2^L$$

$$\Longrightarrow L - 1 < \log_2(n+1) \leq L$$

$$\Longrightarrow L = \lceil \log_2(n+1) \rceil$$

Abrundung: Mit dem Klammerpaar $\lfloor\ \rfloor$, das die nächste nach unten gerundete natürliche Zahl ermittelt, läßt sich für $n \geq 2$ und $L \geq 2$ der Zusammenhang zwischen n und L auch anders darstellen:

$$2^{L-1} \leq n \leq 2^L - 1 < 2^L$$

$$\Longrightarrow L - 1 \leq \log_2(n) < L$$

$$\Longrightarrow L = 1 + \lfloor \log_2(n) \rfloor$$

Sowohl beim Auf- wie beim Abrundung ist zu beachten, daß die Herleitungen nur für $n \geq 2$ und $L \geq 2$ gelten. Damit gilt insgesamt:

$$2^{L-1} \leq n \leq 2^L - 1 \iff L = \lceil \log_2(n+1) \rceil = 1 + \lfloor \log_2(n) \rfloor \quad (21)$$

Zwei Beispiele:

$$n = 15 \implies L = \lceil \log_2(16) \rceil = 1 + \lfloor \log_2(15) \rfloor = 4$$
$$n = 16 \implies L = \lceil \log_2(17) \rceil = 1 + \lfloor \log_2(16) \rfloor = 5$$

Zusammenhang zwischen n und m

Die Bit-Dimension m, die im Shor-Verfahren eine große Rolle spielt, ist durch

$$n^2 \leq 2^m < 2n^2$$

definiert. Durch Umformung ergibt sich

$$2\log_2(n) \leq m < 1 + 2\log_2(n)$$

$$\implies \lceil 2\log_2(n) \rceil = m < 1 + \lceil 2\log_2(n) \rceil$$

Damit gilt insgesamt:

$$n^2 \leq 2^m < 2n^2 \implies m = \lceil 2\log_2(n) \rceil \quad (22)$$

Zwei Beispiele:

$$n = 16 \implies m = \lceil 2\log_2(16) \rceil = 8$$
$$n = 17 \implies m = \lceil 2\log_2(17) \rceil = 9$$

Zusammenhang zwischen L und m

Um die Beziehung zwischen L und m herzustellen, ist es ratsam n-Werte in der Nachbarschaft von 2^k zu beobachten. Siehe Tabelle 12.

n	L	m
31	5	10
32	6	10
33	6	11
63	6	12
64	7	12
65	7	13

Tabelle 12: Beispiele für den Zusammenhang zwischen n, m und L

Anhand dieser Tabelle läßt sich verallgemeinern, daß für $n \neq 2^k$

$$L = \lfloor \frac{m+1}{2} \rfloor \quad (23)$$

gilt. (Der Wert $n = 2^k$ ist für das Shor-Verfahren uninteressant.) Wegen dieser einfachen linearen Abhängigkeit lassen sich alle operativen Aufwände gleichermaßen über L und m angeben.

3.2 Definition von q-Bits und Multi-q-Bits

Bit: Ein Bit ist ein Element einer Menge, die nur aus zwei Elementen besteht. Sei das Bit mit b bezeichnet und die zugehörige Bit-Menge mit B, dann läßt sich ein Bit u.a. folgendermaßen darstellen.

- Für $B = \{0,1\}$ und $b \in B$ ist $b = 0$ oder $b = 1$.
- Für $B = \{v_1, v_2\}$ und $b \in B$ ist $b = v_1$ oder $b = v_2$.
- Für $B = \{|0>, |1>\}$ und $|b> \in B$ ist $|b> = |0>$ oder $|b> = |1>$.
- Für $B = \{\underline{e}_1, \underline{e}_2\}$ und $\underline{b} \in B$ ist $\underline{b} = \underline{e}_1$ oder $\underline{b} = \underline{e}_2$.

q-Bit: Sei $B = \{v_1, v_2\}$ eine orthonormale Basis für einen Hilbertraum $\mathcal{H}$ der Dimension $d = 2$ über $\mathbb{C}$. Dann wird in diesem Raum (dieser Ebene) der Einheitskreis als Sphäre

$$\mathcal{S}^1_{\mathcal{H}} = \{v = \lambda_1 v_1 + \lambda_2 v_2 \in \mathcal{H} : |\lambda_1|^2 + |\lambda_2|^2 = 1\} \tag{24}$$

bezeichnet. Jedes Element $b \in \mathcal{S}^1_{\mathcal{H}}$ dieses Einheitskreises nennt man q-Bit.

In Anlehnung an die vier o.g. Darstellungsmöglichkeiten von Bits lassen sich für q-Bits u.a folgende Äquivalenzbeziehungen bilden:

$$\begin{aligned} 0 &\Leftrightarrow v_1 \Leftrightarrow |0> \Leftrightarrow \underline{e}_1 \\ 1 &\Leftrightarrow v_2 \Leftrightarrow |1> \Leftrightarrow \underline{e}_2 \end{aligned}$$

Im Unterschied zum einfachen Bit, das nur einen von zwei möglichen Werten annehmen kann, ist das q-Bit eine 1-normierte Linearkombination aus diesen beiden Basiswerten. Die Koeffizienten λ_1 und λ_2 sind frei wählbar, solange sie der Bedingung der Sphärengleichung Gl.(24) genügen.

Wahrscheinlichkeit: Um einen Bezug vom q-Bit zum einfachen Bit zu erhalten, läßt sich für die Koeffizienten λ_1 und λ_2 folgendes Wahrscheinlichkeitsmaß definieren:

$$P\{v_1\} = |\lambda_1|^2 \ , \ P\{v_2\} = |\lambda_2|^2 \tag{25}$$

D.h. die Wahrscheinlichkeit, daß $v = \lambda_1 v_1 + \lambda_2 v_2$ den Wert v_1 annimmt, ist gleich $|\lambda_1|^2$, und die Wahrscheinlichkeit, daß v den Wert v_2 annimmt, ist gleich $|\lambda_2|^2$.

Multi-Bit: Sei B definiert als Bit-Menge, d.h. als Menge von zwei Elementen. Dann ist ein Multi-Bit x der Dimension n ein Element der Menge B^n. Es existieren insgesamt also 2^n Elemente $x \in B^n$.

Für $n = 2$ läßt sich eine Multi-Bit-Menge u.a. folgendermaßen darstellen.

- Für $B = \{0,1\}$ ist $B^2 = \{(0,0), (1,0), (0,1), (1,1)\}$.
- Für $B = \{v_1, v_2\}$ ist $B^2 = \{(v_1, v_1), (v_2, v_1), (v_1, v_2), (v_2, v_2)\}$.

- Für $B = \{|0>, |1>\}$ ist $B^2 = \{|0>|0>, |1>|0>, |0>|1>, |1>|1>\}$.
- Für $B = \{\underline{e}_1, \underline{e}_2\}$ ist $B^2 = \{\underline{e}_1, \underline{e}_2, \underline{e}_3, \underline{e}_4\}$.

Multi-q-Bit: Sei $d = n$ die Bit-Dimension, $N = 2^n$ die vektorielle Dimension und $B = \{\underline{e}_1, \ldots, \underline{e}_N\}$ eine orthonormale Basis für einen Hilbertraum $\mathcal{H}^{\otimes n}$ in $\mathbb{C}$. Dann wird in diesem Raum die Einheitskugel als Sphäre

$$\mathcal{S}^1_{\mathcal{H}^{\otimes n}} = \{\underline{v} = \lambda_1 \underline{e}_1 + \ldots + \lambda_N \underline{e}_N \in \mathcal{H} : |\lambda_1|^2 + \ldots + |\lambda_N|^2 = 1\} \tag{26}$$

bezeichnet. Jedes Element $x \in \mathcal{S}^1_{\mathcal{H}^{\otimes n}}$ dieser Einheitskugel nennt man Multi-q-Bit.

In Anlehnung an die vier o.g. Darstellungsmöglichkeiten von Multi-Bits lassen sich für Multi-q-Bits Äquivalenzbeziehungen bilden, wie in den folgenden Beziehungen für $n = 2$ und $N = 4$ gezeigt wird.

$$\begin{aligned}
(0,0) &\Leftrightarrow (v_1, v_1) \Leftrightarrow |0>|0> \Leftrightarrow \underline{e}_1 \\
(1,0) &\Leftrightarrow (v_2, v_1) \Leftrightarrow |1>|0> \Leftrightarrow \underline{e}_2 \\
(0,1) &\Leftrightarrow (v_1, v_2) \Leftrightarrow |0>|1> \Leftrightarrow \underline{e}_3 \\
(1,1) &\Leftrightarrow (v_2, v_2) \Leftrightarrow |1>|1> \Leftrightarrow \underline{e}_4
\end{aligned}$$

Im Unterschied zum einfachen Multi-Bit, das nur einen von $N = 2^n$ möglichen Werten annehmen kann, ist das Multi-q-Bit eine 1-normierte Linearkombination aus allen diesen Basiswerten. Die Koeffizienten λ_1 bis λ_N sind frei wählbar, solange sie der Bedingung der Sphärengleichung Gl.(26) genügen.

Wahrscheinlichkeit: Um einen Bezug vom Multi-q-Bit zum einfachen Multi-Bit zu erhalten, läßt sich für die Koeffizienten λ_1 bis λ_N folgendes Wahrscheinlichkeitsmaß definieren:

$$P\{\underline{e}_1\} = |\lambda_1|^2 \ , \ \ldots \ , \ P\{\underline{e}_N\} = |\lambda_N|^2 \tag{27}$$

D.h. die Wahrscheinlichkeit, daß $\underline{v} = \lambda_1 \underline{e}_1 + \ldots + \lambda_N \underline{e}_N$ den Wert $\underline{e}_1$ annimmt, ist gleich $|\lambda_1|^2$, und die Wahrscheinlichkeit, daß $\underline{v}$ den Wert $\underline{e}_k$ annimmt, ist gleich $|\lambda_k|^2$.

Argument- und Werte-Hibertraum: In Büchern über Quanten-Rechner, die von Physikern geschrieben werden, ist die sogenannte „Dirac-Notation", d.h. die Darstellung mit der Bit-Menge $B = \{|0>, |1>\}$ am weitesten verbreitet. Zu dieser Notationsart gibt es noch weitere Konventionen, die für diese Arbeit von Bedeutung sind und daher hier beschrieben werden.

Sei $\mathcal{H}^{\otimes n}$ ein Hilbertraum, der definiert wird über eine orthonormale Basis B^n mit $B = \{|0>, |1>\}$. Seine Elemente, oder Basisvektoren, lassen sich darstellen durch

$$|b> = |b_1>|b_2> \ldots |b_n> \tag{28}$$

wobei $b_k = 0$ oder $b_k = 1$ ist. Dabei ist b_1 das LSB (least significant bit) und b_n das MSB (most significant bit). Damit läßt sich jeder Basisvektor über $N = 2^n$ auch darstellen durch

$$|b>=|S> \quad \text{mit} \quad S \;=\; \sum_{k=1}^{N} b_k \; 2^{k-1} \quad \text{und} \quad 0 \leq S \leq 2^n - 1 \tag{29}$$

Zwei Beispiele für $n = 4$ oder $N = 16$:

$$|b>=|1>|1>|0>|1>=|11>$$
$$|b>=|1>|0>|1>|0>=|5>$$

Multi-q-Bit-Hilberträume der Dimension n lassen sich als Tensorprodukt von mehreren Multi-q-Bit-Hilberträumen zerlegen. Einen Multi-q-Bit-Hilbertraum, der aus dem Tensorprodukt von zwei Multi-q-Bit-Hilberträumen besteht, stellt die folgende Gleichung dar:

$$\mathcal{H}^{\otimes n} = \mathcal{H}^{\otimes n_1} \otimes \mathcal{H}^{\otimes n_2} = \mathcal{H}_1 \otimes \mathcal{H}_2 \tag{30}$$

Hierbei gilt selbstverständlich $n = n_1 + n_2$. Bei diesen Teil-Hilberträumen spricht man auch von „Registern“.

Das Register mit der Dimension n_1 enthält das LSB und wird in [CR] **Argument-Hilbertraum** genannt. Auf dem Quanten-Rechner werden in diesem Register Gate-Operationen, wie z.B. Hadamard-Gate-Operationen oder inverse Fourier-Transformationen (siehe hierzu Kapitel 3.4), durchgeführt.

Das Register mit der Dimension n_2 enthält das MSB und wird in [CR] **Werte-Hilbertraum** genannt. Auf dem Quanten-Rechner werden in diesem Register in Form von Black-Box-Gates Funktionen oder Funktionsräume definiert. Aus diesem Grund wird dieses Register oft auch „Black-Box-Register“ oder „Orakel-Register“ genannt.

Mit diesen zwei Registern läßt sich das Multi-q-Bit darstellen durch

$$|b> \;=\; |S> \;=\; |S_1>|S_2> \quad \text{wobei} \quad S \;=\; S_1 + S_2 2^{n_1} \tag{31}$$

$$0 \;\leq\; S_1 \;\leq\; 2^{n_1} - 1 \;\;,\;\; 0 \;\leq\; S_2 \;\leq\; 2^{n_2} - 1 \tag{32}$$

Zwei Beispiele für $n_1 = n_2 = 2$:

$$|b>=|3>|2>=|11>$$
$$|b>=|1>|1>=|5>$$

Der gesamte Hilbertraum, als Tensorprodukt $\mathcal{H}_1 \otimes \mathcal{H}_2$ dargestellt, wird in [CR] auch **Rechner-Hilbertraum** genannt.

3.3 Das Wahrscheinlichkeitsmodell für q-Bits und Multi-q-Bits

Wählt man für q-Bits und Multi-q-Bits die mathematische Darstellung, also die Vektorform

$$\underline{v} = \sum_{k=1}^{N} \lambda_k \underline{e}_k \quad \text{mit} \quad \sum_{k=1}^{N} |\lambda_k|^2 = 1, \tag{33}$$

dann kann unter Berufung auf Kapitel 3.2 für das Multi-q-Bit folgendes Wahrscheinlichkeitsmaß definiert werden:

$$P\{\underline{e}_k\} = |\lambda_k|^2 \tag{34}$$

Da $\underline{e}_k$ Basisvektor des Hilbertraums ist, wird die korrespondierende Wahrscheinlichkeit im folgenden auch **Basis-Wahrscheinlichkeit** genannt.

Selbstverständlich kann die Berechnung der Wahrscheinlichkeit auf die Summe einiger Basisvektoren angewendet werden. Z.B. ist die Wahrscheinlichkeit, daß die Messung von $\underline{v}$ den Wert $\underline{e}_k$ oder $\underline{e}_l$ ergibt, gleich

$$P\{\underline{e}_k, \underline{e}_l\} = P\{\underline{e}_k\} + P\{\underline{e}_l\} = |\lambda_k|^2 + |\lambda_l|^2 \tag{35}$$

Die Summe aller Basis-Wahrscheinlichkeiten ist 1, wie Gl.(33) belegt. Wie bereits in Kapitel 3.2 erwähnt, wird in Quanten-Algorithmen mit Argument- und Werte-Hilberträumen gearbeitet. Um mit Hilfe von Gl.(35) Basisvektoren auf Argument- und Werte-Hilberträume abzubilden, ist es hilfreich, von der mathematischen Darstellung auf die physikalische Dirac-Notation umzusteigen:

$$\underline{e}_{a+1} \iff |a> \quad \text{für} \quad a \in [0, N-1] \tag{36}$$

Hierbei ist $|a>$ Basisvektor im Rechner-Hilbertraum, der, wie auch in Kapitel 3.2 erwähnt, als Tensorprodukt von Argument- und Werte-Hilbertraum dargestellt werden kann.

$$\underline{e}_{1+x+N_1 y} \iff |x>|y> \quad \text{für} \quad x \in [0, N_1-1] \quad \text{und} \quad y \in [0, N_2-1] \tag{37}$$

In dieser Gleichung sind $N_1 = 2^{b_1}$ und $N_2 = 2^{b_2}$, wobei b_1 die Bitanzahl des Argument-Hilbertraums und b_2 die Bitanzahl des Werte-Hilbertraums ist. Selbstverständlich gilt $N = N_1 N_2$. Die Wahrscheinlichkeit für den Basisvektor $|x>|y>$ wird durch $P\{|x>|y>\}$ beschrieben.

Wie groß ist die Wahrscheinlichkeit für alle Basisvektoren mit gleichem x ?

$$P\{|x>\} = \sum_{y=0}^{N_2-1} P\{|x>|y>\} \tag{38}$$

$P\{|x>\}$ ist die Wahrscheinlichkeit für ein $x \in \mathcal{H}_1$. Da $\mathcal{H}_1$ den Argument-Hilbertraum darstellt, wird $P\{|x>\}$ im folgenden **Argument-Wahrscheinlichkeit** genannt. Die Summe aller Argument-Wahrscheinlichkeiten ist 1.

3.4 Gates

Klassische Computer bestehen aus *Leitungen* und *Gates*. Leitungen tragen Information unverändert von einem Ende eines Systems zum anderen, also durch das System hindurch, während logische Gates die Information manipulieren. Zum Beispiel wird die Bitfolge

$$(0,1,0,0,1) = (v_1, v_2, v_1, v_1, v_2) = \underline{e}_{19}$$

endlicher Länge durch Anwendung elementarer logischer Gates verändert zu

$$(0,1,0,0,1) \;\rightarrow\; (0,1,1,1,0) \;\Leftrightarrow\; \underline{e}_{19} \;\rightarrow\; \underline{e}_{15}.$$

Dabei werden Ein-Bit-, Zwei-Bit- oder Mehr-Bit-Operationen ausgeführt.

Ein *Gate* ist somit eine einfache oder zusammengesetzte Operation über ein Bit oder über mehrere Bits. In der mathematischen Darstellung wird diese Operation durch eine Transformationsmatrix beschrieben. Bei einem *Quanten-Gate* ist die Transformationsmatrix unitär.

Hadamard-Gate: Das Hadamard-Gate ist ein (unitäres) Quanten-Gate, das genau ein Bit manipuliert, somit also ein Ein-Bit-Quanten-Gate.

$$H \cdot |x> = \frac{1}{\sqrt{2}} \left[|0> + (-1)^x |1> \right] \tag{39}$$

Die Matrixform des Hadamard-Gates lautet:

$$\underline{\underline{H}} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{40}$$

Durch die Matrixdarstellung ist gewährleistet, daß

$$\begin{aligned} H \cdot |0> &= \tfrac{1}{\sqrt{2}}[|0> + |1>] \;\Leftrightarrow\; \underline{\underline{H}}\,\underline{e}_1 = \tfrac{1}{\sqrt{2}}[\underline{e}_1 + \underline{e}_2] \\ H \cdot |1> &= \tfrac{1}{\sqrt{2}}[|0> - |1>] \;\Leftrightarrow\; \underline{\underline{H}}\,\underline{e}_2 = \tfrac{1}{\sqrt{2}}[\underline{e}_1 - \underline{e}_2] \end{aligned}$$

ist. Diese Darstellung entspricht somit der Gl.(39).

R_k-Gate: Das R_k-Gate ist auch ein (unitäres) Ein-Bit-Quanten-Gate.

$$R_k \cdot |x> = exp(i\frac{2\pi x}{2^k}) \cdot |x> \tag{41}$$

Die Matrixform des R_k-Gate lautet:

$$\underline{\underline{R}}_k = \begin{pmatrix} 1 & 0 \\ 0 & exp(i\frac{2\pi}{2^k}) \end{pmatrix} \tag{42}$$

Durch die Matrixdarstellung ist gewährleistet, daß

$$\begin{aligned} R_k \cdot |0> &= |0> \quad \Leftrightarrow\; \underline{\underline{R}}_k\,\underline{e}_1 = \underline{e}_1 \\ R_k \cdot |1> &= exp(i\tfrac{2\pi}{2^k}) \cdot |1> \;\Leftrightarrow\; \underline{\underline{R}}_k\,\underline{e}_2 = exp(i\tfrac{2\pi}{2^k}) \cdot \underline{e}_2 \end{aligned}$$

ist. Diese Darstellung entspricht somit der Gl.(41).

Allgemeine Gates: Ein beliebiges Gate A kann folgende Form haben:

$$\underline{\underline{A}} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1N} \\ a_{21} & a_{22} & \cdots & a_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ a_{M1} & a_{M2} & \cdots & a_{MN} \end{pmatrix} \tag{43}$$

Ist $\underline{\underline{A}}$ unitär, dann ist es auch Quanten-Gate.

Überlagerung mehrerer nichtkorrelierter Gates: Um für die Überlagerung mehrerer nichtkorrelierter Ein-Bit-Gates eine gemeinsame Matrixdarstellung zu erhalten, ist es zunächst von Interesse, die Überlagerung zweier unkorrelierter Ein-Bit-Gates zu untersuchen. Die Ergebnisse hierzu lassen sich verallgemeinern auf mehrere Gates.

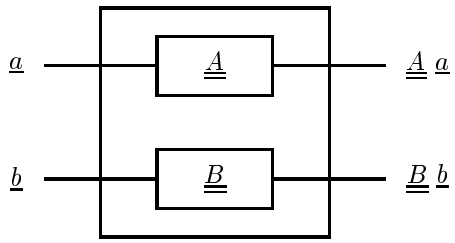

Abbildung 2: Überlagerung zweier nichtkorrelierter Gates

Seien die beiden Ein-Bit-Gates $\underline{\underline{A}}$ und $\underline{\underline{B}}$ gemäß Abbildung 2 definiert durch die Matrizen

$$\underline{\underline{A}} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \quad \underline{\underline{B}} = \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix} \tag{44}$$

Dann gilt in der Darstellung der Quantenbits:

$$\begin{aligned}
(v_1, v_1) \;&\rightarrow\; (a_{11}v_1 + a_{21}v_2,\ b_{11}v_1 + b_{21}v_2) = \\
&\quad\; a_{11}b_{11}(v_1, v_1) + a_{21}b_{11}(v_2, v_1) + a_{11}b_{21}(v_1v_2) + a_{21}b_{21}(v_2v_2) \\
\Longrightarrow\ \underline{e}_1 \;&\rightarrow\; a_{11}b_{11} \cdot \underline{e}_1 \;+\; a_{21}b_{11} \cdot \underline{e}_2 \;+\; a_{11}b_{21} \cdot \underline{e}_3 \;+\; a_{21}b_{21} \cdot \underline{e}_4 \\
\\
(v_2, v_1) \;&\rightarrow\; (a_{12}v_1 + a_{22}v_2,\ b_{11}v_1 + b_{21}v_2) = \\
&\quad\; a_{12}b_{11}(v_1, v_1) + a_{22}b_{11}(v_2, v_1) + a_{12}b_{21}(v_1v_2) + a_{22}b_{21}(v_2v_2) \\
\Longrightarrow\ \underline{e}_2 \;&\rightarrow\; a_{12}b_{11} \cdot \underline{e}_1 \;+\; a_{22}b_{11} \cdot \underline{e}_2 \;+\; a_{12}b_{21} \cdot \underline{e}_3 \;+\; a_{22}b_{21} \cdot \underline{e}_4 \\
\\
(v_1, v_2) \;&\rightarrow\; (a_{11}v_1 + a_{21}v_2,\ b_{12}v_1 + b_{22}v_2) = \\
&\quad\; a_{11}b_{12}(v_1, v_1) + a_{21}b_{12}(v_2, v_1) + a_{11}b_{22}(v_1v_2) + a_{21}b_{22}(v_2v_2) \\
\Longrightarrow\ \underline{e}_3 \;&\rightarrow\; a_{11}b_{12} \cdot \underline{e}_1 \;+\; a_{21}b_{12} \cdot \underline{e}_2 \;+\; a_{11}b_{22} \cdot \underline{e}_3 \;+\; a_{21}b_{22} \cdot \underline{e}_4 \\
\\
(v_2, v_2) \;&\rightarrow\; (a_{12}v_1 + a_{22}v_2,\ b_{12}v_1 + b_{22}v_2) = \\
&\quad\; a_{12}b_{12}(v_1, v_1) + a_{22}b_{12}(v_2, v_1) + a_{12}b_{22}(v_1v_2) + a_{22}b_{22}(v_2v_2) \\
\Longrightarrow\ \underline{e}_4 \;&\rightarrow\; a_{12}b_{12} \cdot \underline{e}_1 \;+\; a_{22}b_{12} \cdot \underline{e}_2 \;+\; a_{12}b_{22} \cdot \underline{e}_3 \;+\; a_{22}b_{22} \cdot \underline{e}_4
\end{aligned}$$

Als Gesamtübertragungsfunktion ergibt sich dann folgende Matrix

$$\begin{aligned} \underline{\underline{A}} \otimes \underline{\underline{B}} &= \begin{pmatrix} a_{11}b_{11} & a_{12}b_{11} & a_{11}b_{12} & a_{12}b_{12} \\ a_{21}b_{11} & a_{22}b_{11} & a_{21}b_{12} & a_{22}b_{12} \\ a_{11}b_{21} & a_{12}b_{21} & a_{11}b_{22} & a_{12}b_{22} \\ a_{21}b_{21} & a_{22}b_{21} & a_{21}b_{22} & a_{22}b_{22} \end{pmatrix} \\ &= \begin{pmatrix} \underline{\underline{A}}b_{11} & \underline{\underline{A}}b_{12} \\ \underline{\underline{A}}b_{21} & \underline{\underline{A}}b_{22} \end{pmatrix} \end{aligned} \tag{45}$$

Interpretieren läßt sich das Ergebnis folgendermaßen:
Man erzeugt zunächst die Matrix $\underline{\underline{A}}$ und überlagert diese dann mit den Elementen aus $\underline{\underline{B}}$. D.h. $\underline{\underline{A}}$ wird von $\underline{\underline{B}}$ überlagert. Hierzu zunächst ein paar Beispiele.

Beispiel 1: Überlagerung zweier Hadamard-Gates (siehe Abbildung 3). Das erste Hadamard-Gate wird überlagert mit den Elementen des zweiten Hamadard-Gates:

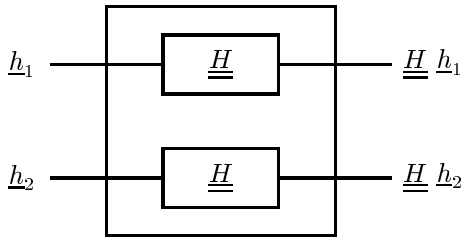

Abbildung 3: Überlagerung zweier Hadamard-Gates

$$\underline{\underline{H}} \otimes \underline{\underline{H}} = \underline{\underline{H}}^{\otimes 2} = \frac{1}{\sqrt{2}} \begin{pmatrix} \underline{\underline{H}} & \underline{\underline{H}} \\ \underline{\underline{H}} & -\underline{\underline{H}} \end{pmatrix} = \frac{1}{2} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix} \tag{46}$$

Beispiel 2: Überlagerung eines Hadamard-Gates mit der Einheitmatrix (siehe Abbildung 4).

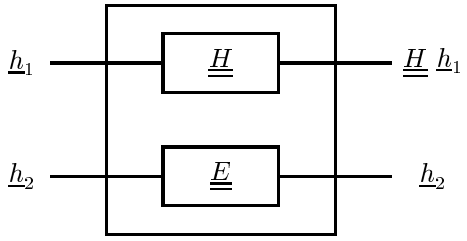

Abbildung 4: Überlagerung eines Hadamard-Gates mit der Einheitmatrix

$$\underline{\underline{H}} \otimes \underline{\underline{E}} = \begin{pmatrix} \underline{\underline{H}} & \underline{\underline{0}} \\ \underline{\underline{0}} & \underline{\underline{H}} \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \end{pmatrix} \tag{47}$$

Beispiel 3: Überlagerung eines Einheits-Gates mit einem Hadamard-Gate (siehe Abbildung 5).

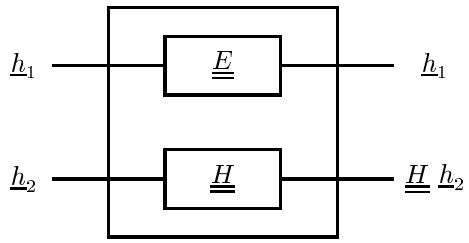

Abbildung 5: Überlagerung eines Einheits-Gates mit einem Hadamard-Gate

$$\underline{\underline{E}} \otimes \underline{\underline{H}} = \frac{1}{\sqrt{2}} \begin{pmatrix} \underline{\underline{E}} & \underline{\underline{E}} \\ \underline{\underline{E}} & -\underline{\underline{E}} \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{pmatrix} \tag{48}$$

3.5 Computational Resources auf dem klassischen Rechner

Die folgenden Eigenschaften von Rechenoperationen werden in [KN] ausführlich beschrieben und beziehen sich auf den klassischen Rechner.

Addition: Die Addition zweier ganzer Zahlen, von denen die größere L Bits lang ist, kostet auf einem klassischen Rechner $O(L)$ Rechenoperationen.

Subtraktion: Die Subtraktion zweier ganzer Zahlen, von denen die größere L Bits lang ist, kostet auf einem klassischen Rechner $O(L)$ Rechenoperationen.

Multiplikation: Die Multiplikation zweier ganzer Zahlen, von denen die größere L Bits lang ist, kostet auf einem klassischen Rechner $O(L^2)$ Rechenoperationen.

Division mit Rest: Die Divsion mit Rest einer L-Bit langen ganzen Zahl durch eine kleinere ganze Zahl kostet auf einem klassischen Rechner $O(L^2)$ Rechenoperationen.

Ein-Bit-Gates: Gates, die auf dem klassischen Rechner gemäß Kapitel 3.4 durch eine 2x2-Matrix beschrieben werden, beinhalten zwei Gleichungen mit jeweils 2 Multiplikationen und einer Addition. Insgesamt enthält ein Gate damit 4 Multiplikationen und zwei Additionen, wobei zu erwähnen ist, daß es sich hierbei um Operationen reellwertiger Zahlen handelt.

Größter gemeinsamer Teiler (ggT): Gemäß [NI] benötigt der Euclid-Algorithmus zum Ermitteln des ggT für zwei Zahlen, deren größte L Bits lang ist, $O(L)$ Divisionen mit Rest, somit insgesamt $O(L^3)$ Rechenoperationen auf einem klassischen Rechner.

Die folgende Rechenoperations-Eigenschaft läßt sich aus den vorhergehenden Eigenschaften ableiten.

Potenzieren modulo n: Die einmalige Berechnung von $a^k (\mathrm{mod}\ n)$, wobei a, k und n natürliche Zahlen sind und sich jeweils durch L Bits darstellen lassen, kostet auf einem klassischen Rechner $O(L^3)$ Operationen, denn:

- Das Umformen von k in eine Bitfolge der Form $(k_1 k_2 \ldots k_L)$ kostet $O(L)$ Divisionen mit Rest und damit $O(L^3)$ Rechen-Operationen.
- Das Berechnen von $a (\mathrm{mod}\ n)$, $a^2 (\mathrm{mod}\ n)$, $a^4 (\mathrm{mod}\ n)$, ..., $a^{2^L} (\mathrm{mod}\ n)$ kostet $O(L)$ Multiplikationen und Divisionen mit Rest und somit insgesamt $O(L^3)$ Rechen-Operationen.
- Das Berechnen von $a^k (\mathrm{mod}\ n)$ erfordert die selektive Multiplikation mit $a^{2^{k_\mu}} (\mathrm{mod}\ n)$. Dies hängt davon ab, ob $k_\mu = 1$ oder $k_\mu = 0$ ist. Dafür werden $O(L)$ Multiplikationen und Divisionen mit Rest und somit insgesamt $O(L^3)$ Rechen-Operationen benötigt.

Der Algorithmus zur Berechnung von $y = a^k (\mathrm{mod}\ n)$ ergibt sich damit zu:

1. Setze $x \leftarrow k$, $y \leftarrow 1$ und $i \leftarrow 0$.
2. Berechne $\mu \leftarrow x (\mathrm{mod}\ 2)$ und $x \leftarrow \lfloor \frac{x}{2} \rfloor$.
   Falls $i = 0$, dann berechne $a (\mathrm{mod}\ n)$.
   Falls $i > 0$, dann berechne $a^{2^i} = a^{2^{i-1}} a^{2^{i-1}} (\mathrm{mod}\ n)$.
3. Falls $\mu = 1$, dann berechne $y \leftarrow y \cdot a^{2^i} (\mathrm{mod}\ n)$.
4. Falls $x > 0$, dann setze $i \leftarrow i + 1$ und gehe nach Punkt 2.
   Falls $x = 0$, dann ist $y = a^k (\mathrm{mod}\ n)$ das gewünschte Ergebnis.
   Ende des Algorithmus.

3.6 Computational Resources auf dem Quanten-Rechner

Für die Berechnung der Anzahl der Rechen-Operationen auf dem Quanten-Rechner findet man in [NI] und [CR] folgende zwei wichtige Regeln:

1. Jedes Ein-Bit-Quanten-Gate (Hadamard-Gate, R_k-Gate, Invertier-Gate) zählt als eine Operation. Diese Zählweise wird belegt durch [NI], Seite 173 unten, Zitat: „... quantify the cost of an algorithm in terms of things like the total number of gates required...“.
2. Für die Berechnungen im Werte-Hilbertraum wird [CR] zitiert: „Es sollte allerdings betont werden, daß bei diesen Betrachtungen stets der Aufwand für die Berechnng der Funktion f *nicht* in Rechnung gestellt wird. Die Berechnung von f wird als klassische bzw. quantenmechanische Black-Box-Routine, als *Orakel* betrachtet, die unmittelbar das Resultat liefert. Den Aufwand messen wir daran, wie oft das klassische bzw. Quanten-Orakel befragt werden muß.“

3.7 Kettenbruch-Zerlegung

Jede positive rationale Zahl $x \in Q$ läßt sich in folgender Weise als Kettenbruch darstellen:

$$x \;=\; a_0 + \frac{1}{a_1 + \frac{1}{a_2 + \frac{1}{\cdots + \frac{1}{a_N}}}} \;=\; [a_0, a_1, a_2, \cdots, a_N] \;=\; \frac{p_N}{q_N} \tag{49}$$

Bei den Elementen a_k des Kettenbruch-Vektors handelt es sich um positive natürliche Zahlen. Ist $x < 1$, dann gilt $a_0 = 0$.

Kettenbruch-Vorwärtszerlegung: Hierbei handelt es sich um die Berechung des Kettenbruch-Vektors $[a_0, a_1, \cdots, a_N]$ aus der rationalen Zahl x. Hierzu ein Beispiel, in dem der Kettenbruch-Vektor zu $x = \frac{19}{17}$ gesucht wird.

$$x \;=\; \tfrac{19}{17} \;=\; 1 + \tfrac{2}{17} \;=\; 1 + \tfrac{1}{\frac{17}{2}} \;=\; 1 + \tfrac{1}{8 + \frac{1}{2}} \;=\; [1, 8, 2]$$

Durch diese Vorgehensweise ist immer gewährleistet, daß für das letzte Kettenglied immer $a_N \geq 2$ gilt. Setzt man allerdings

$$a_N \;=\; (a_N - 1) + \tfrac{1}{a_{N+1}}$$

dann entsteht ein neues letztes Kettenelement $a_{N+1} = 1$ und die Anzahl der Kettenbruchelemente erhöht sich um 1.

$$x \;=\; \tfrac{19}{17} \;=\; [1, 8, 1, 1]$$

D.h. der Kettenbruch-Vektor kann je nach Bedarf eine ungerade oder gerade Anzahl von Elementen enthalten.

Kettenbruch-Rückwärtsberechnung: Hierbei handelt es sich um die Ermittlung der rationalen Zahl $x = \frac{p_N}{q_N}$ aus dem Kettenbruch-Vektor. Gemäß [NI] kann dies mit Hilfe eines Algorithmus erreicht werden. Für

$$[a_0, a_1, a_2, \cdots, a_N] \;=\; \tfrac{p_N}{q_N}$$

ergeben sich folgende Start-Werte:

$$p_0 \;=\; a_0,\; q_0 \;=\; 1,\; p_1 \;=\; 1 + a_0 a_1,\; q_1 \;=\; a_1 \tag{50}$$

und für $2 \leq n \;\leq N$ folgende weitere Iterationsschritte:

$$p_n \;=\; a_n p_{n-1} + p_{n-2},\; q_n \;=\; a_n q_{n-1} + q_{n-2} \tag{51}$$

Hierzu das Beispiel aus der Kettenbruch-Vorwärtszerlegung. Mit $x = [1, 8, 2]$ gilt $a_0 = 1$, $a_1 = 8$ und $a_2 = 2$ und damit

$$p_0 = 1,\; q_0 = 1,\; p_1 = 9,\; q_1 = 8,\; p_2 = 19,\; q_2 = 17,\; x = \tfrac{p_2}{q_2} = \tfrac{19}{17}$$

Zusammenhang zwischen N und L: Für $x > 1$ ist $p_N > q_N$ und $a_0 \geq 1$. Somit bestimmt p_N die Länge L in der L-Bit-Arithmetik-Darstellung der beiden

Bruchzahlen [siehe Gl.(21) in Kapitel 3.1]. Aus Gl.(51) läßt sich leicht zeigen, daß wegen $p_n > p_{n-1}$

$$p_n = a_n p_{n-1} + p_{n-2} > 2p_{n-2}$$

gilt. Das bedeutet, daß

$$p_0 = a_0 \geq 1,\ p_2 > 2,\ p_4 > 4,\ p_6 > 8,\ p_{2k} > 2^k$$

und damit

$$2^{\lfloor \frac{N}{2} \rfloor} < p_N$$

In Analogie zu Kapitel 3.1 läßt sich p_N durch L Bits darstellen

$$2^{L-1} \leq p_N \leq 2^L - 1 < 2^L$$

und damit stellt sich eine direkte Beziehung zwischen N und L ein:

$$\lfloor \frac{N}{2} \rfloor < L \tag{52}$$

Für $x < 1$ ist $p_N < q_N$ und $a_0 = 0$. Hierbei hängt die Länge L von q_N ab und $q_0 = 1$, aber die Beziehung zwischen N und L läßt sich in gleicher Weise darstellen wie bei p_N.

Computational Resources: Bei der Kettenbruch-Vorwärtszerlegung werden N „Divisionen mit Rest“ benötigt. Mit Gl.(52) sind dies $O(L)$ Operationen. Auf einem klassischen Rechner verbraucht gemäß [KN] und [NI] jede „Division mit Rest“ $O(L^2)$ Operationen (siehe Kapitel 3.5). Damit ergibt sich für die Kettenbruch-Vorwärtszerlegung eine Gesamtzahl von $O(L^3)$ Operationen.
Bei der Kettenbruch-Rückwärtsberechnung werden N Iterationen, also gemäß Gl.(52) L Berechnungsschritte benötigt, in denen jeweils eine Multiplikation und eine Addition verlangt werden [siehe Gl.(51)]. Gemäß [KN] und [NI] verbraucht jeder dieser Iterationsschritte $O(L^2)$ Operationen. Damit ergibt sich auch für die Kettenbruch-Rückwärtsberechnung eine Gesamtzahl von $O(L^3)$ Operationen.

3.8 Ausblendeigenschaft von Exponentialsummen

Die Ausblendeigenschaft von Exponentialsummen ist ein zentraler Bestandteil des Shor-Verfahrens. Hierbei seien k, l, n und N natürliche Zahlen.

$$x \;=\; \frac{1}{N}\sum_{k=0}^{N-1}\exp(i2\pi\frac{kl}{N}) \;=\; \begin{cases} 1 & \text{für } l=0 \text{ oder } l=\pm nN \\ 0 & \text{für alle anderen } l \end{cases} \tag{53}$$

Um Gl.(53) zu beweisen, wird sie mit einem Exponentialsummanden multipliziert.

$$\begin{aligned} x\cdot\exp(i2\pi\tfrac{\nu l}{N}) &= \tfrac{1}{N}\sum_{k=0}^{N-1}\exp(i2\pi\tfrac{(k+\nu)l}{N}) \\ &= \tfrac{1}{N}\sum_{k=\nu}^{N+\nu-1}\exp(i2\pi\tfrac{kl}{N}) \\ &= \tfrac{1}{N}\sum_{k=\nu}^{N-1}\exp(i2\pi\tfrac{kl}{N}) + \tfrac{1}{N}\sum_{k=N}^{N+\nu-1}\exp(i2\pi\tfrac{kl}{N}) \\ &= \tfrac{1}{N}\sum_{k=\nu}^{N-1}\exp(i2\pi\tfrac{kl}{N}) + \tfrac{1}{N}\sum_{k=0}^{\nu-1}\exp(i2\pi\tfrac{kl}{N}) \quad = \; x \end{aligned}$$

Daraus folgt

$$x^2 = x\cdot\tfrac{1}{N}\sum_{\nu=0}^{N-1}\exp(i2\pi\tfrac{\nu l}{N}) = \tfrac{1}{N}\sum_{\nu=0}^{N-1} x\cdot\exp(i2\pi\tfrac{\nu l}{N}) = \tfrac{1}{N}\sum_{\nu=0}^{N-1} x \;=\; x$$

Mit

$$x^2 - x = x\cdot(x-1) = 0$$

ist

$$x = 0 \;\vee\; x = 1$$

Aber x kann nur 1 werden, wenn $\exp(i2\pi\frac{kl}{N}) = 1$ ist. Dies ist für variables k nur bei $l=0$ oder $l=\pm nN$ der Fall. Damit ist Gl.(53) bewiesen (q.e.d.).

4 Das Shor-Verfahren

Aus Sicht der Computational Resources benötigt der Algorithmus von G. L. Miller auf einem klassischen Rechner mit $O(L^2 2^L)$ Operationen einen wesentlich höheren Rechenaufwand als herkömmliche Faktorzerlegungsalgorithmen. Setzt man diesen Algorithmus allerdings auf einem Quanten-Rechner um, dann reduziert sich der Aufwand erheblich. Diese Umsetzung ist zum ersten Mal von P. Shor durchgeführt worden und wird im folgenden dargestellt.

4.1 Das Vierphasenmodell des Shor-Verfahrens

Ziel des Shor-Verfahrens ist die Ermittlung der Ordnung r von a modulo n gemäß Gl.(5). Die weiteren Schritte zur Zerlegung der Zahl n in Faktoren, die im Algorithmus von G. L. Miller beschrieben werden, sind nicht Gegenstand des Shor-Verfahrens. Damit sind a und n die Eingangsparameter und r ist der Ausgangsparameter, wie man anhand von Abbildung 6 erkennen kann.

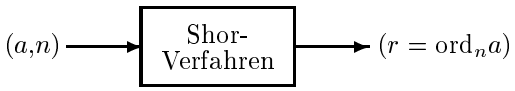

Abbildung 6: Ein- und Ausgangsverhalten des Shor-Verfahrens

Für die Darstellung des Quanten-Algorithmus werden zwei Register benötigt:

- $\mathcal{H}_1$: Der „Argument-Hilbertraum“ mit $\dim(\mathcal{H}_1) = q = 2^m$. Für die Bit-Dimension m hat P. Shor folgende Vorgaben getroffen (Siehe [SH]):

  $$n^2 \leq q = 2^m < 2n^2 \tag{54}$$

  Hieraus errechnet sich m gemäß Gl.(22) in Kapitel 3.1.

- $\mathcal{H}_2$: Der „Werte-Hilbertraum“ mit $\dim(\mathcal{H}_2) = 2^L$. Für die Bit-Dimension L gilt bekanntlich:

  $$2^{L-1} \;\leq\; n \;\leq\; 2^L - 1 \tag{55}$$

  Hieraus errechnet sich L gemäß Gl.(21) in Kapitel 3.1.

Mit Hilfe der beiden Register $\mathcal{H}_1$ und $\mathcal{H}_2$ kann das Shor-Verfahren grob in vier Phasen oder Funktionsblöcke eingeteilt werden. Diese vier Blöcke sind in Abbildung 7 dargestellt und haben von links nach rechts folgende Aufgaben:

- $\underline{\underline{M}}_H$: Das Hadamard-Gate mit Bit-Dimension m in $\mathcal{H}_1$ erzeugt für einen Basis-Eingangsvektor eine gleichverteilte Superposition aller Basis-Vektoren in $\mathcal{H}_1$.

- $\underline{\underline{M}}_f$: Das Controlled-U-Gate bildet zu jedem Basis-Vektor $|k>$ in $\mathcal{H}_1$ den zugehörigen Modulo-Funktionswert $f(k)$ gemäß Gl.(4) und damit den Basisvektor $|f(k)>$ in $\mathcal{H}_2$.

- $\underline{\underline{M}}_{IFT}$: Die inverse Fourier-Transformation (IFT) der Bit-Dimension m in $\mathcal{H}_1$ erzeugt aus der diskreten Zuordnung von Basis-Vektor $|k>$ in $\mathcal{H}_1$ und dem zugehörigem Funktionswert-Vektor $|f(k)>$ in $\mathcal{H}_2$ eine Spektral-Transformation. Mit Spektral-Transformationen kann man u.a. Periodizitäten erkennen. In diesem Fall ist die Periode die noch unbekannte Ordnung r.

- $\underline{\underline{M}}_{KBZ}$: Die Kettenbruch-Zerlegung (KBZ) erzeugt durch Auswertung von Argument-Wahrscheinlichkeiten die Periodendauer und damit die gesuchte Ordnung $r = \mathrm{ord}_n a$.

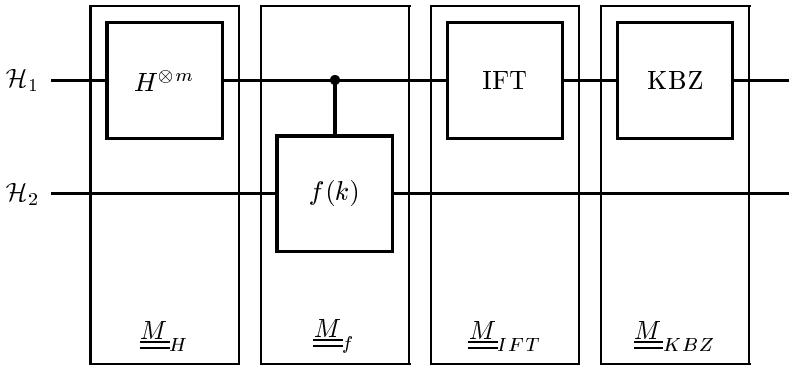

Abbildung 7: Vier-Phasen-Modell des Shor-Verfahrens

4.2 Das Shor-Verfahren als physikalischer Algorithmus

Der physikalische Algorithmus stellt das Shor-Verfahren mit Basisvektoren in Dirac-Notation dar.

$$|x>|y> \tag{56}$$

Hierbei repräsentiert $|x>$ den „Argument-Hilbertraum“ $\mathcal{H}_1$ mit der Dimension $\dim(\mathcal{H}_1) = q = 2^m$ und $|y>$ den „Werte-Hilbertraum“ $\mathcal{H}_2$ mit der Dimension $\dim(\mathcal{H}_2) = 2^L$.

4.2.1 Das Vierphasenmodell in physikalischer Darstellung

Phase 0: Der Startvektor

Im Gegensatz zu den Darstellungen in [CR] und [NI] wird der Startvektor nicht mit $|v_0> = |0>|0>$ initialisiert, sondern erhält den Wert

$$|v_0> = |0>|1> \tag{57}$$

Begründung: Im Werte-Hilbertraum macht gemäß Gl.(4) der Algorithmus

$$f(v+1) = [a \cdot f(v)] (\mathrm{mod}\ n)$$

nur dann Sinn, wenn $f(0) = 1$ ist und nicht $f(0) = 0$.

Phase 1: Das Hadamard-Gate

Das Hadamard-Gates mit Bit-Dimension m im Argument-Hilbertraum $\mathcal{H}_1$ erzeugt aus dem Basis-Vektor von Startvektor $|v_0>$ eine gleichverteilte Superposition aller Basis-Vektoren im Argument-Hilbertraum $\mathcal{H}_1$.

$$|v_1> = \frac{1}{\sqrt{q}} \sum_{k=0}^{q-1} |k>|1> \tag{58}$$

Phase 2: Das Controlled-U-Gate

Das Controlled-U-Gate ordnet jedem k im Argument-Hilbertraum einen Funktionswert der Modulo-Exponentialfunktion $f(k) = a^k (\mathrm{mod}\ n)$ im Werte-Hilbertraum zu.

$$|v_2> = \frac{1}{\sqrt{q}} \sum_{k=0}^{q-1} |k>|f(k)> = \frac{1}{\sqrt{q}} \sum_{k=0}^{q-1} |k>|a^k \ (\mathrm{mod}\ n)> \tag{59}$$

Phase 3: Die inverse Fourier-Transformation

Die inverse Fourier-Transformation ereignet sich im Argument-Hilbertraum $\mathcal{H}_1$. Die Taktfrequenz dieser Transformation hängt von q ab.

$$|k> \xrightarrow{\mathrm{IFT}} \frac{1}{\sqrt{q}} \sum_{c=0}^{q-1} |c> \exp(i 2\pi \frac{ck}{q})$$

Unter Einbeziehen des Werte-Hilbertraum ergibt der Basisvektor:

$$|k>|f(k)> \xrightarrow{\text{IFT}} \frac{1}{\sqrt{q}} \sum_{c=0}^{q-1} |c>|f(k)> \exp(i2\pi \frac{ck}{q}) \tag{60}$$

Mit Gl.(60) erfolgt die Transformation von $|v_2>$ nach $|v_3>$:

$$|v_3> = \frac{1}{q} \sum_{c=0}^{q-1} \sum_{k=0}^{q-1} |c>|f(k)> \exp(i2\pi \frac{ck}{q}) \tag{61}$$

Phase 4: Ergebnis mit oder ohne Kettenbruch-Zerlegung

Aufgrund der Periodizität von $f(k)$ über der noch unbekannten Ordnung r ist es möglich, Gl.(61) in folgender Weise zu vereinfachen.

$$|v_3> = \sum_{c=0}^{q-1} \sum_{k=0}^{r-1} \lambda_{ck} \cdot |c>|f(k)> \tag{62}$$

Dabei ist r an dieser Stelle, wie bereits geschildert, noch unbekannt. Wie diese Vereinfachung durchgeführt werden kann, wird in den folgenden Kapiteln erläutert. Aus Gl.(62) läßt sich für jeden Basis-Vektor $|c>|f(k)>$ folgende Basis-Wahrscheinlichkeit ableiten.

$$P(|c>|f(k)>) = |\lambda_{ck}|^2 \tag{63}$$

Die Basis-Wahrscheinlichkeiten werden in folgender Weise zu Argument-Wahrscheinlichkeiten zusammengefaßt:

$$P(|c>) = \sum_{k=0}^{r-1} P(|c>|f(k)>) \tag{64}$$

Mit der wichtigen Forderung von Gl.(54) ergibt sich aus der Menge aller Argument-Wahrscheinlichkeiten ein Grenzbereich

$$\frac{4}{\pi^2} P(|0>) \;<\; P(|c>) \;\leq\; P(|0>) \tag{65}$$

Aus diesem Grenzbereich wird ein c-Wert herausgesucht, der in eine der beiden gebrochen-rationalen Gleichungen

$$\frac{c}{q} \;=\; \frac{d}{r} \quad \text{oder} \quad |\frac{c}{q} - \frac{d}{r}| \;\leq\; \frac{1}{2q} \tag{66}$$

eingesetzt wird. Hierdurch wird die Ordnung r berechnet.

Mit diesen vier Phasen wurde das Shor-Verfahren zunächst grob beschrieben. Eine ausführliche Untersuchung, zeigt, daß das Verfahren in zwei Kategorien eingeteilt werden kann.

- Im Idealfall sind die Berechnung der Argument-Wahrscheinlichkeiten und die Auswertung der linken Seite von Gl.(66) sehr einfach.

- Im Realfall gestaltet sich die Berechnung der Argument-Wahrscheinlichkeiten umfangreicher, so daß bei der Auswertung der rechten Seite von Gl.(66) nicht auf die Kettenbruch-Zerlegung verzichtet werden kann.

In dieser Arbeit soll auch gezeigt werden, daß es neben der gebrochen-rationalen Gl.(66) auch noch andere Lösungswege zur Ermittlung der Ordnung r gibt, die für die Bewertung des stochastischen Algorithmus (siehe Kapitel 4.5) von Bedeutung sind.

In den folgenden Kapiteln werden Idealfall und Realfall im Zusammenhang mit Gl.(65) und Gl.(66) näher beschrieben. Eine ausführliche Erläuterung der Rolle der Kettenbruch-Zerlegung im Shor-Verfahren erfolgt in Kapitel 4.2.5.

4.2.2 Der Idealfall

Im Idealfall teilt die noch unbekannte Ordnung r den Dimensionswert q und ist folglich eine Potenz von 2. D.h. $q = Xr$. Dann gilt:

$$|v_0> = |0>|1>$$

$$|v_1> = \tfrac{1}{\sqrt{q}} \sum_{k=0}^{q-1} |k>|1> = \tfrac{1}{\sqrt{q}} \sum_{x=0}^{X-1} \sum_{y=0}^{r-1} |xr+y>|1>$$

$$k = xr + y: \; f(xr+y) = a^{xr+y} (\mathrm{mod}\ n) = a^y \ (\mathrm{mod}\ n) = f(y)$$

$$\begin{aligned} |v_2> &= \tfrac{1}{\sqrt{q}} \sum_{x=0}^{X-1} \sum_{y=0}^{r-1} |xr+y>|f(xr+y)> \\ &= \tfrac{1}{\sqrt{q}} \sum_{x=0}^{X-1} \sum_{y=0}^{r-1} |xr+y>|f(y)> \end{aligned}$$

$$|k>|f(y)> \xrightarrow{\mathrm{IFT}} \tfrac{1}{\sqrt{q}} \sum_{c=0}^{q-1} |c>|f(y)> \exp(i2\pi\tfrac{ck}{q})$$

$$\begin{aligned} |v_3> &= \tfrac{1}{q} \sum_{x=0}^{X-1} \sum_{y=0}^{r-1} \sum_{c=0}^{q-1} |c>|f(y)> \exp(i2\pi\tfrac{c(xr+y)}{q}) \\ &= \tfrac{1}{q} \sum_{c=0}^{q-1} |c> \sum_{y=0}^{r-1} |f(y)> \exp(i2\pi\tfrac{cy}{q}) \sum_{x=0}^{X-1} \exp(i2\pi\tfrac{cr}{q}x) \end{aligned} \tag{67}$$

Einsetzen von $q = Xr$ in $|v_3>$ ergibt

$$|v_3> = \tfrac{1}{Xr} \sum_{c=0}^{Xr-1} |c> \sum_{y=0}^{r-1} |f(y)> \exp(i2\pi\tfrac{cy}{Xr}) \sum_{x=0}^{X-1} \exp(i2\pi\tfrac{c}{X}x)$$

Aufgrund der Ausblendeigenschaft von Exponentialsummen (siehe Kapitel 3.8) verschwinden alle Summenterme über x mit Ausnahme derjenigen, für die $c = dX = \frac{dq}{r}$ mit $d = 0, 1, \ldots$ ist. Hierfür kann man auch schreiben:

$$\frac{c}{q} = \frac{d}{r} \tag{68}$$

Das ist der linke Teil von Gl.(66). Damit ergibt sich:

$$\begin{aligned} |v_3> &= \frac{X}{Xr} \sum_{d=0}^{dX \le Xr-1} |dX> \sum_{y=0}^{r-1} |f(y)> \exp(i2\pi\frac{yd}{r}) \\ &= \frac{1}{r} \sum_{d=0}^{r-1} |dX> \sum_{y=0}^{r-1} |f(y)> \exp(i2\pi\frac{yd}{r}) \end{aligned}$$

Aus $|v_3>$ lassen sich nun folgende Basis-Wahrscheinlichkeiten ableiten:

$$P\{|dX > f(y)>\} \;=\; |\tfrac{1}{r}\exp(i2\pi\tfrac{yd}{r})|^2 \;=\; \tfrac{1}{r^2}$$

Für $c = dX = \frac{dq}{r}$ ergeben sich über den Laufindex $d = 0, 1 \ldots r-1$ insgesamt r sogenannte RELEVANTE Argument-Wahrscheinlichkeiten:

$$P\{|c>\} \;=\; \sum_{y=0}^{r-1} P\{|c> |f(y)>\} \;=\; \sum_{y=0}^{r-1} \tfrac{1}{r^2} \;=\; \tfrac{1}{r}$$

Diese r Argument-Wahrscheinlichkeit existieren somit nur bei $c = \frac{dq}{r}$. Sie sind außerdem maximal und untereinander gleichverteilt.

$$P_{max} \;=\; \frac{1}{r} \tag{69}$$

Der Idealfall läßt sich damit folgendermaßen zusammenfasssen:

1. Für alle c, für die Gl.(68) gilt, ist $P\{|c>\} = \frac{1}{r}$.
2. Für alle c, für die Gl.(68) nicht gilt, ist $P\{|c>\} = 0$.
3. Insgesamt existieren r von Null verschiedene RELEVANTE Argument-Wahrscheinlichkeiten.
4. Die Ordnung r teilt q und ist folglich eine Potenz der Zahl 2.

4.2.3 Der Realfall: Die Regel von Shor

In den meisten Fällen ist q nicht durch r teilbar. Dann existiert eine positive ganze Zahl X und eine ganze Zahl Y mit $0 < Y < r$, so daß $q = Xr + Y$ ist. Es gilt:

$$|v_0> = \; |0> |1>$$

$$\begin{aligned} |v_1> &= \frac{1}{\sqrt{q}} \sum_{k=0}^{q-1} |k> |1> \\ &= \frac{1}{\sqrt{q}} \sum_{x=0}^{X-1} \sum_{y=0}^{r-1} |xr+y> |1> \;+\; \frac{1}{\sqrt{q}} \sum_{x=X}^{X} \sum_{y=0}^{Y-1} |xr+y> |1> \\ &= \frac{1}{\sqrt{q}} \sum_{x=0}^{X} \sum_{y=0}^{Y-1} |xr+y> |1> \;+\; \frac{1}{\sqrt{q}} \sum_{x=0}^{X-1} \sum_{y=Y}^{r-1} |xr+y> |1> \end{aligned}$$

$$k = xr + y: \;\; f(xr+y) = a^{xr+y} \pmod n = a^y \pmod n = f(y)$$

$$
\begin{aligned}
|v_2> = \;& \tfrac{1}{\sqrt{q}} \sum_{k=0}^{q-1} |k> |f(k)> \\
= \;& \tfrac{1}{\sqrt{q}} \sum_{x=0}^{X} \sum_{y=0}^{Y-1} |xr+y> |f(y)> \\
& + \tfrac{1}{\sqrt{q}} \sum_{x=0}^{X-1} \sum_{y=Y}^{r-1} |xr+y> |f(y)>
\end{aligned}
$$

$$
|k> |f(y)> \xrightarrow{\text{IFT}} \tfrac{1}{\sqrt{q}} \sum_{c=0}^{q-1} |c> |f(y)> \exp(i2\pi \tfrac{ck}{q})
$$

$$
\begin{aligned}
|v_3> = \;& \tfrac{1}{q} \sum_{c=0}^{q-1} |c> \sum_{y=0}^{Y-1} |f(y)> \exp(i2\pi \tfrac{cy}{q}) \sum_{x=0}^{X} \exp(i2\pi \tfrac{cr}{q} x) \\
& + \tfrac{1}{q} \sum_{c=0}^{q-1} |c> \sum_{y=Y}^{r-1} |f(y)> \exp(i2\pi \tfrac{cy}{q}) \sum_{x=0}^{X-1} \exp(i2\pi \tfrac{cr}{q} x)
\end{aligned}
\tag{70}
$$

Herleitung von P_{max} für den Realfall

Aufgrund der Ausblendeigenschaft von Exponentialfunktionen (siehe Kapitel 3.8) liegt die maximale Argument-Wahrscheinlichkeit in Gl.(70) auf jeden Fall bei $c = 0$ und $c = \frac{q}{2}$. Aus Gl.(70) erhält man für $c = 0$:

$$
P(|0> |f(y)>) \;=\; \begin{cases} \frac{1}{q^2}(X+1)^2 & \text{für} \quad 0 \le y \le Y-1 \\ \frac{1}{q^2} X^2 & \text{für} \quad Y \le y \le r-1 \end{cases}
$$

$$
\begin{aligned}
P_{max} \;&=\; P(|0>) \;=\; \sum_{y=0}^{r-1} P\{|0> |f(y)>\} \\
&= \tfrac{1}{q^2}[Y(X+1)^2 + (r-Y)X^2] \;=\; \tfrac{(2X+1)Y + rX^2}{(Y+rX)^2} \;=\; g(Y)
\end{aligned}
$$

Unterzieht man dieser von Y abhängigen Funktion eine Kurvendiskussion im ganzzahligen Interval $[0, r]$, dann bewegen sich die Funktionswerte im Bereich von

$$
g(Y=0) = g(Y=r) = \tfrac{1}{r} \;\le\; P_{max} \le\; g(Y=\tfrac{r}{2}) = \tfrac{1}{r}[1 + \tfrac{1}{(2X+1)^2}]
$$

Für das Beispiel $n = 21$ und $a = 2$ gilt $q = 512$, $r = 6$ und $X = 85$. Man erkennt, daß schon für kleine n der Wert X derart hoch ist, daß er in der Gleichung vernachläßigt werden kann. Daraus folgt:

$$
P_{max} \;=\; P(|0>) \;=\; P(|\frac{q}{2}>) \;\approx\; \frac{1}{r} \tag{71}
$$

Herleitung von P_{min} für den Realfall

Im Idealfall werden in Gl.(67) die Summenterme über x maximal, wenn Gl.(68) gilt. D.h. c ist eine ganzzahliges Vielfaches von $\frac{q}{r}$ und es gilt:

$$
c \;=\; \frac{dq}{r}
$$

Im Realfall ist q nicht durch r teilbar. Deshalb muß für c eine ganzzahlige Annäherung getroffen werden, so daß die Summenterme über x in Gl.(70) weiterhin maximal gewählt werden. Mit

$$\frac{dq}{r} - \frac{1}{2} \;\leq\; c \;\leq\; \frac{dq}{r} + \frac{1}{2} \tag{72}$$

über den Laufparameter $d = 0, 1, \ldots r-1$ ergeben sich insgesamt r verschiedene auf- bzw. abgerundete c-Werte. Durch Umformung erhält man die rechte Seite von Gl.(66):

$$|\frac{c}{q} - \frac{d}{r}| \;\leq\; \frac{1}{2q} \tag{73}$$

Diese sogenannte **Regel von Shor** besagt, daß die Summenterme über x in Gl.(70) maximal werden, wenn rc „fast durch q teilbar " ist, d.h. wenn

$$d - \tfrac{r}{2q} \;\leq\; \tfrac{rc}{q} \;\leq\; d + \tfrac{r}{2q}$$

Der Grenzbereich dieses Intervalls wird durch

$$\frac{rc}{q} = d \pm \frac{r}{2q} \tag{74}$$

beschrieben. Wenn man in Gl(70) die beiden Laufparameter X und $X-1$ durch einen Wert Z ersetzt, dann ergibt sich für die einzelnen Basis-Wahrscheinlichkeiten im Grenzbereich gemäß Gl.(74):

$$\begin{aligned} P\{|c> |f(y)>\} \;&=\; \tfrac{1}{q^2}|\sum_{x=0}^{Z} \exp(i2\pi\tfrac{rc}{q}x)|^2 = \tfrac{1}{q^2}|\sum_{x=0}^{Z} \exp[i2\pi(d \pm \tfrac{r}{2q})x]|^2 \\ &=\; \tfrac{1}{q^2}|\sum_{x=0}^{Z} \exp(\pm i\pi\tfrac{rx}{q})|^2 \;=\; \tfrac{1}{q^2}|\sum_{x=0}^{Z} [\exp(\pm i\pi\tfrac{r}{q})]^x|^2 \end{aligned}$$

Mit Hilfe der Potenzreihenentwicklung über x ergibt sich

$$P\{|c> |f(y)>\} \;= \frac{1}{q^2} \;\Big|\frac{1-\exp[\pm i\pi\frac{r}{q}(Z+1)]}{1-\exp(\pm i\pi\frac{r}{q})}\Big|^2$$

Wegen $r \leq \varphi(n)$ und der wichtigen Forderung aus Gl.(54) gilt:

$$r \leq \varphi(n) < n << n^2 \leq q \;\Longrightarrow\; r << q \tag{75}$$

Damit gilt wegen $q = Xr + Y$ auch $Y < r << q$ und $r - Y << q$. Somit kann folgende Abschätzung gemacht werden:

- Ist $Z = X$, dann gilt: $\frac{r}{q}(Z+1) = \frac{Xr+r}{q} = \frac{q+(r-Y)}{q} = 1 + \frac{r-Y}{q} \approx 1$.
- Ist $Z = X - 1$, dann gilt: $\frac{r}{q}(Z+1) = \frac{Xr}{q} = \frac{q-Y}{q} = 1 - \frac{Y}{q} \approx 1$.

Mit dieser Abschätzung gilt allgemein:

$$\begin{aligned} P\{|c> |f(y)>\} &\approx \frac{1}{q^2} \left|\frac{1-\exp(\pm i\pi)}{1-\exp(\pm i\pi\frac{r}{q})}\right|^2 \\ &= \frac{1}{q^2} \frac{1-\cos(\pi)}{1-\cos(\pi\frac{r}{q})} = \frac{1}{q^2} \frac{1}{\sin^2(\frac{\pi}{2}\frac{r}{q})} \end{aligned}$$

Für kleine x gilt $\sin(x) \approx x$:

$$P\{|c> |f(y)>\} \approx \frac{1}{q^2} \frac{1}{(\frac{\pi}{2}\frac{r}{q})^2} = \frac{4}{\pi^2 r^2}$$

$$P\{|c>\} = \sum_{y=0}^{r-1} P\{|c> |f(y)>\} \approx r \cdot \frac{4}{\pi^2 r^2} = \frac{4}{\pi^2 r} > \frac{1}{3r}$$

Somit gilt für den Realfall als unterster Grenzwert der Argument-Wahrscheinlichkeit, für den die Regel von Shor, Gl.(73), noch gilt:

$$P_{min} \approx \frac{4}{\pi^2 r} > \frac{1}{3r} \tag{76}$$

Die Bedeutung von P_{min} und P_{max} für den Realfall

Aus den Berechnungen für P_{min} und P_{max} kann gefolgert werden, daß

$$|\frac{c}{q} - \frac{d}{r}| \leq \frac{1}{2q} \implies \frac{1}{3r} < \frac{4}{\pi^2 r} \approx P_{min} < P(|c>) \leq P_{max} \approx \frac{1}{r} \tag{77}$$

Da aufgrund der Ausblendeigenschaft von Exponentialsummen (siehe Kapitel 3.8) P_{max} immer bei $c = 0$ anzutreffen ist, kann der rechte Teil von Gl.(77) folgendermaßen formuliert werden:

$$\frac{1}{3}P(|0>) < \frac{4}{\pi^2}P(|0>) < P(|c>) \leq P(|0>) \tag{78}$$

Über Gl.(72) gibt es somit insgesamt r verschiedene auf- bzw. abgerundete c-Werte, für die $P(|c>)$ innerhalb der Grenzen von P_{min} und P_{max} liegt. Das ist eine wichtige Eigenschaft, die auch für den stochastischen Shor-Algorithmus (siehe Kapitel 4.5) von Bedeutung ist.

4.2.4 Berechnung der Ordnung r

Aus Idealfall und Realfall des Shor-Verfahrens lassen sich allgemeine Regeln zur Bestimmung der Ordnung r ableiten. Hierzu gibt es drei verschiedene Alternativen:

1. Durch $P(|0>) \approx \frac{1}{r}$ läßt sich r einfach ermitteln.
2. Man ermittelt alle $P(|c>)$, die Gl.(78) erfüllen. Gl.(78) gilt auch für den Idealfall. Die gesamte Anzahl dieser Argument-Wahrscheinlichkeiten ergibt die Ordnung r. Die zugehörigen c-Werte werden im folgenden **RELEVANT** genannt, im Gegensatz zu allen anderen c-Werten, die Gl.(78) nicht erfüllen und somit **NICHT-RELEVANT** sind.

3. Aus der Menge der relevanten c-Werte, die Gl.(78) erfüllen, wird der kleinste von Null verschiedene c-Wert ausgewählt und in Gl.(68) bzw. Gl.(73) eingesetzt. Über das Kettenbruch-Verfahren wird $d = 1$ und die Ordnung r ermittelt.

Für den stochastischen Algorithmus des Shor-Verfahrens (siehe Kapitel 4.5) sind die Varianten 2 und 3 von Interesse. Hierbei ist zu beachten, daß Gl.(78) auch für hohe Werte von r ihre Gültigkeit hat, wenn Gl.(75) und insbesondere Gl.(54) eingehalten wird.

4.2.5 Das Kettenbruch-Verfahren für die Ordnung r

Hat man sich im Idealfall für einen geeigneten $|c>$-Wert mit $c \neq 0$ entschieden, dann kann man anhand der Gl.(68) durch Kürzen die kleinsten natürlichen Zahlen für d und r ermitteln. Damit ist die Ordnung $r = \mathrm{ord}_n a$ bestimmt.

Im Realfall ergeben sich meistens folgende Schwierigkeiten:

- Die Zahlen c und q lassen sich nicht kürzen.
- Die Zahlen c und q lassen sich zwar kürzen, aber der gekürze Bruch besteht aus zu großen Zahlen, aus denen ein Vielfaches der Ordnung r, nicht aber die Ordnung r direkt abgeleitet wird.

In diesen Fällen garantiert Gl.(73) über das Verfahren der Kettenbruch-Zerlegung einen adequaten Wert für r. Dieses Verfahren wird in Kapitel 3.7 ausführlich beschrieben. Der folgende Algorithmus verwendet Gl.(73) und ermittelt die Parameter d und r.

1. Mit Hilfe der Kettenbruch-Vorwärtszerlegung (siehe Kapitel 3.7) wird der Bruch $\frac{c}{q}$ in die Ketten-Form $[a_0, a_1, a_2, ..., a_v]$ gebracht. Da $\frac{c}{q} < 1$ ist, gilt immer $a_0 = 0$.
2. Die minimale Ketten-Form $[a_0, a_1]$ wird erzeugt und $k = 1$ gesetzt.
3. Mit der aktuellen Ketten-Form wird über die Kettenbruch-Rückwärtsberechnung (siehe Kapitel 3.7) ein neuer Bruch $\frac{d}{r}$ berechnet.
4. Genügt dieser Bruch der Gl.(73), dann sind d und r die gesuchten Parameter und der Algorihmus ist beendet.
5. Genügt dieser Bruch der Gl.(73) nicht, dann wird k um 1 erhöht und das Element a_k an das Ende der Ketten-Form hinzugefügt. Danach wird zu Punkt 3 zurückgesprungen.

Das Kettenbruch-Verfahren sollte in allen Fällen, auch in den sogenannten Idealfällen angewendet werden, um sicherzustellen, daß die Zahlen d und r hinreichend klein werden.

4.2.6 Beispiele für das Vierphasenmodell

Beispiel 1 zum Shor-Verfahren ($n = 3$, $a = 2$) stark vereinfacht

Für $n = 3$ und $a = 2$ wird das Shor-Verfahren dargestellt. Mit Gl.(54) und Gl.(55) werden die Dimensionen von Argument- und Werte-Hilbert-Raum ermittelt.

$$\begin{array}{lrcl} \mathcal{H}_1: & m & = & 1 + \lfloor 2\log_2(3) \rfloor = 4 \implies q = 2^m = 16 \\ \mathcal{H}_2: & L & = & 1 + \lfloor \log_2(3) \rfloor = 2 \end{array}$$

Für dieses Beispiel wird eine **vereinfachte** Darstellung mit $q = 4$ gewählt. Die Vektor-Dimension des Werte-Hilbertraums kann von $2^L = 4$ auf 3 reduziert werden. Das Shor-Verfahren funktioniert auch mit diesen verminderten Werten.

Damit ergeben sich folgende Gleichungen:

$$\begin{aligned} |v_0> &= |0>|1> \\ |v_1> &= \tfrac{1}{2}\sum_{k=0}^{3}|k>|1> = \tfrac{1}{2}\sum_{x=0}^{1}\{|2x>|1> + |2x+1>|1>\} \\ |v_2> &= \tfrac{1}{2}\sum_{k=0}^{3}|k>|f(k)> = \tfrac{1}{2}\sum_{x=0}^{1}\{|2x>|1> + |2x+1>|2>\} \\ |k>|f(k)> &\xrightarrow{\text{IFT}} \tfrac{1}{2}\sum_{c=0}^{3}|c>|f(k)>\exp(i2\pi\tfrac{ck}{4}) \\ |v_3> &= \tfrac{1}{4}\sum_{c=0}^{3}|c>\sum_{x=0}^{1}\{|1>\exp(i2\pi\tfrac{c2x}{4}) + |2>\exp(i2\pi\tfrac{c(2x+1)}{4})\} \end{aligned}$$

Mit der Ausblendeigenschaft von Exponentialsummen (siehe Kapitel 3.8) verschwinden alle Exponentialsummen mit Ausnahme von $c = 0$ und $c = 2$ und man erhält schließlich:

$$|v_3> = \tfrac{1}{2}|0>\{|1> + |2>\} + \tfrac{1}{2}|2>\{|1> - |2>\}$$

Daraus folgen die maximalen Basis-Wahrscheinlichkeiten:

$$P\{|0>|1>\} = P\{|0>|2>\} = P\{|2>|1>\} = P\{|2>|2>\} = \tfrac{1}{4}$$

Für alle anderen Basis-Wahrscheinlichkeiten gilt $P\{|c>|f(k)>\} = 0$. Aufsummiert ergeben die maximalen Argument-Wahrscheinlichkeiten:

$$\begin{array}{lrcl} P\{|0>\} & = & P\{|0>|1>\} + P\{|0>|2>\} & = \tfrac{1}{2} \\ P\{|2>\} & = & P\{|2>|1>\} + P\{|2>|2>\} & = \tfrac{1}{2} \end{array}$$

Die |c>-Elemente mit relevanter Argument-Wahrscheinlichkeit $P\{|c>\}$ sind somit $|0>$ und $|2>$. Damit ergeben sich gemäß Kapitel 4.2.4 drei verschiedene Wege zum Berechnen der Ordnung r.

1. $P\{|0>\} = \frac{1}{2} = \frac{1}{r} \Rightarrow r = 2$.
2. Es gibt insgesamt $r = 2$ relevante Argument-Wahrscheinlichkeiten.

3. Setzt man $c = 2$, dann erhält man $\frac{c}{q} = \frac{2}{4} = \frac{1}{2} = \frac{d}{r}$.

Für Lösungsweg 3 ist das Kettenbruch-Verfahren nicht nötig. Es führt auch über Gl.(68) zu $r = 2$.

Beispiel 2 zum Shor-Verfahren ($n = 15$, $a = 4$)

Für $n = 15$ und $a = 4$ wird das Shor-Verfahren dargestellt. Mit Gl.(55) und Gl.(54) werden die Dimensionen von Argument- und Werte-Hilbert-Raum ermittelt.

$$\begin{array}{lrcl} \mathcal{H}_1: & m & = & 1 + \lfloor 2\log_2(15) \rfloor = 8 \implies q = 2^m = 256 \\ \mathcal{H}_2: & L & = & 1 + \lfloor \log_2(15) \rfloor = 4 \end{array}$$

Damit ergeben sich folgende Gleichungen:

$$\begin{aligned} |v_0> &= |0>|1> \\ |v_1> &= \tfrac{1}{16}\sum_{k=0}^{255}|k>|1> = \tfrac{1}{16}\sum_{x=0}^{127}\{|2x>|1> + |2x+1>|1>\} \\ |v_2> &= \tfrac{1}{16}\sum_{k=0}^{255}|k>|f(k)> = \tfrac{1}{16}\sum_{x=0}^{127}\{|2x>|1> + |2x+1>|4>\} \\ |k>|f(k)> &\xrightarrow{\text{IFT}} \tfrac{1}{16}\sum_{c=0}^{255}|c>|f(k)>\exp(i2\pi\tfrac{ck}{256}) \\ |v_3> &= \tfrac{1}{256}\sum_{c=0}^{255}|c>\sum_{x=0}^{127}\{|1>\exp(i2\pi\tfrac{c2x}{256}) + |4>\exp(i2\pi\tfrac{c(2x+1)}{256})\} \end{aligned}$$

Mit der Ausblendeigenschaft von Exponentialsummen (siehe Kapitel 3.8) verschwinden alle Exponentialsummen mit Ausnahme von $c = 0$ und $c = 128$ und man erhält schließlich:

$$|v_3> = \tfrac{1}{2}|0>\{|1> + |4>\} + \tfrac{1}{2}|128>\{|1> - |4>\}$$

Daraus folgen die maximalen Basis-Wahrscheinlichkeiten:

$$P\{|0>|1>\} = P\{|0>|4>\} = P\{|128>|1>\} = P\{|128>|4>\} = \tfrac{1}{4}$$

Für alle anderen Basis-Wahrscheinlichkeiten gilt $P\{|c>|f(k)>\} = 0$. Aufsummiert ergeben die maximalen Argument-Wahrscheinlichkeiten:

$$\begin{array}{lcl} P\{|0>\} & = & P\{|0>|1>\} + P\{|0>|4>\} = \frac{1}{2} \\ P\{|128>\} & = & P\{|128>|1>\} + P\{|128>|2>\} = \frac{1}{2} \end{array}$$

Die |c>-Elemente mit relevanter Argument-Wahrscheinlichkeit $P\{|c>\}$ sind somit $|0>$ und $|128>$. Damit ergeben sich gemäß Kapitel 4.2.4 drei verschiedene Wege zum Berechnen der Ordnung r.

1. $P\{|0>\} = \frac{1}{2} = \frac{1}{r} \Rightarrow r = 2$.
2. Es gibt insgesamt $r = 2$ relevante Argument-Wahrscheinlichkeiten.

3. Setzt man $c = 128$, dann erhält man $\frac{c}{q} = \frac{128}{256} = \frac{1}{2} = \frac{d}{r}$

Für Lösungsweg 3 ist das Kettenbruch-Verfahren nicht nötig. Es führt auch über Gl.(68) zu $r = 2$.

Beispiel 3 zum Shor-Verfahren ($n = 39$, $a = 5$)

In diesem Beispiel sind $n = 39$ und $a = 5$. Mit Gl.(55) und Gl.(54) werden die Dimensionen von Argument- und Werte-Hilbert-Raum ermittelt.

$$\begin{array}{lrcl} \mathcal{H}_1: & m & = & 1 + \lfloor 2\log_2(39) \rfloor = 11 \implies q = 2^m = 2048 \\ \mathcal{H}_2: & L & = & 1 + \lfloor \log_2(39) \rfloor = 6 \end{array}$$

Damit ergeben sich folgende Gleichungen:

$$\begin{aligned} |v_0> &= |0>|1> \\ |v_1> &= \tfrac{1}{32\sqrt{2}} \sum_{k=0}^{2047} |k>|1> \\ &= \tfrac{1}{32\sqrt{2}} \sum_{x=0}^{511} \{|4x>|1> + |4x+1>|1> \\ &\qquad + |4x+2>|1> + |4x+3>|1>\} \\ |v_2> &= \tfrac{1}{32\sqrt{2}} \sum_{x=0}^{511} \{|4x>|1> + |4x+1>|5> \\ &\qquad + |4x+2>|25> + |4x+3>|8>\} \\ |k>|f(k)> &\xrightarrow{\text{IFT}} \tfrac{1}{32\sqrt{2}} \sum_{c=0}^{2047} |c>|f(k)> \exp(i2\pi\tfrac{ck}{2048}) \\ |v_3> &= \tfrac{1}{2048} \sum_{c=0}^{2047} |c> \sum_{x=0}^{511} \{|1> \exp(i2\pi\tfrac{c4x}{2048}) + |5> \exp(i2\pi\tfrac{c(4x+1)}{2048}) \\ &\qquad + |25> \exp(i2\pi\tfrac{c(4x+2)}{2048}) \\ &\qquad + |8> \exp(i2\pi\tfrac{c(4x+3)}{2048})\} \end{aligned}$$

Mit der Ausblendeigenschaft von Exponentialsummen (siehe Kapitel 3.8) verschwinden alle Exponentialsummen über c mit Ausnahme von $c = 0$, $c = 512$, $c = 1024$ und $c = 1536$ und man erhält schließlich:

$$\begin{aligned} |v_3> = & \tfrac{1}{4}|0> \{|1> + |5> + |25> + |8>\} \\ + & \tfrac{1}{4}|512> \{|1> + i|5> - |25> - i|8>\} \\ + & \tfrac{1}{4}|1024> \{|1> - |5> + |25> - |8>\} \\ + & \tfrac{1}{4}|1536> \{|1> - i|5> - |25> + i|8>\} \end{aligned}$$

Daraus folgen die maximalen Basis-Wahrscheinlichkeiten mit $c = 0$, 512, 1024, 1536 und $f(k) = 1, 5, 8, 25$:

$$P\{|c>|f(k)>\} = \tfrac{1}{16}$$

Für alle anderen Basis-Wahrscheinlichkeiten gilt $P\{|c>|f(k)>\} = 0$. Aufsummiert ergeben die maximalen Argument-Wahrscheinlichkeiten:

$$P\{|0>\} = P\{|512>\} = P\{|1024>\} = P\{|1536>\} = \tfrac{1}{4}$$

Die |c>-Elemente mit relevanter Argument-Wahrscheinlichkeit $P\{|c>\}$ sind somit $|0>$, $|512>$, $|1024>$ und $|1536>$. Damit ergeben sich gemäß Kapitel 4.2.4 drei verschiedene Wege zum Berechnen der Ordnung r.

1. $P\{|0>\} = \frac{1}{4} = \frac{1}{r} \Rightarrow r = 4$.
2. Es gibt insgesamt $r = 4$ relevante Argument-Wahrscheinlichkeiten.
3. Setzt man $c = 512$, dann erhält man $\frac{c}{q} = \frac{512}{2048} = \frac{1}{4} = \frac{d}{r}$

Für Lösungsweg 3 ist das Kettenbruch-Verfahren nicht nötig. Es führt auch über Gl.(68) zu $r = 4$.

Beispiel 4 zum Shor-Verfahren ($n = 57$, $a = 11$)

Das folgende Beispiel kann nicht auf das Kettenbruch-Verfahren zur Ermittlung der Ordnung r verzichten. Für $n = 57$ und $a = 11$ ermitteln Gl.(55) und Gl.(54) die Dimensionen von Argument- und Werte-Hilbert-Raum:

$$\begin{array}{lrcccl} \mathcal{H}_1: & m & = & 1 + \lfloor 2\log_2(57) \rfloor & = & 12 \implies q = 2^m = 4096 \\ \mathcal{H}_2: & L & = & 1 + \lfloor \log_2(57) \rfloor & = & 6 \end{array}$$

Damit ergeben sich folgende Gleichungen:

$$|v_0> = |0>|1>$$

$$|v_1> = \frac{1}{64}\sum_{k=0}^{4095}|k>|1> = \frac{1}{64}\{\sum_{k=0}^{4091}|k>|1> + \sum_{k=4092}^{4095}|k>|1>\}$$

$$\begin{aligned} |v_2> = \frac{1}{64}\sum_{x=0}^{681} & \{|6x>|1> + |6x+1>|11> \\ & + |6x+2>|7> + |6x+3>|20> \\ & + |6x+4>|49> + |6x+5>|26>\} \\ +\frac{1}{64}\sum_{y=0}^{3} & |4092+y>|f(4092+y)> \\ = \frac{1}{64}\sum_{x=0}^{682} & \{|6x>|1> + |6x+1>|11> \\ & + |6x+2>|7> + |6x+3>|20>\} \\ +\frac{1}{64}\sum_{x=0}^{681} & \{|6x+4>|49> + |6x+5>|26>\} \end{aligned}$$

$$|k>|f(k)> \xrightarrow{\text{IFT}} \frac{1}{64}\sum_{c=0}^{4095}|c>|f(k)>\exp(i2\pi\tfrac{ck}{4096})$$

$$\begin{aligned} |v_3> = \frac{1}{4096}\sum_{c=0}^{4095}|c>\{\sum_{x=0}^{682} & [\,|1>\exp(i2\pi\tfrac{c6x}{4096}) \\ & + |11>\exp(i2\pi\tfrac{c(6x+1)}{4096}) \\ & + |7>\exp(i2\pi\tfrac{c(6x+2)}{4096}) \\ & + |20>\exp(i2\pi\tfrac{c(6x+3)}{4096})\,] \\ + \sum_{x=0}^{681} & [\,|49>\exp(i2\pi\tfrac{c(6x+4)}{4096}) \\ & + |26>\exp(i2\pi\tfrac{c(6x+5)}{4096})\,]\,\} \end{aligned}$$

Die |c>-Elemente mit relevanter Argument-Wahrscheinlichkeit $P\{|c>\}$ liegen gemäß Gl.(78) bei $c = 0, 683, 1365, 2048, 2731$ und 3413. Werden gemäß Kapitel 4.2.4 die ersten zwei Wege zur Berechung von r gewählt, dann gilt:

1. $P\{|0>\} \approx \frac{1}{6} = \frac{1}{r} \;\Rightarrow\; r = 6.$
2. Es gibt insgesamt $r = 6$ relevante Argument-Wahrscheinlichkeiten.

Der dritte Lösungsweg zur Berechnung von r geht über die Kettenbruch-Zerlegung. Für $c = 683$ gilt:

$$\frac{c}{q} = \frac{683}{4096}$$

Dieser Bruch läßt sich nicht kürzen. Deshalb wird nun ein geeigneter Bruch $\frac{d}{r}$ gesucht, der Gl.(73) erfüllt.

Die Kettenbruch-Vorwärtszerlegung ergibt:

$$\frac{c}{q} \;=\; \frac{683}{4096} \;=\; [0, 5, 1, 340, 2]$$

Das Starten der Kettenbruch-Rückwärtsberechnung mit den ersten beiden Elementen der Kette ergibt:

$$\frac{d}{r} \;=\; [0, 5] \;=\; \frac{1}{5}$$

Der Vergleich mit Gl.(73) liefert:

$$\left|\frac{c}{q} - \frac{d}{r}\right| \;=\; \left|\frac{683}{4096} - \frac{1}{5}\right| \;=\; \frac{681}{5 \cdot 4096} \;>\; \frac{1}{2 \cdot 4096} \;=\; \frac{1}{2q}$$

Der verkürzte Kettenbruch erfüllt die Anforderungen von Gl.(73) nicht. Die Ketten-Form wird um ein Element erweitert:

$$\frac{d}{r} \;=\; [0, 5, 1] \;=\; \frac{1}{5+1} \;=\; \frac{1}{6}$$

Jetzt liefert der Vergleich mit Gl.(73):

$$\left|\frac{c}{q} - \frac{d}{r}\right| \;=\; \left|\frac{683}{4096} - \frac{1}{6}\right| \;=\; \frac{1}{3 \cdot 4096} \;<\; \frac{1}{2 \cdot 4096} \;=\; \frac{1}{2q}$$

Damit ist $\frac{d}{r} = \frac{1}{6}$ der gesuchte Bruch. Es ist $d = 1$ und $r = 6$ ist die gesuchte Ordnung, was auch zu erwarten war.

4.2.7 Eigenschaften des Shor-Verfahrens

(A) Die Gesamtwahrscheinlichkeit ist immer kleiner als 1.

In Beispiel 4 aus Kapitel 4.2.6 stehen am Ende 6 verschiedene c-Werte zur Auswahl, von denen nur der kleinste von Null verschiedene für das Kettenbruch-Verfahren genutzt wird. Unterzieht man jedoch alle 6 Werte der Kettenbruch-Zerlegung, dann kommt man zu folgendem Schluß:

- Der Wert $c = 0$ kann für das Kettenbruch-Verfahren nicht genutzt werden.

- Die Werte $c = 683$ und $c = 3413$ führen mittels Kettenbruch-Verfahren zu $r = 6$.
- Die Werte $c = 1365$ und $c = 2731$ führen mittels Kettenbruch-Verfahren irrtümlich zu $r = 3$.
- Der Wert $c = \frac{q}{2} = 2048$ führt mittels Kettenbruch-Verfahren irrtümlich zu $r = 2$.

Es führen nur diejenigen c zu einer korrekten Ordnung r, für die gemäß Gl.(68) d und die korrekte Ordnung r teilerfremd sind (Im o.g. Beispiel sind dies die Zahlen $c = 683$ mit $d = 1$ und $c = 3413$ mit $d = 5$). Diese Aussage läßt sich für alle Beispiele verallgemeinern.

In Gl.(2) und Gl.(3) wurden die Funktion $\mathbb{Z}_r^*$ und $\varphi(r)$ eingeführt. Hierbei ist $\mathbb{Z}_r^*$ die Menge aller positiven natürlichen Zahlen kleiner als r, die zudem auch noch teilerfremd zu r sind. Die Euler-Funktion $\varphi(r)$ beschreibt die Anzahl dieser Elemente. Diese Anzahl ist hier von Bedeutung.

Zusammenfassung:

Gl.(68) und Gl.(73) führen nur für diejenigen c zu einer korrekten Berechnung der Ordnung r, für die ein Wert d ermittelt wird, der teilerfremd zu r und kleiner als r ist. Das trifft für $\varphi(r)$ Werte von c zu. Faßt man Gl.(71) und Gl.(76) zu einen Ausdruck der Form

$$P_{min} = \frac{4}{\pi^2 r} \;\leq\; P\{|c>\} \;\leq\; P_{max} \approx \frac{1}{r} \tag{79}$$

zusammen und multipliziert diesen mit der Anzahl $\varphi(r)$ derjenigen c-Werte, die zur korrekten Berechnung der Ordnung r führen, dann ergibt sich als Gesamtwahrscheinlichkeit für die Ermittlung einer korrekten Ordnung r

$$\frac{4\varphi(r)}{\pi^2 r} \;\leq\; P_{Gesamt} \;\leq \frac{\varphi(r)}{r} \;<\; 1 \tag{80}$$

Die Wahrscheinlichkeit, mit nur einem Versuch über das Shor-Verfahren den richtigen Wert von r zu finden, ist somit immer kleiner 1 !

Beispiele:

In Beispiel 2 aus Kapitel 4.2.6 ist $\varphi(2) = 1$, damit ist $P_{Gesamt} = \frac{1}{2}$.

Für Beispiel 3 gilt $\varphi(4) = 2$, somit ist auch hier $P_{Gesamt} = \frac{1}{2}$.

(B) Die höchsten Argument-Wahrscheinlichkeiten $P\{|c>\}$ sind kein Garant für die korrekte Ermittlung der Ordnung r.

Beispiel 4 aus Kapitel 4.2.6 macht noch einen weiteren Umstand deutlich:
Die höchsten Argument-Wahrscheinlichkeiten findet man immer bei $c = 0$ und $c = \frac{q}{2}$. Allerdings kann $c = 0$ nicht zur Kettenbruch-Zerlegung herangezogen werden, und die Wahl von $c = \frac{q}{2}$ würde immer zu $r = 2$ führen. Dies ist ein wichtiger Grund, warum es Sinn macht, all diejenigen c-Werte für die Kettenbruch-Zerlegung heranzuziehen, die Gl.(77) erfüllen.

4.3 Das Shor-Verfahren als mathematisch-vektorieller Algorithmus

Der mathematisch-vektorielle Algorithmus stellt das Shor-Verfahren mit Basisvektoren in der Vektorform $\underline{e}_k$ dar. In Analogie zu Gl.(36) and Gl.(37) ergibt sich für Gl.(56) in Kapitel 4.2 folgende Äquivalenzbeziehung:

$$|x>|y> \Longrightarrow \underline{e}_{1+x+q\,y} \tag{81}$$

Hierbei repräsentiert x den „Argument-Hilbertraum" $\mathcal{H}_1$ mit der Dimension $\dim(\mathcal{H}_1) = q = 2^m$, und y den „Werte-Hilbertraum" $\mathcal{H}_2$ mit der Dimension $\dim(\mathcal{H}_2) = 2^L$, in Analogie zu Kapitel 4.2.

4.3.1 Das Vierphasenmodell in vektorieller Darstellung

Phase 0: Der Startvektor

In Analogie zu Gl.(57) aus Kapitel 4.2.1 erhält man für den Startvektor

$$\underline{v}_0 \;=\; \underline{e}_{1+q} \tag{82}$$

Für jedes nachfolgende Gate ist nur die $(q+1)$-te Spalte interessant.

Phase 1: Das Hadamard-Gate

Mit Hilfe eines Hadamard-Gates der Bit-Dimension m im Argument-Hilbertraum $\mathcal{H}_1$ ergibt sich für den Rechner-Hilbertraum $\mathcal{H}_1 \otimes \mathcal{H}_2$ ein Übertragungs-Gate der Form

$$\underline{\underline{M}}_H \;=\; \underline{\underline{H}}^{\otimes m} \otimes \underline{\underline{E}}^{\otimes L} \tag{83}$$

Hierbei ist $\underline{\underline{E}}$ die Einheitsmatrix. Sie ermöglicht die Durchleitung im Werte-Hilbertraum. In der $(q+1)$-ten Spalte dieses Übertragungs-Gate sind nur die Zeilen $q+1$ bis $2q$ von Null verschieden. Diese Elemente sind bei der Berechnung von $\underline{v}_1$ von Interesse.

Beispiel: Für $m = 2$, $q = 4$, und $L = 1$ sind in der fünften Spalte die Elemente der Zeilen 5 bis 8 von Null verschieden (siehe Fettdruck).

$$\underline{\underline{H}}^{\otimes 2} \otimes \underline{\underline{E}} = \tfrac{1}{2} \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & -1 & 1 & -1 & 0 & 0 & 0 & 0 \\ 1 & 1 & -1 & -1 & 0 & 0 & 0 & 0 \\ 1 & -1 & -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{1} & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & \mathbf{1} & -1 & 1 & -1 \\ 0 & 0 & 0 & 0 & \mathbf{1} & 1 & -1 & -1 \\ 0 & 0 & 0 & 0 & \mathbf{1} & -1 & -1 & 1 \end{pmatrix}$$

Somit ergibt sich der Vektor $\underline{v}_1$ als Produkt von $\underline{\underline{M}}_H$ und $\underline{v}_0$ zu:

$$\underline{v}_1 \;=\; \underline{\underline{M}}_H \cdot \underline{v}_0 \;=\; \frac{1}{\sqrt{q}} \sum_{k=0}^{q-1} \underline{e}_{1+k+q} \;=\; \frac{1}{\sqrt{q}} \sum_{k=q+1}^{2q} \underline{e}_k \tag{84}$$

Diese Gleichung steht in Analogie zu Gl.(58) in Kapitel 4.2.1.

Phase 2: Das Controlled-U-Gate

Abbildung 8 zeigt die Funktionsweise des Controlled-U-Gates. Im Argument-Hilbertraum definiert der Wert k diejengen Controlled-Bits, die gesetzt sind. Im Werte-Hilbertraum ist w der Eingangsparameter. In Abhängigkeit von k und w wird der Funktionswert $f(k,w)$ im Werte-Hilbertraum ermittelt.

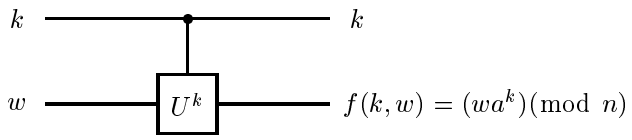

Abbildung 8: Controlled-U-Gate

Abbildung 8 ergibt für den Rechner-Hilbertraum folgende Relation:

$$|k> |w> \longrightarrow |k> |f(k,w)> = |k> |(wa^k)(\mathrm{mod}\ n)> \tag{85}$$

Bei der Übersetzung in die Vektorenschreibweise gilt mit $q = 2^m$:

$$\underline{e}_{1+k+qw} \longrightarrow \underline{e}_{1+k+qf(k,w)} \tag{86}$$

Gl.(86) erzeugt eine Matrix $\underline{\underline{M}}_f$, in der jede Spalte ein Eins-Element und sonst nur Null-Elemente enthält. Für das Shor-Verfahren gilt $w = 1$. Deshalb kann die Modulo-Exponentialfunktion $f(k,w)$ folgendermaßen vereinfacht werden:

$$f(k) = f(k, w = 1) = a^k \ (\mathrm{mod}\ n)$$

Damit ergibt sich für das Shor-Verfahren folgendes Gate-Resultat:

$$\underline{v}_2 \ = \ \underline{\underline{M}}_f \cdot \underline{v}_1 \ = \ \frac{1}{\sqrt{q}} \sum_{k=0}^{q-1} \underline{e}_{1+k+qf(k)} \tag{87}$$

Gl.(87) steht in Analogie zu Gl.(59) in Kapitel 4.2.1.

Phase 3: Die inverse Fourier-Transformation

Die inverse Fourier-Transformation findet, wie bereits geschildert, im Argument-Hilbertraum $\mathcal{H}_1$ statt. Die Taktfrequenz dieser Transformation hängt von q ab. Mit Hilfe von Gl.(60) erhält man folgende Beziehung:

$$\underline{e}_{1+k+qf(k)} \ \stackrel{\mathrm{IFT}}{\longrightarrow} \ \frac{1}{\sqrt{q}} \sum_{c=0}^{q-1} \underline{e}_{1+c+qf(k)} \exp(i2\pi \frac{ck}{q}) \tag{88}$$

Durch diese Zuweisungsgleichung wird eine Matrix $\underline{\underline{M}}_{IFT}$ definiert, auf deren Hauptdiagonale 2^L inverse Fourier-Transformationsmatrizen $\underline{\underline{M}}_{IFT0}$ der Dimension $q = 2^m$ liegen, wobei L die Bit-Dimension des Werte-Hilbertraums ist.

$$\underline{\underline{M}}_{IFT0} = (m_{ck})_{1\le c\le q, 1\le k\le q} : \ m_{ck} = \frac{1}{\sqrt{q}} \exp(i2\pi \frac{(c-1)(k-1)}{q}) \tag{89}$$

$$\underline{\underline{M}}_{IFT} = \underline{\underline{M}}_{IFT0} \otimes \underline{\underline{E}}^{\otimes L} = \begin{pmatrix} \underline{\underline{M}}_{IFT0} & \underline{\underline{0}} & \cdots & \underline{\underline{0}} \\ \underline{\underline{0}} & \underline{\underline{M}}_{IFT0} & \cdots & \underline{\underline{0}} \\ \cdots & \cdots & \ddots & \cdots \\ \underline{\underline{0}} & \underline{\underline{0}} & \cdots & \underline{\underline{M}}_{IFT0} \end{pmatrix} \tag{90}$$

Mit Hilfe von Gl.(88) oder mit Hilfe dieser Matrix wird $\underline{v}_3$ ermittelt.

$$\underline{v}_3 \;=\; \underline{\underline{M}}_{IFT} \cdot \underline{v}_2 \;=\; \frac{1}{q} \sum_{c=0}^{q-1} \sum_{k=0}^{q-1} \underline{e}_{1+c+qf(k)} \exp(i2\pi \frac{ck}{q}) \tag{91}$$

Diese Gleichung steht in Analogie zu Gl.(61) in Kapitel 4.2.1.

Da es insgesamt höchstens r verschiedene Funktionswerte für $f(k)$ gibt und q verschiedene Werte von c, ist die Anzahl der Basisvektoren $\underline{e}_{1+c+qf(k)}$ auf qr beschränkt, wobei an dieser Stelle die Ordnung r noch nicht bekannt ist.

$$\underline{v}_3 \;=\; \sum \lambda_\nu \cdot \underline{e}_\nu \;|_{\nu \in \{1+c+qf(k):\; 0 \le c \le q-1,\; 0 \le k \le q-1\}} \tag{92}$$

An dieser Stelle ist das mathematische Modell erschöpft, d.h. die mathematischen Vektoren müssen in die physikalischen Darstellung, die Dirac-Notation, zurückgerechnet werden.

$$\underline{e}_\nu = \underline{e}_{1+x+qy} \;\Longrightarrow\; |x>|y>:\; |v_3> = \sum_\nu \lambda_\nu \cdot |x>|y> \tag{93}$$

Bei der Umrechnung wird ausgehend von einem beliebigen Vektor $\underline{e}_\nu$ folgendermaßen verfahren:

$$x = (\nu - 1)(\mathrm{mod}\ q)\ ,\ \ y = (\nu - x - 1)/q \tag{94}$$

Mit Gl.(93) errechnet man die Basis-Wahrscheinlichkeit

$$P\{|x>|y>\} = P\{\underline{e}_\nu\} = |\lambda_\nu|^2 \tag{95}$$

Als Summe aller Basis-Wahrscheinlichkeiten mit gleichem $|x>$-Wert ergibt sich die Argument-Wahrscheinlichkeit zu

$$P\{|x>\} = \sum P\{|x>|y>\} \tag{96}$$

Phase 4: Ergebnis mit oder ohne Kettenbruch-Zerlegung

Das weitere Verfahren zur Ermittlung von r geht analog wie in Kapitel 4.2.1 und wird hier nicht weiter erläutert.

Fazit: Der mathematische Algorithmus des Shor-Verfahrens ist eine Analogie zum physikalischen Algorithmus. In Phase 0 wird mittels Gl.(81) von der Dirac-Notation in die mathematische Schreibweise umgeschalten. Vor der Phase 4 wird mittels Gl.(93) aus der mathematischen Schreibweise in die Dirac-Notation umgerechnet.

Es gibt Quanten-Algorithmen, bei denen das Umschalten von der Dirac-Notation in die vektorielle Schreibweise von Vorteil ist. Zum Beispiel kann das Deutsch-Josza-Verfahren in wenigen mathematischen Schritten hergeleitet werden (siehe [SC]), während sich die physikalische Darstellung wesentlich komplexer und komplizierter gestaltet (siehe [NI]).
Das Shor-Verfahren bietet in der vektoriellen Darstellung keine Vorteile gegenüber der Dirac-Notation. Allerdings ist die mathematische Darstellung gut zum Programmieren geeignet und ist als Vorstufe für den stochastischen Algoritmus notwendig.

4.3.2 Beispiele für das Vierphasenmodell

In diesem Kapitel wird auf Beispiele von Kapitel 4.2.6 zurückgegriffen.

Beispiel 1 zum Shor-Verfahren ($n = 3$, $a = 2$) stark vereinfacht

Für $n = 3$ und $a = 2$ ist bekanntlich $m = 4$, $q = 16$ und $L = 2$. Für eine maßstabsgerechte Matrizen-Darstellung werden hier, analog zu Kapitel 4.2.6, die Parameter des Argument-Hilbertraums auf $m = 2$, $q = 4$ und die Dimension des Werte-Hilbertraums von $2^L = 4$ auf 3 heruntergesetzt. Damit ist die Gesamt-Dimension dieses Beispiels 12. Die Wahl dieser verminderten Parameter funktioniert bekanntlich für dieses Beispiel.

$$\underline{v}_0 = \underline{e}_5$$

$$\underline{v}_1 = \tfrac{1}{2} \sum_{k=0}^{3} \underline{e}_{5+k} = \tfrac{1}{2} \sum_{x=0}^{1} \{\underline{e}_{5+2x} + \underline{e}_{5+(2x+1)}\}$$

ν	$f(\nu)$	$1+\nu+q$	$1+\nu+qf(\nu)$	$1+c+qf(\nu)$
$2x$	1	$5+2x$	$5+2x$	$5+c$
$2x+1$	2	$5+(2x+1)$	$9+(2x+1)$	$9+c$

$$\underline{v}_2 = \tfrac{1}{2} \sum_{x=0}^{1} \{\underline{e}_{5+2x} + \underline{e}_{9+(2x+1)}\} = \tfrac{1}{2}\{\underline{e}_5 + \underline{e}_7 + \underline{e}_{10} + \underline{e}_{12}\}$$

$$\underline{e}_{1+\nu+qf(x)} \xrightarrow{\text{IFT}} \tfrac{1}{\sqrt{q}} \sum_{c=0}^{q-1} \underline{e}_{1+c+qf(x)} \exp(i2\pi \tfrac{c\nu}{q})$$

$$\underline{v}_3 = \tfrac{1}{4} \sum_{c=0}^{3} \sum_{x=0}^{1} \{\underline{e}_{5+c} \exp(i2\pi \tfrac{c2x}{4}) + \underline{e}_{9+c} \exp(i2\pi \tfrac{c(2x+1)}{4})\}$$

Mit der Ausblendeigenschaft von Exponentialsummen (siehe Kapitel 3.8) verschwinden alle Exponentialsummen über c mit Ausnahme von $c = 0$ und $c = 2$ und man erhält schließlich:

$$\underline{v}_3 = \tfrac{1}{2} \{\underline{e}_5 + \underline{e}_7 + \underline{e}_9 - \underline{e}_{11}\}$$

Die Umwandlung in die physikalische Darstellung ergibt:

$$|v_3> = \tfrac{1}{2}\{|0>|1> + |2>|1> + |0>|2> - |2>|2>\}$$

Die |c>-Elemente mit großer Argument-Wahrscheinlichkeit $P\{|c>\}$ sind $|0>$ und $|2>$. Über die drei Lösungswege aus Kapitel 4.2.4 ergibt sich die gesuchte Ordnung $r = 2$.

Die Gesamt-Hadamard-Matrix $\underline{\underline{M}}_H$ und die Gesamt-IFT-Matrix $\underline{\underline{M}}_{IFT}$ sind 12 x 12-Matrizen, die in ihrer Hauptdiagonale durch 3 Einzel-Hadamard-Matrizen bzw. 3 Einzel-IFT-Matrizen der Dimension 4 x 4 dargestellt werden.

Die Matrix des Controlled-U-Gates $\underline{\underline{M}}_f$ errechnet sich über Gl.(86). Setzt man in dieser Gleichung $q = 4$ und $f(k, w) = (w2^k) (\mathrm{mod}\ 3)$, dann erhält man:

$$\underline{e}_{1+k+4w} \longrightarrow \underline{e}_{1+k+4f(k,w)}$$

Damit ergeben sich folgende Matrizenberechnungen:

$$\underline{v}_0^T = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$\underline{\underline{M}}_H = \frac{1}{2} \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{1} & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{1} & -1 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{1} & 1 & -1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{1} & -1 & -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & -1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & -1 & 1 \end{pmatrix}$$

$$\underline{v}_1^T = \frac{1}{2} \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$\underline{\underline{M}}_f = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$\underline{v}_2^T = \frac{1}{2} \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \end{pmatrix}$$

$$\underline{\underline{M}}_{IFT} = \frac{1}{2}\begin{pmatrix}
1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & i & -1 & -i & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & -1 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & -i & -1 & i & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \mathbf{1} & 1 & \mathbf{1} & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \mathbf{1} & i & \mathbf{-1} & -i & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \mathbf{1} & -1 & \mathbf{1} & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \mathbf{1} & -i & \mathbf{-1} & i & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & \mathbf{1} & 1 & \mathbf{1} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & \mathbf{i} & -1 & \mathbf{-i} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & \mathbf{-1} & 1 & \mathbf{-1} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & \mathbf{-i} & -1 & \mathbf{i}
\end{pmatrix}$$

$$\underline{v}_3^T = \frac{1}{2}\begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & -1 & 0 \end{pmatrix}$$

Die Gesamt-IFT-Matrix wird durch Multiplikation mit $\underline{v}_2$ nur in den Spalten 5, 7, 10 und 12 mit einem von Null verschiedenen Element multipliziert. In dieser Matrix sind nur die von Null verschiedenen Elemente fett unterlegt, die einen Beitrag zu $\underline{v}_3$ liefern. Das sind in Summe alle q^2 Elemente einer Einzel-IFT-Matrix, gleichmäßig verteilt auf r von insgesamt L Einzel-IFTs in der Diagonale der Gesamt-IFT-Matrix.

Beispiel 2 zum Shor-Verfahren ($n = 15$, $a = 4$)

Für $n = 15$ und $a = 4$ ist bekanntlich $m = 8$, $q = 256$ und $L = 4$. Damit ergibt die Gesamt-Dimension des Rechner-Hilbertraums $2^{m+L} = 4096$.

$$\underline{v}_0 = \underline{e}_{257}$$

$$\underline{v}_1 = \frac{1}{16}\sum_{k=0}^{255}\underline{e}_{257+k} = \frac{1}{16}\sum_{x=0}^{127}\{\underline{e}_{257+2x} + \underline{e}_{258+2x}\}$$

ν	$f(\nu)$	$1+\nu+q$	$1+\nu+qf(\nu)$	$1+c+qf(\nu)$
$2x$	1	$257+2x$	$257+2x$	$257+c$
$2x+1$	4	$258+2x$	$1026+2x$	$1025+c$

$$\underline{v}_2 = \frac{1}{16}\sum_{x=0}^{127}\{\underline{e}_{257+2x} + \underline{e}_{1026+2x}\}$$

$$\underline{e}_{1+\nu+qf(x)} \xrightarrow{\text{IFT}} \frac{1}{\sqrt{q}}\sum_{c=0}^{q-1}\underline{e}_{1+c+qf(x)}\exp(i2\pi\frac{\nu c}{q})$$

$$\underline{v}_3 = \frac{1}{256}\sum_{c=0}^{255}\sum_{x=0}^{127}\{\underline{e}_{257+c}\exp(i2\pi\frac{c2x}{256}) + \underline{e}_{1025+c}\exp(i2\pi\frac{c(2x+1)}{256})\}$$

Mit der Ausblendeigenschaft von Exponentialsummen (siehe Kapitel 3.8) verschwinden alle Exponentialsummen über c mit Ausnahme von $c = 0$ und $c = 128$ und man erhält schließlich:

$$\underline{v}_3 = \frac{1}{2}\{\underline{e}_{257} + \underline{e}_{385} + \underline{e}_{1025} - \underline{e}_{1153}\}$$

Die Umwandlung in die physikalische Darstellung ergibt:

$$|v_3> = \frac{1}{2}\{|0>|1> + |128>|1> + |0>|4> - |128>|4>\}$$

Die |c>-Elemente mit großer Argument-Wahrscheinlichkeit $P\{|c>\}$ sind $|0>$ und $|128>$. Über die drei Lösungswege aus Kapitel 4.2.4 ergibt sich die gesuchte Ordnung $r = 2$.

Beispiel 3 zum Shor-Verfahren ($n = 39$, $a = 5$)

Für $n = 39$ und $a = 5$ ist bekanntlich $m = 11$, $q = 2048$ und $L = 6$. Damit ergibt die Gesamt-Dimension des Rechner-Hilbertraums $2^{m+L} = 131072$.

$$\underline{v}_0 = \underline{e}_{2049}$$

$$\underline{v}_1 = \frac{1}{32\sqrt{2}} \sum_{k=0}^{2047} \underline{e}_{2049+k} = \frac{1}{32\sqrt{2}} \sum_{x=0}^{511} \{\underline{e}_{2049+4x} + \underline{e}_{2050+4x} + \underline{e}_{2051+4x} + \underline{e}_{2052+4x}\}$$

ν	$f(\nu)$	$1+\nu+q$	$1+\nu+qf(\nu)$	$1+c+qf(\nu)$
$4x$	1	$2049+4x$	$2049+4x$	$2049+c$
$4x+1$	5	$2050+4x$	$10242+4x$	$10241+c$
$4x+2$	25	$2051+4x$	$51203+4x$	$51201+c$
$4x+3$	8	$2052+4x$	$16388+4x$	$16385+c$

$$\underline{v}_2 = \frac{1}{32\sqrt{2}} \sum_{x=0}^{511} \{\underline{e}_{2049+4x} + \underline{e}_{10242+4x} + \underline{e}_{51203+4x} + \underline{e}_{16388+4x}\}$$

$$\underline{e}_{1+\nu+qf(x)} \xrightarrow{\text{IFT}} \frac{1}{\sqrt{q}} \sum_{c=0}^{q-1} \underline{e}_{1+c+qf(x)} \exp(i2\pi\frac{c\nu}{q})$$

$$\begin{aligned}\underline{v}_3 = \frac{1}{2048} \sum_{c=0}^{2047} \sum_{x=0}^{511} \{&\underline{e}_{2049+c} \exp(i2\pi\tfrac{c4x}{2048}) + \underline{e}_{10241+c} \exp(i2\pi\tfrac{c(4x+1)}{2048}) \\ &+ \underline{e}_{51201+c} \exp(i2\pi\tfrac{c(4x+2)}{2048}) + \underline{e}_{16385+c} \exp(i2\pi\tfrac{c(4x+3)}{2048})\}\end{aligned}$$

Mit der Ausblendeigenschaft von Exponentialsummen (siehe Kapitel 3.8) verschwinden alle Exponentialsummen über c mit Ausnahme von $c = 0$, $c = 512$, $c = 1024$ und $c = 1536$ und man erhält schließlich:

$$\begin{aligned}\underline{v}_3 = &\frac{1}{4}\{\underline{e}_{2049} + \underline{e}_{2561} + \underline{e}_{3073} + \underline{e}_{3585}\} \\ + &\frac{1}{4}\{\underline{e}_{10241} + i\underline{e}_{10753} - \underline{e}_{11265} - i\underline{e}_{11777}\} \\ + &\frac{1}{4}\{\underline{e}_{51201} - \underline{e}_{51713} + \underline{e}_{52225} - \underline{e}_{52737}\} \\ + &\frac{1}{4}\{\underline{e}_{16385} - i\underline{e}_{16897} - \underline{e}_{17049} + i\underline{e}_{17561}\}\end{aligned}$$

Die Umwandlung in die physikalische Darstellung ergibt:

$$\begin{aligned}|v_3> = &\frac{1}{4}\{|0>|1> + |512>|1> + |1024>|1> + |1536>|1>\} \\ + &\frac{1}{4}\{|0>|5> + i|512>|5> - |1024>|5> - i|1536>|5>\} \\ + &\frac{1}{4}\{|0>|25> - |512>|25> + |1024>|25> - |1536>|25>\} \\ + &\frac{1}{4}\{|0>|8> - i|512>|8> - |1024>|8> + i|1536>|8>\}\end{aligned}$$

Die |c>-Elemente mit großer Argument-Wahrscheinlichkeit $P\{|c>\}$ sind $|0>$, $|512>$, $|1024>$ und $|1536>$. Über die drei Lösungswege aus Kapitel 4.2.4 ergibt sich die gesuchte Ordnung $r = 4$.

4.3.3 Computational Resources des Shor-Verfahrens

Anhand der mathematisch-vektoriellen Darstellung in Kapitel 4.3.1 lassen sich die Computational Resources des Shor-Verfahrens einfach darstellen.

Klassischer Rechner

Betrachtet man die Computational Resources des Shor-Verfahrens auf einem klassischen Rechner, dann sind bei diesem Verfahren gegenüber den in Kapitel 2 vorgestellten Faktorisierungs-Algorithmen deutliche Nachteile zu erkennen.

- Phase 1 (Hadamard-Gate im Argument-Hilbertraum): Die Matrix $\underline{\underline{M}}_H$ besteht aus 2^L Hadamard-Untermatrizen der Dimension q x q, die in der Hauptdiagonale von $\underline{\underline{M}}_H$ liegen (siehe Matrix-Darstellung des Beispiels in Kapitel 4.3.2). Aus der zweiten Untermatrix von oben wird nur die linke Spalte für das Shor-Verfahren benutzt. Damit werden aus $\underline{\underline{M}}_H$ insgesamt $q = 2^m$ reellwertige Koeffizienten für Multiplikationen verwendet.

- Phase 2 (Controlled-Gate im Werte-Hilbertraum) besteht aus zwei Teilen.

  Im ersten Teil wird die Matrix $\underline{\underline{M}}_f$ des Controlled-U-Gates gemäß Gl.(85) und Gl.(86) erzeugt. Die Hauptaufgabe liegt in der Berechnung der Modulo-Exponentialfunktion $f(k, w) = (wa^k) (\mathrm{mod}\ n)$ für alle k und alle w. Durch sukzessives Berechnen von $f(k + 1, w) = af(k, w)$ ergeben sich $O(2^{m+L})$ Multiplikationen und Divisionen mit Rest und damit insgesamt $O(L^2 2^{m+L})$ ganzzahlige Operationen.

  Im zweiten Teil wird diese Matrix während der Laufzeit des Shor-Verfahrens durchlaufen. Da für das Shor-Verfahren nur $w = 1$ von Interesse ist, genügt die Berechnung nur weniger Matrixelemente von $\underline{\underline{M}}_f$ mit Hilfe der reduzierten Modulo-Exponentialfunktion $f(k) = a^k (\mathrm{mod}\ n)$. Damit kann durch sukzessives Rechnen die Anzahl der Multiplikationen und Divisionen mit Rest auf $O(2^m)$ und der Gesamtaufwand der ganzzahligen Operationen auf $O(L^2 2^m)$ reduziert werden.

- Phase 3 (Inverse Fourier-Transformation im Argument-Hilbertraum): Die Matrix $\underline{\underline{M}}_{IFT}$ besteht aus 2^L IFT-Untermatrizen der Dimension q x q, die in der Hauptdiagonale von $\underline{\underline{M}}_{IFT}$ liegen (siehe Matrix-Darstellung des Beispiels in Kapitel 4.3.2). Aus der gesamten Matrix werden insgesamt 2^{2m} reellwertige Koeffizienten für Multiplikationen und Additionen verwendet.

Insgesamt wird der Aufwand des Shor-Verfahrens durch die Phase 3 definiert. Bei den hier genannten Abschätzungen muß beachtet werden, daß für reellwertige Multiplikationen und Additionen ein weiterer nicht unwesentlicher Faktor in die Gesamtberechnung einfließt. Dadurch fallen die ganzzahligen Operationen aus Phase 2 nicht ins Gewicht.

Betrachtet man das Shor-Verfahren als den ersten Teil eines Faktorisierungs-Algorithmus, bei dem im zweiten Teil noch die Kettenbruch-Zerlegung mit $O(m^3)$ Rechen-Operationen und der ggT mit $O(L^3)$ Rechen-Operationen hinzukommen, dann muß diesem zweiten Teil keine große Bedeutung zugemessen werden.

Somit benötigt der gesamte Faktorisierungs-Algorithmus auf dem klassischen Rechner $O(2^{2m})$ Operationen.

Quanten-Rechner

Der eigentliche Vorteil des Shor-Verfahrens liegt auf dem Quanten-Rechner. Für diesen wurde es konzipiert. Die Anzahl der Gate-Operationen wird hier u.a. nach den in Kapitel 3.6 beschriebenen Regeln ermittelt:

- In Phase 1 (Hadamard-Gate im Argument-Hilbertraum) wird das Hadamard-Gate der Bit-Dimension m durch eine Hintereinanderschaltung von m Ein-Bit-Hadamard-Gates realisiert. Folglich ist die Anzahl der Gate-Operationen gleich m. Abbildung 9 zeigt ein Beispiel für $m = 3$.

- Für die Phase 2 (Controlled-Gate im Werte-Hilbertraum) wird die ausführliche Beschreibung von [NI] (Kapitel 5.3, Box 5.2) kurz zusammengefaßt. Auch hier besteht der Algorithmus zur Berechnung der Modulo-Exponentialfunktion $f(k) = a^k \pmod n$ aus zwei Teilen.

  Der erste Teil berechnet die Modulo-Exponentialfunktionen des Controlled-U-Gates. Zunächst wird $a^2 \pmod n$ durch Quadrieren von $a \pmod n$ berechnet, danach sukzessive $a^4 \pmod n$ durch Quadrieren von $a^2 \pmod n$, usw. Hierdurch ergeben sich insgesamt m Controlled-U-Gates, bei denen das eine aus dem anderen durch Quadrieren hervorgeht. Jedes Quadrieren modulo n benötigt eine Multiplikation und eine Division mit Rest. Damit betragen die Gesamtkosten dieses ersten Teils $O(m^3)$ Gate- und Rechen-Operationen. Abbildung 10 zeigt die Reihenschaltung der Controlled-U-Gates für $m = 3$.

  Der zweite Teil verschaltet diese m Gates durch Multiplikation in Abhängigkeit von den Zuständen der Controlled-Bits im Argument-Hilbertraum. Maximal $m - 1$ Multiplikationen und $m - 1$ Divisionen mit Rest werden hier durchgeführt, wodurch sich insgesamt $O(m^3)$ Rechen-Operationen ergeben.

  Sowohl der erste wie auch der zweite Teil von Phase 2 benötigen $O(m^3)$ Gate- und Rechen-Operationen. Aus diesem Grund ist es unerheblich, ob der erste Teil in die Aufwands-Berechnungen einfließt oder nicht, denn gemäß Kapitel 3.6, Absatz 2, ist die Berechnung von Funktionswerte im Werte-Hilbertraum nicht Bestandteil des Gesamtaufwands.

  In diesem Punkt hält sich [NI] nicht an die Vorgaben von [CR].

  Trotzdem kann festgehalten werden, daß Phase 2 insgesamt $O(m^3)$ Gate- und Rechen-Operationen benötigt.

- In Phase 3 (Inverse Fourier-Transformation im Argument-Hilbertraum) wird die IFT der Dimension m durch m Hadamard-Gates, $\frac{m}{2}(m-1)$ Controlled-R_k-Gates und maximal $\frac{m}{2}$ Invertier-Gates dargestellt. Die Berechnung dieser Zahlen wird in [NI], Kapitel 5.1, sehr ausführlich beschrieben. Abbildung 11 zeigt eine Gate-Realisierung für eine inversen Fourier-Transformation der Bit-Dimension $m = 3$.

Jedes Gate bedeutet eine Operation auf dem Quanten-Rechner, somit werden bis zu $(\frac{m^2}{2}+m)$ Gates, also insgesamt $O(m^2)$ Gate-Operationen durchlaufen.

Damit beträgt der Gesamtaufwand des Shor-Verfahrens auf dem Quanten-Rechner $O(m^3)$ Gate- und Rechen-Operationen.

Betrachtet man das Shor-Verfahren als den ersten Teil eines Faktorisierungs-Algorithmus, bei dem im zweiten Teil die Kettenbruch-Zerlegung mit $O(m^3)$ Gate-Operationen und der ggT mit $O(L^3)$ Rechen-Operationen hinzukommen, dann verbraucht der gesamte Faktorisierungs-Algorithmus aufgrund dieser zusätzlichen Funktionen auf dem Quanten-Rechner insgesamt $O(m^3)$ Rechen- und Gate-Operationen.

Klassischer Rechner contra Quanten-Rechner

Die folgenden Betrachtungen beziehen sich nur auf das Shor-Verfahren.

Bezüglich der Computational Resources für das Shor-Verfahren erhält man

- $O(2^{2m})$ reellwertige multiplikative bzw. additive Operationen auf dem klassischen Rechner und
- $O(m^3)$ Gate- und Rechen-Operationen auf dem Quanten-Rechner.

Natürlich lassen sich diese beiden Zahlen in dieser Form nicht miteinander vergleichen. Daß das Shor-Verfahren dem Quanten-Rechner trotzdem den Vorzug gegenüber dem klassischen Rechner gibt, ist allein darauf zurückzuführen, daß die Anzahl der Operationen auf dem Quanten-Rechner in Bezug auf den Wert m in Polynomform gebracht werden kann, während die Anzahl der Operationen auf dem klassischen Rechner exponentiell ist.

Da das Shor-Verfahren ein Quanten-Algorithmus ist, wird in den Kapiteln 4.4 und 4.5 unter Zuhilfenahme des in Kapitel 3.2 eingeführten Wahrscheinlichkeitsmaßes ein stochastischer Algorithmus untersucht, der in Abhängigkeit von m versucht, das Shor-Verfahren auf einem klassischen Rechner dem Shor-Verfahren auf einem Quanten-Rechner anzunähern. Dieser Algorithmus verfolgt das Ziel, den (reellen) exponentiellen Gesamtaufwand des Shor-Verfahren auf einem klassischen Rechner auf eine polynomiale Größe zu reduzieren.

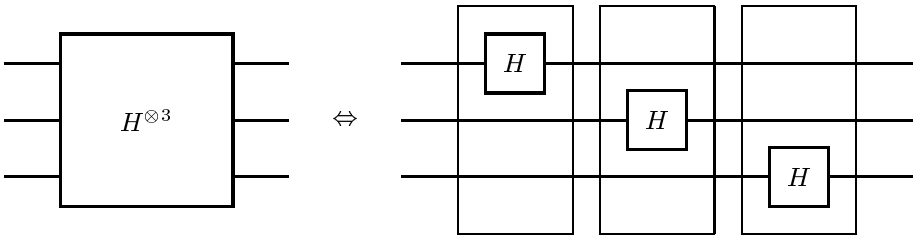

Abbildung 9: Hadamard-Gate der Dimension $m = 3$

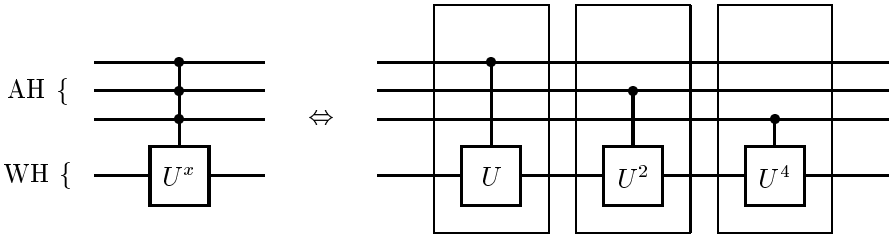

Abbildung 10: Controlled-U-Gate für einen Argument-Hilbertraum (AH) der Dimension $m = 3$. {Der Werte-Hilbertraum (WH) wurde zu einer Linie zusammengefaßt.}

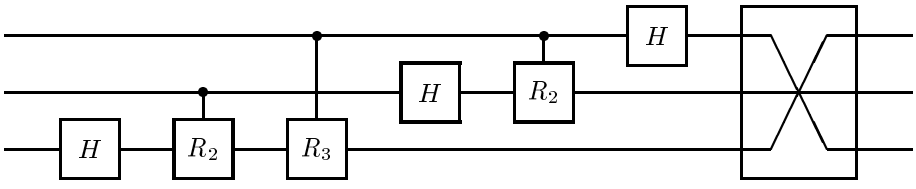

Abbildung 11: Inverse Fourier-Transformation der Dimension $m = 3$

4.4 Die Grundidee des mathematisch-stochastischen Algorithmus

Die Grundidee des mathematisch-stochastischen Algorithmus stützt sich auf Kapitel 2.4 („Messungen“) von [SC], d.h auf Messungen von Multi-q-Bits.

4.4.1 Einfache Messungen

Wie in [SC], Kapitel 2.4, und in Kapitel 3.3 dieser Arbeit geschildert wird, kann für ein beliebiges Multi-q-Bit $\underline{v} \in \mathcal{S}^1_{\mathcal{H}^{\otimes n}}$ jedem Basisvektor $\underline{e}_\mu$ über den Koeffizienten λ_μ eine Wahrscheinlichkeit P zugewiesen werden.

$$\underline{v} = \sum_{k=1}^{N} \lambda_k \ \underline{e}_k \ , \ P(\underline{e}_\mu) = |\lambda_\mu|^2 \ , \ N = 2^n$$

Eine „einfache Messung“ eines Multi-q-Bits bedeutet die willkürliche Auswahl eines Basisvektors unter Berücksichtigung der Wahrscheinlichkeit aller im Multi-q-Bit vertretenen Basisvektoren. Die willkürliche Auswahl erfolgt über den Zufallsgenerator $\mathcal{Z}$. Siehe Abbildung 12.

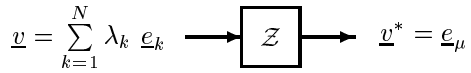

Abbildung 12: Einfache Messung

Hierbei wird nun folgendermaßen verfahren:
Über die Wahrscheinlichkeiten P werden zunächst folgende Intervallfunktionen definiert:

$$\begin{aligned} I(\underline{e}_1) &= [\ 0\ ,\ P(\underline{e}_1)\ ] \\ I(\underline{e}_2) &=]\ P(\underline{e}_1)\ ,\ P(\underline{e}_1) + P(\underline{e}_2)\ ] \\ I(\underline{e}_3) &=]\ P(\underline{e}_1) + P(\underline{e}_2)\ ,\ P(\underline{e}_1) + P(\underline{e}_2) + P(\underline{e}_3)\ ] \\ I(\underline{e}_k) &=]\ \sum_{\nu=1}^{k-1} P(\underline{e}_\nu)\ ,\ \sum_{\nu=1}^{k} P(\underline{e}_\nu)\ ] \ \text{ für } k > 1 \end{aligned} \tag{97}$$

Abbildung 13 verdeutlicht dies:

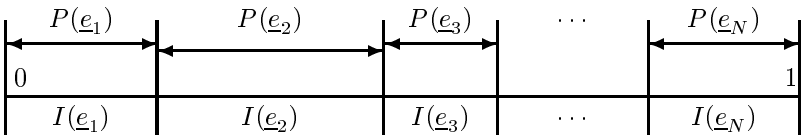

Abbildung 13: Zusammenhang zwischen $P(\underline{e}_k)$ und $I(\underline{e}_k)$

Mit Hilfe des Zufallsgenerators $\mathcal{Z}$ wird ein beliebiger Wert ρ zwischen 0 und 1 generiert. Liegt ρ innerhalb eines beliebigen Intervalls $I(\underline{e}_\mu)$, dann nimmt das gemessene Multi-q-Bit den zugehörigen Basisvektor $\underline{v}^* = \underline{e}_\mu$ an. Dieses gemessene Multi-q-Bit wird durch einen Stern $(*)$ gekennzeichnet.

Hierzu einfaches Beispiel: Sei das Multi-q-Bit definiert durch

$$\underline{v} \;=\; \tfrac{1}{\sqrt{2}}\underline{e}_1 + \tfrac{1}{2}(\underline{e}_3 + \underline{e}_4)$$

Dann gilt für die Wahrscheinlichkeiten der Basisvektoren

$$P(\underline{e}_1) = \tfrac{1}{2}\ , \quad P(\underline{e}_2) = 0\ , \quad P(\underline{e}_3) = P(\underline{e}_4) = \tfrac{1}{4}$$

und für die zugehörigen Intervalle gilt

$$I(\underline{e}_1) = [\ 0\ ,\ \tfrac{1}{2}\ ]\ , \quad I(\underline{e}_2) = \emptyset\ , \quad I(\underline{e}_3) =]\ \tfrac{1}{2}\ ,\ \tfrac{3}{4}\ ]\ , \quad I(\underline{e}_4) =]\ \tfrac{3}{4}\ ,\ 1\ ]$$

Wählt der Zufallsgenerator $\rho = 0.197$ aus, dann liegt dieser Wert innerhalb des Intervalls $I(\underline{e}_1)$, und für das gemessene Multi-q-Bit gilt $\underline{v}^* = \underline{e}_1$. Wählt er hingegen $\rho = 0.768$, dann liegt dieser Wert innerhalb des Intervalls $I(\underline{e}_4)$, und für das gemessene Multi-q-Bit gilt $\underline{v}^* = \underline{e}_4$.

4.4.2 Phasenbehaftete Messungen

Im Beispiel aus Kapitel 4.4.1 hat das ungemessene Multi-q-Bit $\underline{v}$ nur positive λ-Koeffizienten. Diese können aber auch negativ, imaginär oder komplexwertig sein, wie das folgende Beispiel zeigt.

$$\underline{v} = -\tfrac{1}{2}\underline{e}_1 + \tfrac{i}{2}\underline{e}_2 + \tfrac{1-i}{2}\underline{e}_4$$

Für die Wahrscheinlichkeiten der Basisvektoren gilt hier:

$$P(\underline{e}_1) \;=\; P(\underline{e}_2) \;=\; \tfrac{1}{4}\ ,\ P(\underline{e}_3) \;=\; 0\ ,\ P(\underline{e}_4) \;=\; \tfrac{1}{2}$$

Die Intervallfunktionen lauten:

$$I(\underline{e}_1) \;=\; [\ 0\ ,\ \tfrac{1}{4}\ ]\ ,\ I(\underline{e}_2) \;=\;]\ \tfrac{1}{4}\ ,\ \tfrac{1}{2}\ ]\ ,\ I(\underline{e}_3) \;=\; \emptyset\ ,\ I(\underline{e}_4) \;=\;]\ \tfrac{1}{2}\ ,\ 1\ ]$$

Um mit Hilfe des Zufallsgenerators $\mathcal{Z}$ aus ungemessenen Multi-q-Bits gemessene „phasenbehaftete“ Multi-q-Bits zu erhalten, wird folgende Regel eingeführt:

Jedes gemessene phasenbehaftete Multi-q-Bit übernimmt die Phase des ausgewählten Basisvektors des ungemessenen Multi-q-Bits. Dieser übernommene Phasen-Parameter hat den Betrag 1.

Diese phasenbehaftete Messung wird in Abbildung 14 dargestellt:

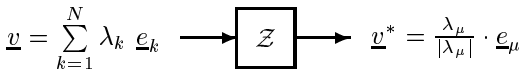

Abbildung 14: Phasenbehaftete Messung

Für positive reelle λ_μ ist $\frac{\lambda_\mu}{|\lambda_\mu|} = 1$. Somit stellt die phasenbehaftete Messung eine Verallgemeinerung der einfachen Messung aus Kapitel 4.4.1 dar.

Für das Beispiel ergibt die phasenbehaftete Messung in Abhängigkeit von ρ, dem zufällig ermittelten Zahlenwert des Generators $\mathcal{Z}$:

$$\underline{v}^* = \begin{cases} -\underline{e}_1 & \text{für } 0 \leq \rho \leq 0.25 \\ i \cdot \underline{e}_2 & \text{für } 0.25 < \rho \leq 0.5 \\ \frac{1-i}{\sqrt{2}}\underline{e}_4 & \text{für } 0.5 < \rho \leq 1 \end{cases}$$

Die Grundidee des stochastischen Algorithmus des Shor-Verfahrens stützt sich auf phasenbehaftete Messungen. Werden z.B. mehrere unterschiedliche phasenbehaftete Messungen durch Addition zusammengefaßt, dann können sich aufgrund gegensätzlicher Phasenwerte gleiche Basisvektoren ausblenden. Diese Eigenschaft würde bei einfachen Messungen nicht gelten.

Aus diesem Grund wird im folgenden bei Messungen immer von **phasenbehafteten** Messungen die Rede sein. Dabei wird unterschieden zwischen:

- **ungemessenen Multi-q-Bits:** Das sind diejenigen Multi-q-Bits, so wie sie in Kapitel 3.2 definiert sind. Diese Multi-q-Bits können aufgrund der durch λ_k implizierten Wahrscheinlichkeiten durch eine Messung jeden Basisvektor annehmen, dessen λ_k ungleich Null ist.
- **gemessenen Multi-q-Bits:** Das sind diejenigen Multi-q-Bits, die aufgrund einer über einen Zufallsgenerator $\mathcal{Z}$ ermittelten Wahrscheinlichkeit durch Messung einen phasenbehafteten Basisvektor-Wert annehmen. Gemessene Multi-q-Bits sind durch einen Stern $(*)$ gekennzeichnet.

Für das zuvorgenannte Beispiel bedeutet dies, daß das ungemessene Multi-q-Bit $\underline{v}$ durch die Wahl von $\rho = 0.438$ in ein gemessenes (phasenbehaftetes) Multi-q-Bit $\underline{v}^* = i\underline{e}_2$ übergeht.

4.4.3 Ein Beispiel für gemessene Multi-q-Bits

Gegeben sei ein System der Bit-Dimension $m = 1$, daß aus zwei Hadamard-Gates in Serienschaltung besteht. Abbildung 15 zeigt dieses System, bei dem wegen der Unitarität der Hadamard-Gates ($\underline{\underline{H}} \cdot \underline{\underline{H}} = \underline{\underline{E}}$) der Input gleich dem Output ist.

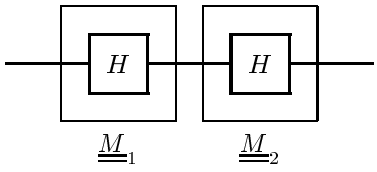

Abbildung 15: Zwei Hadamard-Gates in Serienschaltung

Für das Hadamard-Gate gilt bekanntlich:

$$\underline{\underline{H}} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

Das System aus Abbildung 15 wird nun von links mit dem Multi-q-Bit $\underline{v} = \frac{\sqrt{3}}{2}\underline{e}_1 + \frac{1}{2}\underline{e}_2$ angeregt. Damit ergibt sich folgendes Übertragungsverhalten:

$$\frac{\sqrt{3}}{2}\underline{e}_1 + \frac{1}{2}\underline{e}_2 \xrightarrow{\underline{\underline{M1}}} \frac{\sqrt{3}+1}{2\sqrt{2}}\underline{e}_1 + \frac{\sqrt{3}-1}{2\sqrt{2}}\underline{e}_2 \xrightarrow{\underline{\underline{M2}}} \frac{\sqrt{3}}{2}\underline{e}_1 + \frac{1}{2}\underline{e}_2$$

Dieses Übertragungsverhalten wird in Abbildung 16 durch ein Fluß-Diagramm dargestellt.

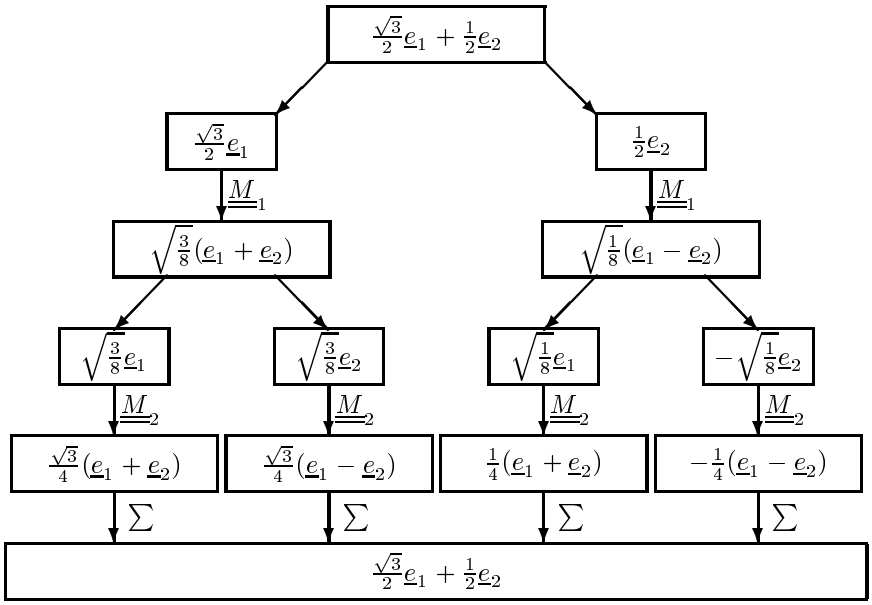

Abbildung 16: Fluß-Diagramm für die Hadamard-Gates in Serienschaltung

Liest man das Diagramm in Abbildung 16 von oben nach unten, dann wird jedes Multi-q-Bit in seine Basisvektoren (einschließlich Phase λ) aufgespalten, auf die dann die Gate-Operationen $\underline{\underline{M}}_1$ bzw. $\underline{\underline{M}}_2$ angewendet werden. Als Gesamtsumme erhält man den Output des Systems, der in diesem Fall bekanntlich gleich dem Input ist. Man erkennt, wie diese einfache Gesamtsumme zustandekommt: mehrere Summanden heben sich durch das Vorhandensein von positiven und negativen Vorzeichen weg.

Die Computational Resources auf dem klassischen Rechner ergeben pro Gate 4 reelle Multiplikationen und 2 reelle Additionen, somit insgesamt 8 Multiplikationen und 4 Additionen. Auf dem Quanten-Rechner wird dieses Beispiel durch 2 Gate-Operationen realisiert.

In Abbildung 16 wurde nur mit ungemessenen Multi-q-Bits gearbeitet. Das Fluß-Diagramm in Abbildung 17 zeigt, wie aus ungemessenen Multi-q-Bits gemessene Größen werden und wie diese im weiteren Verlauf verarbeitet werden.

Auch Abbildung 17 wird von oben nach unten gelesen. Gleich zu Beginn wird auf das Multi-q-Bit, den Input, der Zufallsgenerator $\mathcal{Z}$ gemäß Kapitel 4.4.1 und 4.4.2 angewendet. Da $I(\underline{e}_1) = [0{,}0.75]$ und $I(\underline{e}_2) =]0.75{,}1]$ ist, ist die Wahrscheinlichkeit, daß die Zufallszahl ρ den Basisvektor $\underline{e}_1$ ermittelt, dreimal so groß, wie wenn $\underline{e}_2$ gewählt wird. In Abbildung 17 wird dies dadurch verdeutlicht, daß drei Linien zur Box von $\underline{e}_1$ führen und von dort auch wegführen.
Auf die gemessenen Multi-q-Bits, d.h. auf die Basisvektoren $\underline{e}_1$ und $\underline{e}_2$ wird nun

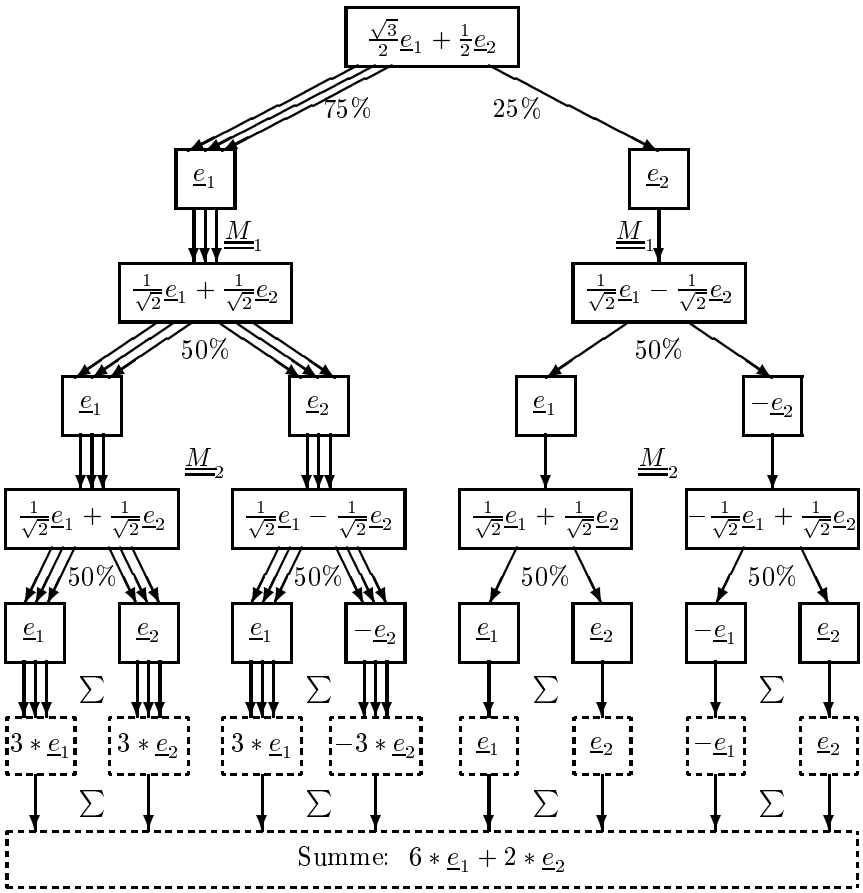

Abbildung 17: Fluß-Diagramm für gemessene Multi-q-Bits innerhalb der Serienschaltung der Hadamard-Gate

die Hadamard-Gate-Operation $\underline{\underline{M}}_1$ angewendet. Als Ergebnis erhält man wieder Multi-Q-Bits als Summe zweier Basisvektoren. Auf diese Multi-q-Bits wird wieder der Zufallsgenerator $\mathcal{Z}$ angewendet, der die Basisvektoren $\underline{e}_1$ und $\underline{e}_2$ mit gleichmäßiger Wahrscheinlichkeit (jeweils zu 50%) auswählt.
Dieses Verfahren wiederholt sich für die Hadamard-Gate-Operation $\underline{\underline{M}}_2$. Als Ergebnis stehen 8 Boxen in einer Zeile, deren Inhalt die Basisvektoren $\underline{e}_1$ und $\underline{e}_2$ mit positiven und negativen Vorzeichen sind. Insgesamt ergeben sich damit 8 **gemessene Pfade**.

Beispiel für einen gemessenen Pfad:

- Erzeugt der Zufallsgenerator $\mathcal{Z}$ beim Input ganz oben in Abbildung 17 die Zufallszahl $\rho = 0.37$, dann durchläuft der Pfad den linken Teil der Abbildung, der zum Basisvektor $\underline{e}_1$ führt. Damit ist $\underline{e}_1$ die gemessene Eingangsgröße.
- Für das Multi-q-Bit, das aus $\underline{\underline{M}}_1\,\underline{e}_1$ entstanden ist, generiert der Zufallsgenerator $\mathcal{Z}$ die Zahl $\rho = 0.89$. Der entsprechende Pfad führt zum Basisvektor $\underline{e}_2$. Dies ist die erste gemessene Ausgangsgröße.

- Für das Multi-q-Bit, das aus $\underline{\underline{M}}_2\underline{e}_2$ entstanden ist, generiert der Zufallsgenarator $\mathcal{Z}$ die Zahl $\rho = 0.63$. Der entsprechende Pfad führt zum Basisvektor $-\underline{e}_2$. Dies ist die zweite gemessene Ausgangsgröße.

- Dieses gemessene Multi-q-Bit befindet sich als Ergebnis des gemessenen Pfades in der vierten Box von links innerhalb der Zeile der 8 Boxen.

Abbildung 17 zeigt die 8 gemessenen Pfade, bei denen die linken 4 Pfade dreimal sooft wie die rechten 4 Pfade durchlaufen werden. Somit wird die linke Seite gegenüber der rechten Seite im Verhältnis 12:4 aufgerufen. Anders ausgedrückt, bei 16 Messungen im gesamten System laufen im Durchschnitt 12 in die linke Hälfte und 4 in die rechte von Abbildung 17.
Im nächsten Schritt wird für alle 8 parallelen Boxen die mögliche Anzahl der jeweiligen Pfad-Durchläufe mit dem Inhalt der Box multipliziert. Das Ergebnis wird in eine gestrichelte Box eingetragen. Diese gestrichelte Box verdeutlicht, daß es sich hierbei nicht mehr um einen Schritt innerhalb eines gemessenen Pfades handelt, sondern um die Nachverarbeitung des gesamten Prozesses. Im letzten Schritt werden alle Boxen zu einer Gesamtsumme aufaddiert. Dabei heben sich Summanden mit positivem und negativem Vorzeichen auf, analog wie im Fluß-Diagramm von Abbildung 16, so daß am Ende ein einfacher Term steht. Mit anderen Worten: Erst die Summierung aller Pfade unter Einbeziehung der Häufigkeit ihres Auftretens ermöglicht eine exakte Beschreibung des Systemverhaltens.
Das Summenergebnis ist jedoch kein Multi-q-Bit, sondern beschreibt das Wahrscheinlichkeitsverhältnis der Basisvektoren untereinander. Die Summe $6 * \underline{e}_1 + 2 * \underline{e}_2$ unterstreicht, daß die Wahrscheinlichkeit von $\underline{e}_1$ dreimal so hoch wie von $\underline{e}_2$ ist. D.h.

$$P(\underline{e}_1) : P(\underline{e}_2) = 3 : 1 \iff |\lambda_1| : |\lambda_2| = \sqrt{3} : 1$$

Außerdem besagt diese Summe, daß die Vorzeichen der Koeffizienten λ_1 und λ_2 positiv sind. Die Vorzeichen wurden bekanntlich in den gemessenen Pfade weitergereicht. Mit diesen Informationen kann das Multi-q-Bit des System-Outputs ermittelt werden.

$$\lambda_1 = \tfrac{\sqrt{3}}{2} \ \wedge \ \lambda_2 = \tfrac{1}{2} \implies \tfrac{\sqrt{3}}{2}\underline{e}_1 + \tfrac{1}{2}\underline{e}_1$$

Dieses Beispiel unterstreicht, daß das Verfahren der gemessenen Pfaden ein adequates Mittel zum Analysieren eines Quanten-Gate-Systems ist.

4.4.4 Computational Resources für gemessene Pfade

Bei der Berechnung der Computational Resources auf einem gemessenen Pfad fällt auf, daß dieser Pfad aus einer sich wiederholenden Abfolge von jeweils zwei in Serie geschalteten Operation besteht. Siehe Abbildung 18.

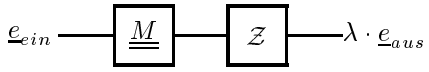

Abbildung 18: Blockschaltbild einer gemessenen Operation

- Die erste Operation besteht aus einem Gate $\underline{\underline{M}}$ der Bit-Dimension m (z.B. Hadamard $\underline{\underline{H}}^{\otimes m}$).
- Die zweite Operation besteht aus dem Zufallsgenerator $\mathcal{Z}$.

Am Anfang dieser zwei Operationen steht ein Basisvektor $\underline{e}_{ein}$ und am Ende ein Basisvektor $\underline{e}_{aus}$, der mit einem Phasen-Koeffizienten λ vom Betrag 1 $\{\lambda = \exp(i\alpha)\}$ versehen ist. Faßt man die beiden Operationen $\underline{\underline{M}}$ und $\mathcal{Z}$ zu einer einzigen zusammen, dann erhält man für die Darstellung der gemessenen Operation eine 2^m-dimensionale Matrix, die nur ein einziges von Null verschiedenes Element mit Betrag 1 enthält.

Beispiel: Ist die Eingangsgröße $\underline{e}_{ein} = \underline{e}_1$ und die gemessene Ausgangsgröße $\underline{e}_{aus} = i\underline{e}_2$, dann ergibt die Serienschaltung beider Operationen im Falle von $m = 1$ folgende Übertragungsfunktion:

$$\underline{\underline{M}} = \begin{pmatrix} 0 & 0 \\ i & 0 \end{pmatrix}$$

Das bedeutet, daß die Serienschaltung von $\underline{\underline{M}}$ und $\mathcal{Z}$ auf einem klassischen Rechner insgesamt nur eine Multiplikation benötigt. Da sich, wie bereits gesagt, auf einem gemessenen Pfad Gate- und Zufallsgenerator-Operation abwechseln, ist die Gesamtanzahl der multiplikativen Operationen auf einem gemessenen Pfad gleich der Anzahl der Gates, die von Anfang bis Ende durchlaufen werden.

Durch die Messung von Pfaden ist ein Verfahren entwickelt worden, das einen Algorithmus auf einem klassischen Rechner einem Algorithmus auf einem Quanten-Rechner in Bezug auf Computational Resources annähert. Der Vergleich beider Algorithmen läßt sich für ein System, das nur aus Ein-Bit-Gates besteht, folgendermaßen formulieren:

Auf einem gemessenen Pfad eines klassischen Rechners ist die Anzahl der multiplikativen Operationen gleich derjenigen Anzahl an Ein-Bit-Gates und somit gleich der Anzahl der Gate-Operationen auf einem Quanten-Rechner.

Diese Beobachtung täuscht allerdings nicht über folgende Tatsachen hinweg:

- Ein einzelner gemessener Pfad ist nur Teil des gesamten Systems. Im Beispiel aus Kapitel 4.4.3 gibt es 8 verschiedene Pfade. Nur in der additiven Summe dieser Pfade läßt sich das System exakt beschreiben.
- Die vereinfachte Übertragungsmatrix mit nur einem von Null verschiedenen Element vom Betrag 1 verrät nicht, mit welcher Wahrscheinlichkeit der korrespondierende Pfad gemessen wurde.

4.4.5 Messung des Hadamard-Gates

Bekanntlich läßt sich das Hadamard-Gate $H^{\otimes m}$ der Bit-Dimension m durch eine Hintereinanderschaltung von m Einbit-Hadamard-Gates realisieren. Abbildung 19 zeigt dies an einem Beispiel für $m = 2$.

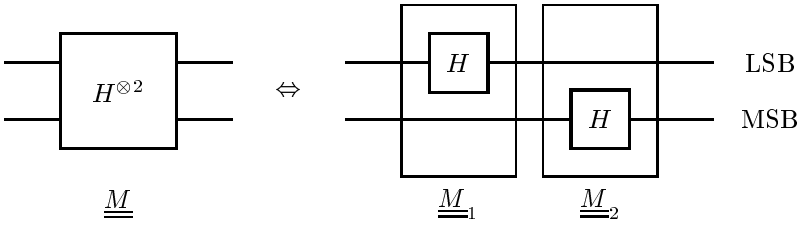

Abbildung 19: Hadamard-Gate der Dimension $m = 2$

Legt man beispielsweise für dieses System das Eins-Signal an das MSB an, wodurch $\underline{e}_3$ der Eingangsvektor zu diesem Gate wird, dann ergibt sich folgendes Übertragungsverhalten:

$$\underline{e}_3 \overset{\text{M1}}{\Longrightarrow} \tfrac{1}{\sqrt{2}}(\underline{e}_3 + \underline{e}_4) \overset{\text{M2}}{\Longrightarrow} \tfrac{1}{2}(\underline{e}_1 + \underline{e}_2 - \underline{e}_3 - \underline{e}_4)$$

In Abbildung 20 wird das Fluß-Diagramm für die gemessenen Pfade zu diesem Beispiel gezeigt, wenn $\underline{e}_3$ der Eingangsvektor ist. In dieser Abbildung ist links der Fluß für das Hadamard-Gate als kompletter Monolith dargestellt, rechts durchläuft das Diagramm die beiden Einbit-Gates $\underline{\underline{M}}_1$ und $\underline{\underline{M}}_2$, analog zur Darstellung in Abbildung 19.

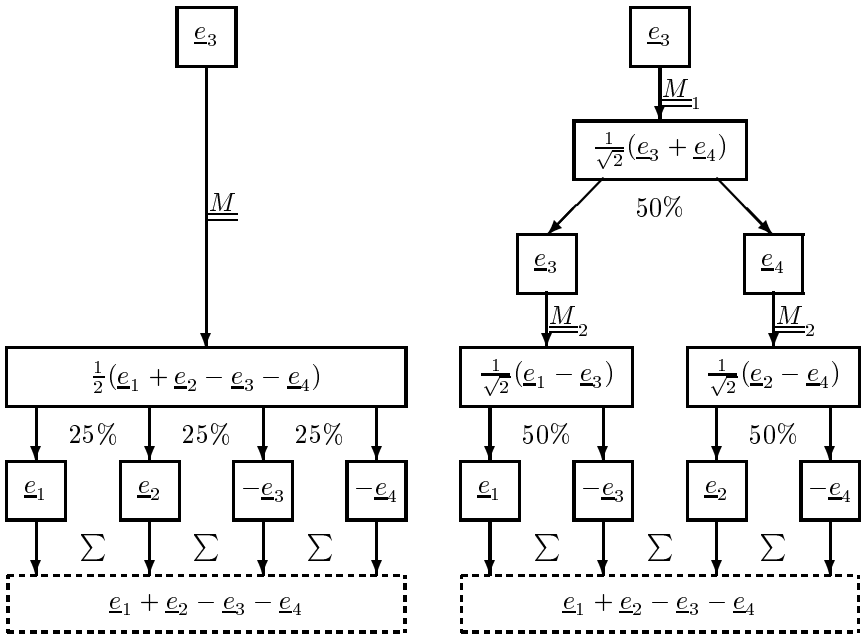

Abbildung 20: Fluß-Diagramm zu gemessenen Multi-q-Bits innerhalb des Hamadard-Gate der Dimension $m = 2$ für das Eingangssignal $\underline{e}_3$

Die Erkenntnisse für $m = 2$ aus Abbildung 20 lassen sich auch auf Hadamard-Gates beliebiger Bit-Dimension m übertragen:

- Alle 2^m gemessenen Pfade werden mit der gleichen Wahrscheinlichkeit $P = (\frac{1}{2})^m$ (Gleichverteilung) durchlaufen und unterscheiden sich nur durch ihr Vorzeichen.
- In der Matrix $\underline{\underline{M}}$ der monolothischen Darstellung definiert jedes Matrixelement m_{ij} den Phasenkoeffizienten des i-ten gemessenen Pfades, wobei $\underline{e}_j$ das Multi-q-Bit des Eingangs und $\underline{e}_i$ der Basis-Vektor des Ausgangs des gemessenen Pfades ist, siehe Abbildung 21.
- Die monolithische Darstellung des Hadamard-Gates links in Abbildung 20 ermöglicht es, den kompletten gemessenen Pfad durch eine einzige Gate-Operation mit anschließender Zufallsoperation über $\mathcal{Z}$ zu ermitteln. Die Kombination aus Gate-Operation und Zufallsoperation ergibt als Gesamtes eine Matrix mit einem einzigen von Null verschiedenen Element mit Betrag 1. (Vergleiche hierzu das Beispiel in Kapitel 4.4.4.) Damit läßt sich auf einem klassischen Rechner der gemessene Pfad des Hadamard-Gates mit Bit-Dimension m durch eine einzige multipikative Operation darstellen.

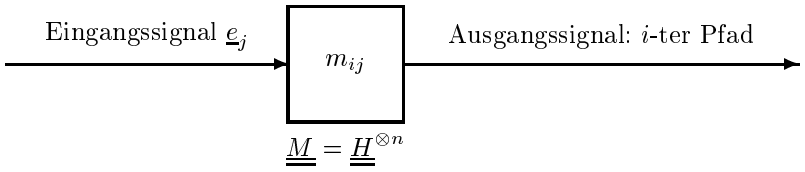

Abbildung 21: Bedeutung der Hadamard-Matrix-Elemente für die Messungen

4.4.6 Messung der Inversen Fourier-Transformation

In Abbildung 11 wurde gezeigt, wie sich anhand von [NI] eine Inverse Fourier-Transformation der Bit-Dimension $m = 3$ durch eine Hintereinanderschaltung mehrerer Einbit-Gates realiseren läßt. Ganz analog dazu wird in Abbildung 22 eine inverse Fourier-Transformation der Bit-Dimension $m = 2$ gezeigt. Links diese IFT als Monolith, rechts zerlegt in einzelne Einbit-Gates.

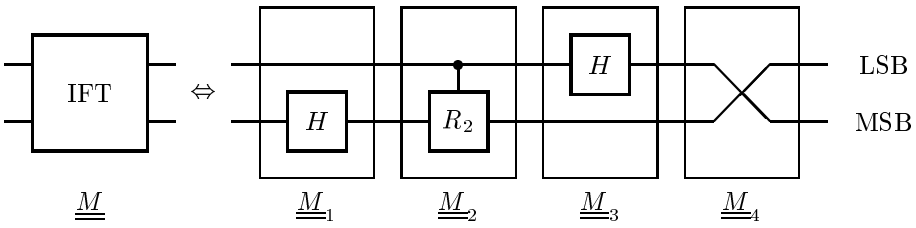

Abbildung 22: Inverse Fourier-Transformation der Dimension $m = 2$

Von den Übertragungsfunktionen sind bis auf $\underline{\underline{M}}_2$ alle Matrizen schon einmal hergeleitet worden.

$$\underline{\underline{M}} = \frac{1}{2}\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{pmatrix} \quad \underline{\underline{M}}_1 = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{pmatrix}$$

$$\underline{\underline{M}}_3 = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \end{pmatrix} \quad \underline{\underline{M}}_4 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Die Matrixdarstellung von $\underline{\underline{R}}_k$ lautet gemäß [NI]:

$$\underline{\underline{R}}_k = \begin{pmatrix} 1 & 0 \\ 0 & \exp(i2\pi/2^k) \end{pmatrix} \quad \Longrightarrow \quad \underline{\underline{R}}_2 = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}$$

$\underline{\underline{M}}_2$ ist ein „Controlled“-$\underline{\underline{R}}_2$-Gate mit folgenden Beziehungen:

$$\begin{array}{llll} |0>|0> \rightarrow |0>|0> & \Longleftrightarrow & \underline{e}_1 \rightarrow \underline{e}_1 \\ |1>|0> \rightarrow |1>|0> & \Longleftrightarrow & \underline{e}_2 \rightarrow \underline{e}_2 \\ |0>|1> \rightarrow |0>|1> & \Longleftrightarrow & \underline{e}_3 \rightarrow \underline{e}_3 \\ |1>|1> \rightarrow |1>|i> & \Longleftrightarrow & \underline{e}_4 \rightarrow i\underline{e}_4 \end{array} \quad \Longrightarrow \quad \underline{\underline{M}}_2 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & i \end{pmatrix}$$

Legt man beispielsweise an den Eingang des in Abbildung 22 dargestellten Systems das Eins-Signal an das LSB an, wodurch $\underline{e}_2$ der Eingangsvektor zu diesem Gate wird, dann ergibt sich folgendes Übertragungsverhalten:

$$\begin{aligned} \underline{e}_2 \quad & \overset{\underline{\underline{M}}1}{\Longrightarrow} \frac{1}{\sqrt{2}}(\underline{e}_2 + \underline{e}_4) \overset{\underline{\underline{M}}2}{\Longrightarrow} \frac{1}{\sqrt{2}}(\underline{e}_2 + i \cdot \underline{e}_4) \\ & \overset{\underline{\underline{M}}3}{\Longrightarrow} \frac{1}{2}(\underline{e}_1 - \underline{e}_2 + i \cdot \underline{e}_3 - i \cdot \underline{e}_4) \overset{\underline{\underline{M}}4}{\Longrightarrow} \frac{1}{2}(\underline{e}_1 + i \cdot \underline{e}_2 - \underline{e}_3 - i \cdot \underline{e}_4) \end{aligned}$$

In Abbildung 23 wird zu diesem Beispiel das Fluß-Diagramm für die gemessenen Pfade gezeigt, bei dem $\underline{e}_2$ der Eingangsvektor ist. In dieser Abbildung ist links der Fluß für das IFT-Gate als kompletter Monolith dargestellt, rechts durchläuft das Diagramm die Einbit-Gates $\underline{\underline{M}}_1$, $\underline{\underline{M}}_2$, $\underline{\underline{M}}_3$ und $\underline{\underline{M}}_4$ analog zur Darstellung in Abbildung 22.

Die Erkenntnisse für $m = 2$ aus Abbildung 23 lassen sich auch auf IFT-Gates beliebiger Bit-Dimension m übertragen:

- Alle 2^m gemessenen Pfade werden mit der gleichen Wahrscheinlichkeit $P = (\frac{1}{2})^m$ (Gleichverteilung) durchlaufen und unterscheiden sich nur durch ihren Phasenkoeffizienten.
- In der Matrix $\underline{\underline{M}}$ der monolothischen Darstellung definiert jedes Matrixelement m_{ij} den Phasenkoeffizienten eines gemessenen Pfades, wobei $\underline{e}_j$ das Multi-q-Bit des Eingangs und $\underline{e}_i$ der Basis-Vektor des Ausgangs des gemessenen Pfades ist. Allerdings handelt es sich hierbei nicht üblicherweise um den j-ten gemessenen Pfad, wie man sich, bedingt durch die Tauschaktion bei $\underline{\underline{M}}_4$, sehr leicht im rechten Teil der Abbildung 23 überzeugen kann.

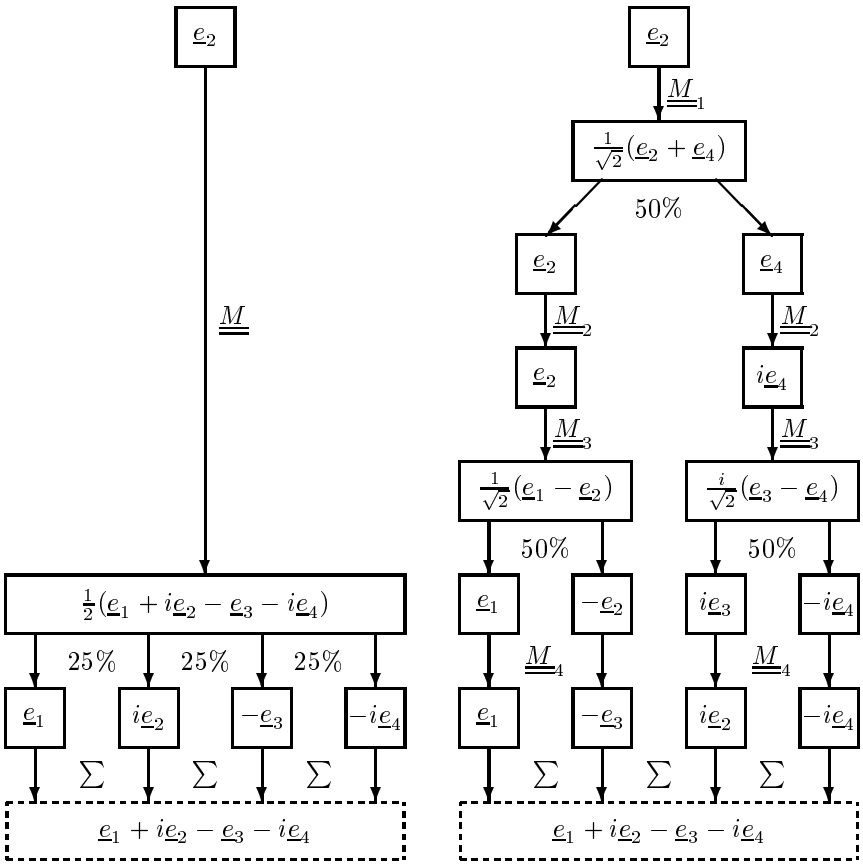

Abbildung 23: Fluß-Diagramm zu gemessenen Multi-q-Bits innerhalb der inversen Fourier-Transformation der Dimension $m = 2$ für das Eingangssignal $\underline{e}_2$

- Mit Bezug auf die Computational Resources verhält sich der gemessene Pfad der monolithischen Darstellung der IFT links in Abbildung 23 ähnlich wie derjenige der monolithischen Darstellung des Hadamard-Gates. Dieser Monolith ermöglicht es, den kompletten gemessenen Pfad durch eine einzige Gate-Operation mit anschließender Zufallsoperation über $\mathcal{Z}$ zu ermitteln. Die Kombination aus Gate-Operation und Zufallsoperation ergibt als Gesamtes eine Matrix mit einem einzigen von Null verschiedenen Element mit Betrag 1. Damit läßt sich auf einem klassischen Rechner der gemessene Pfad der IFT mit Bit-Dimension m durch eine einzige multipikative Operation darstellen.

4.4.7 Computational Resources im Werte-Hilbertraum

Bei der Aufwands-Abschätzung des gemessenen Pfades im Shor-Algorithmus spielt die einmalige Berechnung von $a^k (\mathrm{mod}\ n)$ im Werte-Hilbertraum eine wichtige Rolle. Dieser Aufwand liegt gemäß Kapitel 3.5 bei $O(L^3)$ Rechen-Operationen und ist als polynomialer Faktor immer mitzuführen.

4.5 Das Shor-Verfahren als mathematisch-stochastischer Algorithmus

Mit Hilfe der in Kapitel 4.4 erläuterten Messungen wird in diesem Kapitel der stochastische Algorithmus für das Shor-Verfahren entwickelt.

4.5.1 Das Vierphasenmodell mit gemessenen Pfaden

Ein einer Messung unterzogenes Multi-q-Bit wird durch einen Stern ($*$) gekennzeichnet. Siehe Abbildung 12 und Abbildung 14.

Phase 0: Der Startvektor

In Analogie zu Gl.(82) aus Kapitel 4.3.1 erhält man für den Startvektor

$$\underline{v}_0 \;=\; \underline{e}_{1+q} \;=\; \underline{v}_0^*$$

Dieser Startvektor wird durch die Messung nicht verändert. Der gemessene ist somit gleich dem ungemessenen Startvektor. Für jedes nachfolgende Gate ist nur die $(q+1)$-te Spalte interessant.

Phase 1: Das Hadamard-Gate

Mit Hilfe des Hadamard-Gates mit Bit-Dimension m im Argument-Hilbertraum $\mathcal{H}_1$ ergibt sich gemäß Kapitel 4.3.1 für $\underline{v}_1$

$$\underline{v}_1 \;=\; \tfrac{1}{\sqrt{q}} \sum_{k=0}^{q-1} \underline{e}_{1+k+q}$$

Nachdem $\underline{v}_1$ einer Messung unterzogen wird, erhält man

$$\underline{v}_1^* \;=\; \underline{e}_{1+\mu+q} \quad \text{für} \quad \mu \in [0, q-1] \tag{98}$$

Phase 2: Das Controlled-U-Gate

Wegen der in Kapitel 4.3.1 erläuterten Funktionszuweisung des Controlled-U-Gates

$$\underline{e}_{1+\mu+q} \longrightarrow \underline{e}_{1+\mu+qf(\mu)}$$

erhält man als gemessenen und ungemessenen Vektor

$$\underline{v}_2 \;=\; \underline{v}_2^* \;=\; \underline{e}_{1+\mu+qf(\mu)} \quad \text{für} \quad \mu \in [0, q-1] \tag{99}$$

Phase 3: Die inverse Fourier-Transformation

Wegen der in Kapitel 4.3.1 erläuterten Funktionszuweisung der IFT

$$\underline{e}_{1+k+qf(k)} \xrightarrow{\text{IFT}} \tfrac{1}{\sqrt{q}} \sum_{c=0}^{q-1} \underline{e}_{1+c+qf(k)} \exp(i2\pi \tfrac{ck}{q})$$

erhält man für den ungemessenen Vektor

$$\underline{v}_3 \;=\; \tfrac{1}{\sqrt{q}} \sum_{c=0}^{q-1} \underline{e}_{1+c+qf(\mu)} \exp(i2\pi \tfrac{c\mu}{q}) \;\; \text{für} \;\; \mu \in [0, q-1]$$

Wird $\underline{v}_3$ einer Messung unterzogen, dann ergibt der gemessene Vektor

$$\underline{v}_3^* \;=\; \underline{e}_{1+c+qf(\mu)} \exp(i2\pi \frac{c\mu}{q}) \;\; \text{für} \;\; c \in [0, q-1] \text{ und } \mu \in [0, q-1] \qquad (100)$$

Phase 4: Ergebnis mit oder ohne Kettenbruch-Zerlegung

An dieser Stelle des stochastischen Verfahrens macht die Kettenbruch-Zerlegung keinen Sinn. Sie wird zu einem späteren Zeitpunkt benötigt und deshalb hier weggelassen.

4.5.2 Beispiel für das Vierphasenmodell

Für das Beispiel $n = 3$ und $a = 2$ aus Kapitel 4.2.6 und 4.3.2 gilt mit $q = 4$:

$$\underline{v}_0 \;=\; \underline{v}_0^* \;=\; \underline{e}_5$$

$$\underline{v}_1 \;=\; \tfrac{1}{2} \sum_{\mu=0}^{3} \underline{e}_{5+\mu} \;\; \overset{\mathcal{Z}:\rho=0.69}{\longrightarrow} \;\; \underline{v}_1^* \;=\; \underline{v}_1|_{\mu=2} \;=\; \underline{e}_{5+2} \;=\; \underline{e}_7$$

Hierbei wurde $\rho = 0.69$ über den Zufallsgenerator $\mathcal{Z}$ willkürlich ermittelt. Der Wert $\mu = 2$ ergibt sich für $\rho > \frac{1}{q}$ aus

$$\rho \in I(\underline{e}_{5+\mu}) =]\tfrac{\mu}{q}, \tfrac{\mu+1}{q}] \;\longrightarrow\; \mu = \lfloor \rho \cdot q \rfloor = \lfloor 0.69 \cdot 4 \rfloor = \lfloor 2.76 \rfloor$$

Der Übergang von $\underline{v}_1^*$ nach $\underline{v}_2^*$ erfolgt ohne Zufallsgenerator.

$$\underline{v}_2^* \;=\; \underline{e}_{1+2+4 \cdot f(2)} \;=\; \underline{e}_7$$

Mit diesem reduzierten Eingangsvektor ergibt der Ausgangsvektor

$$\underline{v}_3 \;=\; \tfrac{1}{2} \sum_{c=0}^{3} \underline{e}_{5+c} \exp(i2\pi \tfrac{2 \cdot c}{4}) \;\; \overset{\mathcal{Z}:\rho=0.38}{\longrightarrow} \;\; \underline{v}_3^* \;=\; \underline{v}_3|_{c=1}$$

$$\underline{v}_3^* \;=\; \underline{e}_{5+1} \exp(i2\pi \tfrac{2 \cdot 1}{4}) \;=\; -\underline{e}_6$$

Auch hier wurde $\rho = 0.38$ über den Zufallsgenerator $\mathcal{Z}$ willkürlich ermittelt. Das Ergebnis für $\underline{v}_3^*$ hätte man auch dadurch erzielt, indem man $\mu = 2$ und $c = 1$ direkt in Gl.(100) eingesetzt hätte.

4.5.3 Interpretation des gemessenen Vierphasenmodells

Aufgrund der Gleichverteilung der Wahrscheinlichkeiten für Hadamard- und IFT-Gate ist die Abhängigkeit des gemessenen Endvektors $\underline{v}_3^*$ in Gl.(100) von den Parametern μ und c einfach zu interpretieren (siehe auch Kapitel 4.5.2). Hierbei ist der Vergleich mit $\underline{v}_3$ gemäß Gl.(91) hilfreich.

- Der Basisvektor von $\underline{v}_3^*$ lautet $\underline{e}_{1+c+qf(\mu)}$. D.h. der Koeffizient des Basisvektors errechnet sich zu $1+c+qf(\mu)$. In der physikalischen Dirac-Notation ergibt dieser Basisvektor $|c> |f(\mu)>$.
- Der Phasen-Koeffizient von $\underline{v}_3^*$ deckt sich mit einem Element der einfachen (q x q)-IFT-Matrix. In dieser einfachen IFT-Matrix würde sich dieses Element in Zeile $(c+1)$ und Spalte $(\mu+1)$ wiederfinden.
- Die Zeilenposition dieses Phasen-Koeffizienten innerhalb der Gesamt-IFT-Matrix ist gleich dem Koeffizienten des Basisvektors von $\underline{v}_3^*$ und wird somit durch $1+c+qf(\mu)$ definiert.
- Die Spaltenposition dieses Phasen-Koeffizienten innerhalb der Gesamt-IFT-Matrix wird durch den Koeffizienten des Basisvektors von $\underline{v}_2^*$, also durch den Term $1+\mu+qf(\mu)$, bestimmt.
- Eine einfache Positionsbeschreibung des Phasen-Koeffizienten von $\underline{v}_3^*$ innerhalb der Gesamt-IFT-Matrix von $\underline{v}_3$ ergibt sich auch mit Hilfe von Gl.(90) und Gl.(91). Zählt man die Untermatrizen $\underline{\underline{M}}_{IFT0}$ von links oben nach rechts unten durch, dann befindet sich der Phasen-Koeffizient in der $[f(\mu)+1]$-ten Untermatrix. Innerhalb dieser Untermatrix ist die Zeilenposition $(c+1)$ und die Spaltenposition $(\mu+1)$, somit die gleiche wie in der einfachen (q x q)-IFT-Matrix.

Abbildung 24 zeigt die Abhängigkeit des Phasen-Koeffizienten von $\underline{v}_2^*$ und $\underline{v}_3^*$.

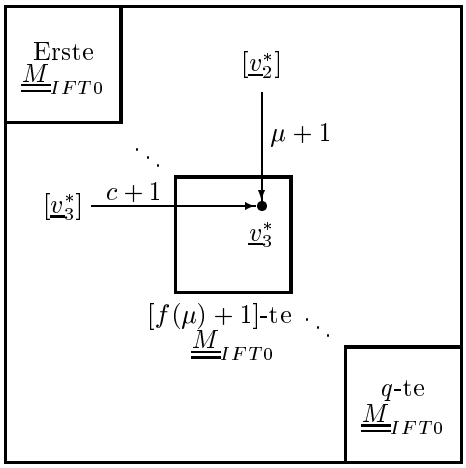

Abbildung 24: Abhängigkeit des IFT-Elements von $\underline{v}_2^*$ und $\underline{v}_3^*$

Für das Beispiel in Kapitel 4.5.2 ist $\underline{v}_3^*$ wegen $\mu = 2$ und $c = 1$ durch das Element $m_{1+c+qf(\mu),1+\mu+qf(\mu)} = m_{6,7}$ in der Gesamt-IFT definiert. Dieses Element befindet sich in der 2-ten Untermatrix in der $(c+1=2)$-ten Zeile und $(\mu+1=3)$-ten Spalte.

Wichtiges Fazit:

Folgende wichtige Erkenntnisse ergeben sich somit:

1. Für den einzelnen gemessenen Pfad des Shor-Verfahrens kann auf die Verwendung des Zufallsgenerators $\mathcal{Z}$, der für ρ Werte im reellen Bereich von 0 bis 1 ermittelt, verzichtet werden, da der Vektor $\underline{v}_3^*$ nur durch die beiden Koeffizienten c und μ definiert wird. Diese Koeffizienten können von einem alternativen Zufallsgenerator im Intervall von 0 bis $q-1$ ermittelt werden.

2. Verfolgt man den gemessenen Pfad des Shor-Algorithmus auf dem klassischen Rechner von seinem Start im Hadamard-Gate über das Controlled-U-Gate bis hin zu seinem Abschluß in der IFT, dann besteht er aus einer einzigen multiplikativen Operation. Diese Operation wird durch den von c und μ abhängigen Koeffizienten der IFT-Matrix definiert.

Bei all diesen Überlegungen muß allerdings berücksichtigt werden, daß gemäß Kapitel 4.4.7 die Computational Resources für die Berechnung der Modulo-Exponentialfunktion $f(k) = a^k \pmod n$ des Controlled-U-Gates im Werte-Hilbertraum bei $O(L^3)$ liegt und als Faktor mit in die Berechnung eingehen muß.

4.5.4 Wahl der Anzahl gemessener Pfade

Am Ende von Kapitel 4.4.4 wurde bereits erwähnt, daß der einzelne gemessene Pfad (**EGP**) das gesamte System des Shor-Verfahrens nicht beschreibt. Zur Berechnung des kompletten mathematischen Algorithmus (**KMA**) werden insgesamt $q^2 = 2^{2m}$ Elemente der Gesamt-IFT-Matrix und damit ebenso viele unterschiedliche EGPn aufsummiert. In Beispiel 1 aus Kapitel 4.3.2 sind diese $q^2 = 16$ Elemente fett unterlegt.

Dem KMA wird nun der Teilsummen-Algorithmus (**TSA**) gegenübergestellt. Hierbei handelt es sich um das Aufsummieren s verschiedenartiger EGPn zu einer Teilsumme, bei der s kleiner gleich der Anzahl aller EGPn im KMA ist. Diese Teilsumme gibt mehr Auskunft über das Gesamtverhalten des Systems als der EGP und weniger Auskunft als der umfassende KMA. Somit liegt der TSA zwischen EGP und KMA. Je mehr EGPn zu einem TSA zusammengefaßt werden, umso ähnlicher wird der TSA dem KMA.

Übertragen auf einen stochastischen Algorithmus für das Shor-Verfahren bedeutet der TSA eine zufällige Auswahl von s aus insgesamt q^2 relevanten Elementen der Gesamt-IFT-Matrix, die zu einer reduzierten IFT-Matrix zusammengefaßt werden und bei der Berechnung eines reduzierten Endvektors $\underline{v}_{3red}$ berücksichtigt werden. Die Anzahl $s = s(m)$ ist ein einfaches Polynom von m mit kleinem Exponenten p.

$$s(m) \;=\; k \cdot m^p \tag{101}$$

Hier stellt sich die Frage, wie groß der TSA, der das Systemverhalten des Shor-Verfahrens annähernd gut beschreiben kann, sein soll. Kann hierbei p hinreichend klein gewählt werden, so daß

$$s(m) << q^2 \tag{102}$$

gilt? Hat p, der Exponent in Gl.(101), eine maximale Obergrenze p_{max} mit

$$p \leq p_{max} \text{ für alle } n \text{ ?} \tag{103}$$

Nur in dem Fall, daß letztere Frage mit „Ja“ beantwortet wird, würde aus mathematischer Sicht eine polynomiale Anzahl an EGPn, insgesamt einen polynomialen Gesamtaufwand bedeuten. Damit würde auch die polynomiale Brücke zwischen den Aufwänden auf dem klassischen Rechner und Quanten-Rechner erhalten bleiben.

Zwar können die EGPn über den Zufallgeneratoren beliebig gewählt werden. **In den TSA sollen aber nur unterschiedliche EGPn eingehen.** D.h. ein bereits vorhandener EGP wird verworfen und die Suche nach einem nicht-schon-vorhandenen EGP wiederholt.

Übertragen auf die Interpretation von Kapitel 4.5.3 besteht ein TSA aus insgesamt $s = s(m)$ EGPn, die aus unterschiedlichen Zahlenpaaren (c, μ) über Gl.(100) gebildet werden.

4.5.5 Teilsummen-Algorithmus mit Sortieren der Zufallszahlen

Im folgenden wird ein TSA in Anlehnung an den KMA gemäß Kapitel 4.3 skizziert.

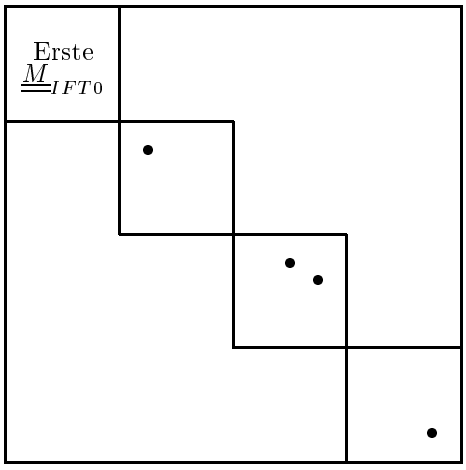

Abbildung 25: Auswahl von 4 Pfaden in der IFT-Matrix

Bild 25 zeigt die Auswahl von vier EGPn anhand der IFT-Matrix. Die Auswahl dieser vier EGPn erfolgt über einen Zufallsgenerator, wobei folgendes Vorgehensweise sich anbietet.

- Die Gesamt-IFT-Matrix beinhaltet u.a. diejenigen $q^2 = 2^{2m}$ von Null verschiedenen Elemente, die mit $\underline{v}_2$ multipliziert werden. Das sind die Phasen-Koeffizienten aller EGPn im KMA. Nummeriert man diese Elemente von links oben nach rechts unten durch, dann erhalten auch alle EGPn dementsprechend eine Nummer zwischen 1 und 2^{2m}.
- Mit einem Zufallsgenerator werden aus diesen 2^{2m} Nummern insgesamt $s(m) = km^p$ Nummern ausgewählt. Die zu diesen Nummern korrespondierenden EGPn werden ermittelt und der Gesamtsumme eines reduzierten $\underline{v}_{3red}$ zugeführt.
- Die Argument-Wahrscheinlichkeiten $P_{red}(|c>)$ werden auf maximale Größen überprüft. Die zugehörigen c-Werte werden der Kettenbruch-Zerlegung unterzogen, um die Ordnung r zu ermitteln.

Beispiel:
Im vereinfachten Beispiel 1 aus Kapitel 4.3.2 wurde die Gesamt-IFT-Matrix bekanntlich folgendermaßen definiert:

$$\underline{\underline{M}}_{IFT} = \frac{1}{2}\begin{pmatrix}
1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & i & -1 & -i & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & -1 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & -i & -1 & i & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \mathbf{1} & 1 & \mathbf{1} & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \mathbf{1} & i & \mathbf{-1} & -i & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \mathbf{1} & -1 & \mathbf{1} & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \mathbf{1} & -i & \mathbf{-1} & i & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & \mathbf{1} & 1 & \mathbf{1} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & \mathbf{i} & -1 & \mathbf{-i} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & \mathbf{-1} & 1 & \mathbf{-1} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & \mathbf{-i} & -1 & \mathbf{i}
\end{pmatrix}$$

Nummeriert man die fett unterlegten Elemente der Gesamt-IFT-Matrix durch, dann finden sich die EGPn des Shor-Algorithmus an den Stellen 1 bis 16 wieder.

$$EGP(\underline{\underline{M}}_{IFT}) = \begin{pmatrix}
x & x & x & x & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
x & x & x & x & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
x & x & x & x & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
x & x & x & x & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \mathbf{[1]} & x & \mathbf{[2]} & x & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \mathbf{[3]} & x & \mathbf{[4]} & x & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \mathbf{[5]} & x & \mathbf{[6]} & x & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \mathbf{[7]} & x & \mathbf{[8]} & x & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & x & \mathbf{[\ 9]} & x & \mathbf{[10]} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & x & \mathbf{[11]} & x & \mathbf{[12]} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & x & \mathbf{[13]} & x & \mathbf{[14]} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & x & \mathbf{[15]} & x & \mathbf{[16]}
\end{pmatrix}$$

Aus den insgesamt 16 Nummern können s Nummern ausgewählt werden. Werden z.B. für die Anzahl $s = 4$ die Nummern 3, 8, 11 und 15 gewählt, dann ergeben sich folgende korrespondierende EGPn:

$$\underline{v}^*_{3,1} = \underline{e}_6 \ \wedge \ \underline{v}^*_{3,2} = -\underline{e}_8 \ \wedge \ \underline{v}^*_{3,3} = i\underline{e}_{10} \ \wedge \ \underline{v}^*_{3,4} = -i\underline{e}_{12}$$

Die Summe der EGPn ergibt einen TSA mit folgendem Ergebnis:

$$\underline{e}_6 - \underline{e}_8 + i\underline{e}_{10} - i\underline{e}_{12}$$

Folgende drei Bemerkungen sind an dieser Stelle von Wichtigkeit:

- Da es sich bei den Ergebnissen der EGPn um Multi-q-Bits handelt, sind ihre Ergebnisse immer vom Betrag 1.
- Die Phasenausrichtung ($\exp[i2\pi x]$) der EGPn bleibt erhalten und geht als Summand in den TSAs ein. Dadurch können sich in Summe mehrere EGPn gegenseitig ausblenden.
- Der TSA ist eine Summe von EGPn, somit als Ganzes kein Multi-q-Bit.

Aus der ersten Bemerkung geht hervor, daß alle EGPn zueinander im gleichen Verhältnis stehen. Die letzte Bemerkung läßt zu, daß ein gemeinsamer gleicher konstanter Faktor für alle EGPn keine Folgen für die Bewertung des Gesamtergebnisses hat, wie man an

$$\underline{v}_{3red} \ = \ \tfrac{1}{4}\{\underline{e}_6 - \underline{e}_8 + i\underline{e}_{10} - i\underline{e}_{12}\}$$

erkennen kann. Hierbei wurden alle EGPn mit $\frac{1}{q}$ multipliziert, so wie das beim Shor-Verfahren üblich ist. Das hat den Vorteil, daß man die EGPn direkt aus dem Shor-Algorithmus entnehmen kann und daß die reduzierten Basis-Wahrscheinlichkeiten P_{red}, die sich analog zum kompletten Shor-Verfahren durch das Betragsquadrat der Koeffizienten der korrespondierenden Vektoren $\underline{e}_i$ ergeben, kleiner 1 bleiben.

Überträgt man $\underline{v}_{3red}$ in die physikalische Dirac-Notation und berechnet die reduzierten Basis-Wahrscheinlichkeiten, dann erhält man:

$$\begin{aligned} P_{red}(|1>|1>) \ &= \ P_{red}(|1>|2>) \ = \ \tfrac{1}{16} \\ P_{red}(|3>|1>) \ &= \ P_{red}(|3>|2>) \ = \ \tfrac{1}{16} \end{aligned}$$

Die reduzierten Argument-Wahrscheinlichkeiten ergeben:

$$P_{red}(|1>) \ = \ P_{red}(|3>) \ = \ \tfrac{1}{8}$$

Unterzieht man $c = 3$ der Kettenbruch-Zerlegung, dann ergibt sich für die Ordnung $r = 4$. Dieser Wert stimmt nicht überein mit der wahren Ordnung $r = 2$ für dieses Beispiel. D.h. das verkürzte Verfahren führt nicht immer zum richtigen Ziel, was auch nicht zu erwarten ist.

Fazit:

Der hier geschilderte Programmier-Algorithmus benutzt für die Zufallswerte Zahlen, die durch die Nummerierung eine eindeutige Zuordnung zu einem EGP darstellen und einem Element der Gesamt-IFT-Matrix eindeutig die Position zuweisen. Um diese Zufallswerte für einen TSA nutzbar zu machen, sind folgende Schritte nötig:

- Aus der Menge $M = [0, 1 \ldots q^2 - 1]$ werden über einen Zufallsgenerator $s(m)$ unterschiedliche Zufallszahlen ausgesucht.
- Diese Zufallszahlen werden sortiert, damit ein speicher-optimierter TSA-Algorithmus diese den EGPn direkt zuordnen und dementsprechend verwenden kann.

Beide genannten Schritte, insbesondere der zweite, sind auf einem 32-Bit-Rechner für Werte $n \geq 91$ sehr zeitintensiv.

4.5.6 Teilsummen-Algorithmus mit Vorkenntnis der Ordnung r

Im folgenden TSA kann auf das Sortieren von Zufallszahlen verzichtet werden, wodurch viel Rechenzeit auf einem 32-Bit-Rechner eingespart wird. Allerdings wird hierbei die Ordnung r von Anfang an als bereits bekannt vorausgesetzt. Da der Hauptschwerpunkt dieser Arbeit nicht im Berechnen der Ordnung r liegt, sondern im Ermitteln von TSAs mit minimaler Anzahl an EGPn, die zur Ordnung r führen, ist dieser Umstand nicht von Bedeutung.

Grundlage des hier vorgestellten TSAs bildet Gl.(70), die hier noch einmal dargestellt wird, allerdings mit vertauschten Summenzeichen.

$$\begin{aligned} |v_3> = \quad & \tfrac{1}{q} \sum_{y=0}^{Y-1} \sum_{x=0}^{X} \sum_{c=0}^{q-1} |c> |f(y)> \exp(i2\pi \tfrac{(xr+y)c}{q}) \\ & + \tfrac{1}{q} \sum_{y=Y}^{r-1} \sum_{x=0}^{X-1} \sum_{c=0}^{q-1} |c> |f(y)> \exp(i2\pi \tfrac{(xr+y)c}{q}) \end{aligned}$$

Diese Gleichung beinhaltet zwei Summenblöcke, von denen der obere Summenblock aus $qY(X+1)$ und der untere Summenblock aus $q(r-Y)X$ Summanden besteht. Insgesamt sind dies $q(Y + rX) = q^2$ Terme, von denen jeder Summand auf eine Zufallszahl z zwischen 0 und $q^2 - 1$ abgebildet werden kann.

Der TSA durchläuft folgende Schritte:

- Auswahl einer reduzierten Anzahl an Zufallszahlen aus der Menge der ganzen Zahlen zwischen 0 und $(q^2 - 1)$.
- Abbildung dieser Zufallszahlen auf die Summanden in $|v_3>$.
- Zusammenfassen dieser Summanden zu einem reduzierten $|v_{3red}>$.
- Bestimmung der reduzierten Basis-Wahrscheinlichkeiten und Berechnung der Ordnung r über die reduzierten Argument-Wahrscheinlichkeiten P_{red}.

Die Abbildung der Zufallszahl z auf einen Summanden in $|v_3>$ erfolgt über die Abhängigkeit des Summanden von den Summations-Koeffizienten c, x und y. An dieser Stelle ist es ratsam, für die Zufallszahl z ein Zufallszahlenpaar (z_1, z_2) vorzudefinieren. Dabei nimmt die erste Zufallszahl z_1 den Summations-Koeffizienten c an, während sich die zweite Zufallszahl z_2 über die Koeffizienten x und y definiert. Beide Zufallszahlen liegen damit zwischen 0 und $q - 1$. Der Zusammenhang zwischen z und dem Zahlenpaar (z_1, z_2) ergibt sich über

$$z = z_1 + q \cdot z_2$$

Aus der Summengleichung von $|v_3>$ ergibt sich eine Zuordnung von z_1 und z_2 einerseits und den Summationsparametern c, x und y andererseits durch

$$\begin{aligned}
z < qY(X+1): \quad & z_1 = c \\
& z_2 = (X+1) \cdot y + x \\
& z = (X+1)q \cdot y + q \cdot x + c \\
z \geq qY(X+1): \quad & z_1 = c \\
& z_2 - Y(X+1) = X(y-Y) + x \\
& z_2 - Y = X \cdot y + x \\
& z = Xq \cdot y + q(x+Y) + c
\end{aligned}$$

Ziel des Verfahrens ist es, aus der jeweiligen Zufallszahl z die Koeffizienten c, x und y zu bestimmen, um diese in $\exp[i2\pi(xr+y)c/q]$ einzusetzen und um den Basis-Vektor $|c> |f(x)>$ zu ermitteln. Diese Summations-Koeffizienten errechnen sich aus z in folgender Weise:

$$\begin{aligned}
z < qY(X+1): \quad & c = z_1 = z \pmod q \\
& z_2 = \lfloor \tfrac{z}{q} \rfloor = (X+1) \cdot y + x \\
& y = \lfloor \tfrac{z_2}{X+1} \rfloor \\
& x = z_2 \pmod{(X+1)} \\
z \geq qY(X+1): \quad & c = z_1 = z \pmod q \\
& z_2 = \lfloor \tfrac{z}{q} \rfloor = X \cdot y + x + Y \\
& y = \lfloor \tfrac{z_2 - Y}{X} \rfloor \\
& x = (z_2 - Y) \pmod X
\end{aligned}$$

Aus den Summations-Koeffizienten werden die Summanden des TSVs ermittelt und $|v_{3red}>$ zugeführt. Aus dieser reduzierten Summe werden die reduzierten Basis-Wahrscheinlichkeiten und daraus die reduzierten Argument-Wahrscheinlichkeiten ermittelt.

Interessant an dem hier vorgestellten TSA mit Kenntnis der Ordnung r ist die Unabhängigkeit von der Zahl a. Dies liegt daran, daß $f(y)$ nicht bekannt sein muß. Dieser Funktionswert definiert in der Summengleichung von $|v_3>$ den Basisvektors eines jeden Summanden. Für die Argument-Wahrscheinlichkeit ist aber nur deren Phasenkoeffizient von Interesse.

5 Die SW-Realisierung

In diesem Kapitel werden alle SW-Programme vorgestellt, die zur Entwicklung des stochastischen Algorithmus beitragen.

5.1 Hilfsprogramme

In diesem Abschnitt werden Programme vorgestellt, die beim stochastischen Algorithmus unterstützend helfen. Hierzu zählen u.a. das Kettenbruch-Verfahren und auch die Zufallsgeneratoren zur Ermittlung von $s(m)$ aus q^2 Zahlen.

5.1.1 ContFrac: Das Kettenbruch-Verfahren

Das Programm „ContFrac“ ermittelt für einen Bruch $\frac{c}{q}$ mittels Kettenbruch-Verfahren einen reduzierten Bruch $\frac{d}{r}$, der der Ungleichung

$$|\tfrac{c}{q} - \tfrac{d}{r}| \;\leq\; \tfrac{1}{2q}$$

genügt. Siehe hierzu auch die Herleitung von Gl.(73) in Kapitel 4.2.1. Dabei werden folgende Schritte durchgeführt:

- Zum Bruch $\frac{c}{q}$ wird mittels Kettenbruch-Vorwärtszerlegung (siehe Kapitel 3.7) ein Kettenbruch-Vektor $[a_0, a_1, \cdots a_N]$ ermittelt.
- Mit einer minimalen Anzahl an Elementen aus dem Kettenbruch-Vektor, beginnend bei $[a_0]$, wird über Kettenbruch-Rückwärtszerlegung (siehe Kapitel 3.7) ein Bruch $\frac{d}{r}$ ermittelt, der Gl.(73) genügt.

Ein Beispiel zur Kettenbruch-Zerlegung findet sich in Beispiel 4 ($n = 57$, $a = 4$) von Kapitel 4.2.6.

5.1.2 RandFunc: Erzeugung von Zufallswerten

Das Programm „RandFunc“ ermittelt eine Menge an Zufallswerten. Hierzu werden zwei Input-Werte benötigt:

- N: Diese Zahl gibt an, daß eine Menge an Zufahlszahlen aus dem Intervall $[0, N-1]$ ausgewählt werden soll. Dabei muß $N \leq 131071 = 2^{17} - 1$ gelten.
- s: Dieser Wert bedeutet die Anzahl an unterschiedlichen Zufallszahlen, die ausgewählt werden sollen. Hierbei muß $s < N$ gelten.

Beispiel: Wird $N = 10$ und $s = 5$ gewählt, dann wählt die Zufallsgenerator-Funktion 5 beliebige Werte zwischen 0 und 9, z.B. 2, 4, 5, 8 und 9.

Wie werden diese Zufallswerte erzeugt? Mit der Funktion

$$rand()$$

aus der C-Library “stdlib.h“ werden ganzzahlige Zufallszahlen zwischen 0 und $RAND\_MAX = 2^{15} - 1 = 32767$ erzeugt. Um mit Hilfe dieser Funktion beliebige Zufallszahlen zwischen 0 und $N-1$ zu generieren, genügt es nicht, diese Zufallszahlen x über die Formel

$$x \;=\; (\tfrac{rand()\cdot N}{32768})(\mathrm{mod}\ N)$$

zu ermittelt. Der Grund hierfür ist, daß z.B im Falle von $N = 2 \cdot 32768 = 65536$ nur gerade x, also die Zahlen $x = 0, 2, 4, \ldots$, als Zufallszahlen zur Auswahl stehen. Im Fall von $N = 3 \cdot 32768 = 98304$ stehen nur die Zahlen $x = 0, 3, 6, \ldots$ zur Verfügung. Aus diesem Grund wird ein Modulo-Faktor z eingeführt, der folgendermaßen hergeleitet wird:

$$z \;=\; \begin{cases} 0 & \text{für} \quad 1 \leq N \leq 1 \cdot 32768 \\ 2 & \text{für} \quad 32768 + 1 \leq N \leq 2 \cdot 32768 \\ 3 & \text{für} \quad 2 \cdot 32768 + 1 \leq N \leq 3 \cdot 32768 \\ 4 & \text{für} \quad 3 \cdot 32768 + 1 \leq N \leq 4 \cdot 32768 \end{cases}$$

Mit Hilfe dieses z werden die Lücken für die Zufallswerte x aufgefüllt, indem die Zufallsfunktion $rand()$ noch ein zweites Mal aufgerufen wird.

$$x \;=\; [\tfrac{rand()\cdot N}{32768} + rand()(\mathrm{mod}\ z)](\mathrm{mod}\ N)$$

In dieser Formel muß zunächst der Zufallswert, der aus $rand()$ hervorgeht, mit N multipliziert werden. Dieses Produkt darf nicht die Grenze eines 32-Bit-Rechners überschreiten, die bei $2^{32} - 1$ liegt. Im maximalen Fall gilt $N \cdot 32768 \leq 2^{32} - 1$, wodurch gezeigt ist, warum $N \leq 2^{17} - 1 = 131071$ gelten muß.

Nachdem alle Zufallszahlen ermittelt worden sind, werden sie in aufsteigender Reihenfolge sortiert

Das Programm „RandFunc“ wird benötigt, um im stochastischen Algorithmus des Shor-Verfahrens aus einer Menge an $N = q^2$ Elementen der Gesamt-IFT-Matrix eine Anzahl an s Zufalls-Elementen zu ermitteln. Siehe hierzu das Programm „ShorRand“.

„RandFunc“ schränkt den stochastischen Shor-Algorithmus allerdings in folgender Weise ein: Da q^2 durch $N \leq 2^{17} - 1$ begrenzt wird, gilt $q^2 \leq 2^{16}$ und folglich $q \leq 2^8 = 256$. Mit $n^2 \leq q$ kann „RandFunc“ daher nur für stochastische Algorithmen mit $n \leq 15$ eingesetzt werden.

5.1.3 RandPair: Erzeugung von Zufallspaaren

Um die Bandbreite der Zufallszahlen auch für $n > 16$ nutzbar zu machen, wurde das Programm „RandFunc“ auf Paare von Zufallszahlen erweitert, wodurch das Programm „RandPair“ entstand. Hierbei werden folgende Input-Parameter benötigt:

- N: Diese Zahl gibt an, daß aus einer Menge an Paaren (x,y) von Zufallszahlen $x \in [0, N-1]$ und $y \in [0, N-1]$ ausgewählt wird. Auch hier ist $N \leq 131071 = 2^{17} - 1$. Insgesamt stehen durch dieses Verfahren N^2 Zahlenpaare zur Verfügung.
- s: Dieser Wert bedeutet die Anzahl an unterschiedlichen und zufälligen Zahlenpaaren, die ausgewählt werden. Hier darf $s < N^2$ sein.

Beispiel: Wird $N = 3$ und $s = 4$ gewählt, dann wählt die Zufallsgenerator-Funktion $rand()$ 4 beliebige Zahlenpaare zwischen $(0, 0)$ und $(2, 2)$ aus, z.B. $(0, 0)$, $(0, 2)$, $(1, 1)$ und $(2, 0)$.

Die Ermittlung der Zufallszahlen x und y für die Zahlenpaare (x, y) erfolgt in analoger Weise wie in „RandFunc“.

Diese Zahlenpaare werden in aufsteigender Reihenfolge sortiert, d.h.

$$(x, y) < (x, y + 1) < (x + 1, 0)$$

Auf diese Art ist es nun möglich, jedem der relevanten $N^2 = q^2$ Elemente der Gesamt-IFT-Matrix des Shor-Algorithmus ein Zahlenpaar zuzuordnen. Die zufällige Auswahl von s Zahlenpaaren entscheidet über die Wahl der Parameter der Gesamt-IFT-Matrix. Da q hier durch $N \leq 2^{17}-1$ begrenzt wird, gilt $q \leq 2^{16}$. Mit $n^2 \leq q$ kann „RandPair“ daher für stochastische Algorithmen mit $n \leq 2^8 = 256$ eingesetzt werden. Siehe hierzu das Programm „ShorPair“.

5.2 Nebenaspekte

In diesem Abschnitt wird ein Programm vorgestellt, das die Dimensionierung des stochastischen Algorithmus des Shor-Verfahrens betrifft.

5.2.1 RecoExpo: Empfehlungen für s aus q^2

Die Funktion „RecoExpo“ (Kurzname für „Recommended Exponent“) nimmt Bezug auf Gl.(101) und Gl.(102). Sie ermittelt

- für jedes n die Dimensionen m und q über Gl.(54).
- eine Darstellung von q^2 in Form von $k_s \cdot m^{p_s}$ mit $k_s \cdot m^{p_s} \leq q^2$.

Somit erhält man:

$$q^2 \;\geq\; s(m) = k_s \cdot m^{p_s} \quad\Longrightarrow\quad 2\log_m(q) \geq \log_m(k_s) + p_s \quad\wedge\quad k_s \;\leq\; \tfrac{q^2}{m^{p_s}}$$

Mit $1 \leq k_s < m$ gilt $0 \leq \log_m(k_s) < 1$. Damit kann p_s abgerundet werden und man erhält:

$$p_s \;=\; \lfloor 2\log_m(q) \rfloor \quad\wedge\quad k_s \;=\; \lfloor \tfrac{q^2}{m^{p_s}} \rfloor \;=\; \lfloor (\tfrac{q}{m^{\frac{p_s}{2}}})^2 \rfloor$$

Die Umformung von k_s in einen Term, der nur von q und nicht von q^2 abhängt, begünstigt die Berechnung für einen 32-Bit-Rechner, da hierdurch auch Werte für $n > 181$ hinzugezogen werden können, für die $q^2 \geq 2^{15}$ oberhalb der Rechnergrenzen liegt.

Der Vergleich von $s(m) = km^p$ mit $q^2 \approx k_s m^{p_s}$ ist für die Bewertung des stochastischen Shor-Algorithmus in Bezug auf polynomiale und exponentielle Aufwände von Interesse.
Interessant ist dabei die Frage, ob es für jedes n eine Zahl $p < p_s$ gibt, bei der die Anzahl von $k \cdot m^p$ Pfaden eines TSAs genügt, um die Ordnung r zu berechnen. Kann man hierzu ein Bildungsgesetz für dieses p in Abhängigkeit von n ableiten? Dazu ein Beispiel:

Für $n = 33$ ist $m = 11$, $q = 2048$ und $q^2 = 4194304$. Das Polynom $s \leq q^2$, das q^2 am nächsten liegt ist $s = 2m^6 = 3543122$. Die Frage lautet hier: Reichen anstelle von $2m^6$ EGPn eine Anzahl von m^4 oder m^5 EGPn zur Berechnung der Ordnung r?

Tabelle 13 stellt die Polynome $s(m) \leq q^2$ dar, die q^2 am nächsten sind.

n	m	q	q^2	$s(m) =$	$k_s \cdot m^{p_s}$
1	0	1	1	$1 =$	$1 \cdot 0^0$
2	2	4	16	$16 =$	$1 \cdot 2^4$
$3-4$	4	16	256	$256 =$	$1 \cdot 4^4$
5	5	32	1024	$625 =$	$1 \cdot 5^4$
$6-8$	6	64	4096	$3888 =$	$3 \cdot 6^4$
$9-11$	7	128	16384	$14406 =$	$6 \cdot 7^4$
$12-16$	8	256	65536	$65536 =$	$2 \cdot 8^5$
$17-22$	9	512	262144	$236196 =$	$4 \cdot 9^5$
$23-32$	10	1024	1048576	$1000000 =$	$1 \cdot 10^6$
$33-45$	11	2048	4194304	$3543122 =$	$2 \cdot 11^6$
$46-64$	12	4096	16777216	$14929920 =$	$5 \cdot 12^6$
$65-91$	13	8192	67108864	$62748517 =$	$1 \cdot 13^7$
$91-128$	14	16384	268435456	$210827008 =$	$2 \cdot 14^7$
$129-181$	15	32768	1073741824	$1025156250 =$	$6 \cdot 15^7$
$182-256$	16	65536	4294967296	$4294967296 =$	$1 \cdot 16^8$
$257-362$	17	131072			$2 \cdot 17^8$
$363-512$	18	262144			$6 \cdot 18^8$
$513-724$	19	524288			$16 \cdot 19^8$
$725-1024$	20	1048576			$2 \cdot 20^9$
$1025-1448$	21	2097152			$5 \cdot 21^9$
$1449-2048$	22	4194304			$14 \cdot 22^9$

Tabelle 13: RecoExpo: Empfehlungen für $s(m) \leq q^2$

5.3 Programme für das Verfahren nach G. L. Miller

Dieses Kapitel beschreibt diejenigen SW-Programme, die Aufschluß über das Verfahren von G. L. Miller geben.

5.3.1 ShorEasy: Die einfache Faktorisierung nach G. L. Miller

Das Programm „ShorEasy“ stellt den Faktorisierungsalgorithmus von G. L. Miller dar, so wie er in Kapitel 2.2 beschrieben ist. Zu einem gewählten n und a werden folgende Schritte durchlaufen:

- Berechnung und Ausgabe der Ordnung $r = \mathrm{ord}_n a$.
- Berechnung und Ausgabe der Faktoren von n mit Hilfe des ggT, falls r der Bedingung in Gl.(13) genügt.
- Darstellung des Abbruchkriteriums, falls r einer der beiden Bedingungen in Gl.(13) nicht genügt.

5.3.2 ShorRate: Die Erfolgsrate des Verfahrens von G. L. Miller

Das Programm „ShorRate“ ist eine Anwendung von „ShorEasy“. Hierbei werden folgende Schritte durchlaufen:

- Für ein gewähltes n werden alle $a \in \mathbb{Z}_n^*$ mit $\mathbb{Z}_n^*$ gemäß Gl.(2) ermittelt.
- Für diese a wird das Programm „ShorEasy“ hintereinander aufgerufen. Die Ergebnisse werden ausgegeben, und die erfolgreichen Faktorisierungen werden gezählt.
- Zum Schluß wird die Anzahl der erfolgreichen Faktorisierungen zusammen mit der Gesamtzahl der über a durchgeführten Faktorisierungen ausgegeben. Einige Beipiele hierzu werden in den Tabellen 8 und 9 dargestellt.

Die Ergebnisse von „ShorRate“ eignen sich besonders zur Auswahl eines entsprechenden Testfalls für die späteren Anwendungen des Shor-Verfahrens unter Berücksichtigung der zu erwartenden Ordnung r.

5.3.3 ShorRatePlus: Die Erfolgsrate mit Zusatzinformationen

Das Programm „ShorRatePlus“ enthält neben den Eigenschaften des Programms von „ShorRate“ noch zusätzliche Informationen:

- Für jeden Testfall in a wird neben der Ordnung r das maximale δ, für das $2^\delta \mid r$ gilt, berechnet. (Siehe [1].)
- Die Faktoren von n werden in Form von $n_k = p_k^{\alpha_k}$ dargestellt.
- Zu jedem n_k werden die Werte a_k mit Hilfe des „Chinesische Resttheorems“ über die Gleichung $a = a_k \pmod{n_k}$ ermittelt.
- Über $a_k^{r_k} = 1 \pmod{n_k}$ wird die Ordnung r_k ermittelt und daraus das maximale δ_k, für das $2^{\delta_k} \mid r_k$ gilt.

Zu diesen Parametern existiert eine Beschreibung in Anhang 4 von [NI]. Sie werden in den erweiterten Ausführungen von Kapitel 8.1 benötigt.

5.4 ShorPure: Das vollständige Shor-Verfahren als SW-Algorithmus

In diesem Kapitel werden mehrere Möglichkeiten vorgestellt, wie das vollständige Shor-Verfahren als SW-Programm programmiert werden kann.

[1] In Abweichung von [NI] wird der Wert d in dieser Arbeit durch δ dargestellt, da d schon in der Kettenbruch-Zerlegung verwendet wird.

5.4.1 ShorPure: Das Shor-Verfahren als Vierphasen-Modell

Das SW-Programm „ShorPure“ erzeugt das Shor-Verfahren als Vierphasen-Modell auf einem klassischen Rechner. In Anlehnung an Kapitel 4 werden folgende Schritte durchlaufen:

1. Neben den Inputwerten n und a wird gefragt, ob einerseits eine lange Ergebnisdatei mit allen Zwischenergebnisse ($\underline{v}_0$ bis $\underline{v}_3$) und allen Argument-Wahrscheinlichkeiten, die größer als 10^{-4} sind, oder ob andererseits nur eine kurze Ergebnisdatei mit denjenigen relevanten Argument-Wahrscheinlichkeiten, die Gl.(78) erfüllen, gewünscht wird.

2. Die Dimensionen m und q des Argument-Hilbertraums werden gemäß Gl.(54) bestimmt.

3. Die Vektoren $\underline{v}_0$ bis $\underline{v}_3$ werden gemäß Kapitel 4.3 bestimmt. Dabei brauchen die Vektoren $\underline{v}_0$ und $\underline{v}_1$ nicht berechnet zu werden, da sie sich direkt aus den Gleichungen Gl.(82) und Gl.(84) ergeben.

4. Der Vektor $\underline{v}_2$ wird gemäß Gl.(87) bestimmt. Hierzu genügt es, einen Hilfsvektor $\underline{v}_{2H}$ der Dimension q zu erzeugen, der nur die Argumente x der Basisvektoren $\underline{e}_x$ von $\underline{v}_2$ ablegt.

   Beispiel: Für das vereinfachte Beispiel 1 ($n = 3$, $a = 2$, $q = 4$) in Kapitel 4.3.2 sehen die Vektoren $\underline{v}_2$ und $\underline{v}_{2H}$ folgendermaßen aus:

   $$\underline{v}_2 \;=\; \tfrac{1}{2}\{\underline{e}_5 + \underline{e}_7 + \underline{e}_{10} + \underline{e}_{12}\} \;\Longrightarrow\; \underline{v}_{2H} = [5, 7, 10, 12]^T$$

5. Der Vektor $\underline{v}_3$ wird gemäß Gl.(91) bestimmt. Hierbei gilt:

   $$\underline{v}_3 \;=\; \underline{\underline{M}}_{IFT} \cdot \underline{v}_2 \;=\; \sum \lambda_\nu \cdot \underline{e}_\nu$$

   Die Werte λ_ν gehen als Basis-Wahrscheinlichkeit $|\lambda_\nu|^2$ über Gl.(94) in die Argument-Wahrscheinlichkeit $P(|c>)$ ein. Alle q Argument-Wahrscheinlichkeiten werden in einem Hilfsvektor $\underline{v}_{3H}$ gesammelt.

6. Aus dem Hilfsvektor $\underline{v}_{3H}$ werden aus q Argument-Wahrscheinlichkeiten diejenigen r Werte von $P(|c>)$ ermittelt, für die Gl.(78) gilt. Der kleineste von Null verschiedene c-Wert wird über das Kettenbruch-Verfahren zur Ermittlung der Ordnung r hergenommen.

7. Die weitere Berechnung der Faktoren von n über das ggT erfolgt wie im Programm „ShorEasy“.

Im folgenden wird in einigen Beispielen die Verteilung der relevanten Argument-Wahrscheinlichkeiten, d.h. derjenigen $P(|c>)$, die Gl.(78) erfüllen, graphisch dargestellt. Die c-Werte, die über Kettenbruch-Verfahren zu einer korrekten Ordnung r führen, werden **fett** unterlegt.

Interessant ist hierbei, daß die relative Verteilung der Wahrscheinlichkeiten für gleiche Ordnungen r immer gleich bleibt. „Relativ“ hat hier folgende Bedeutung:

- Die Zahlenwerte c der relevanten $P(|c>)$ stehen immer in einem bestimmten Verhältnis zur Gesamt-Dimension q.

- Die Hierarchie der relevanten Zahlenwerte $P(|c>)$ ist örtlich festgelegt. Das bedeutet, daß die größten, die zweitgrößten, die drittgrößten, usw. $P(|c>)$ immer an der (relativ) gleichen Stelle stehen.
- Die Zahlenwerte der größten, zweitgrößten, usw. $P(|c>)$ sind bis auf geringfügige Abweichungen immer gleich groß.

Beispiel 1: Sei $n = 15$ und $a = 7$. Dann gilt $m = 8$, $q = 256$ und $r = 4$.

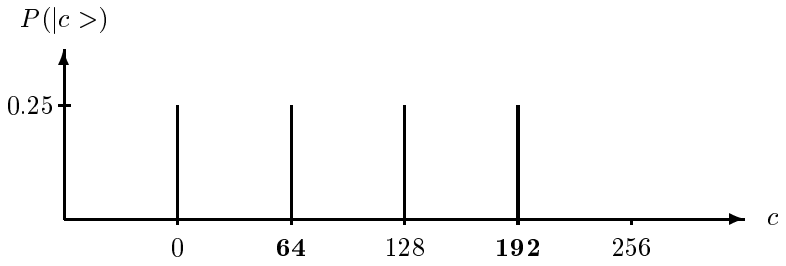

Abbildung 26: Argument-Wahrscheinlichkeiten für $n = 15$, $a = 7$ und $r = 4$.

Folgende Erkenntnisse können aus Abbildung 26 gezogen und für alle Verfahren mit Ergebnis $r = 4$ verallgemeinert werden:

- Die einzigen Argument-Wahrscheinlichkeiten liegen mit $P = 0.25$ bei $c = 0$, $c = \frac{q}{4}$, $c = \frac{q}{2}$ und $c = \frac{3q}{4}$.
- Davon führen nur die Werte $c = \frac{q}{4}$ und $c = \frac{3q}{4}$ mittels Kettenbruch-Zerlegung zur korrekten Ordnung $r = 4$.

Zusatzbemerkung:
Dieses Beispiel stellt einen der Idealfälle dar, die in Kapitel 4.2.2 schon erwähnt worden sind. Alle Idealfälle mit $r = 2^{\mu}$ haben ähnliche Verläufe wie der in Beispiel 1 dargestellte.

Beispiel 2: Sei $n = 21$ und $a = 2$. Dann gilt $m = 9$, $q = 512$ und $r = 6$.

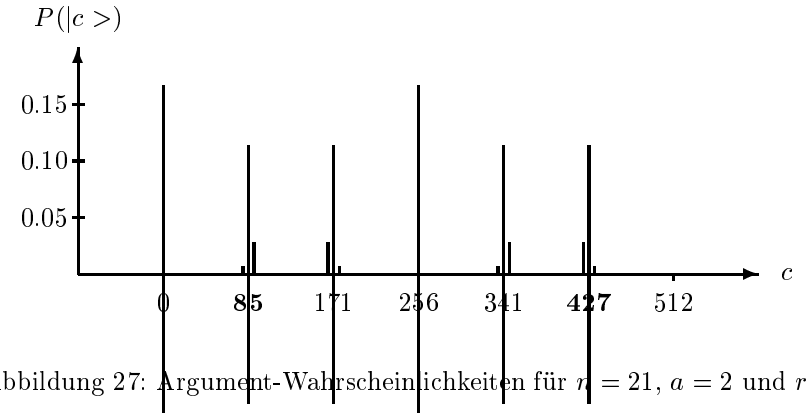

Abbildung 27: Argument-Wahrscheinlichkeiten für $n = 21$, $a = 2$ und $r = 6$.

Folgende Erkenntnisse können aus Abbildung 27 gezogen und für alle Verfahren mit Ergebnis $r = 6$ verallgemeinert werden:

- Die höchsten Argument-Wahrscheinlichkeiten liegen mit $P = 0.166$ bei $c = 0$ und $c = \frac{q}{2}$, allerdings führen beide c-Werte über Kettenbruch-Zerlegung nicht zur korrekten Ordnung r.
- Von den anderen vier relevanten Argument-Wahrscheinlichkeiten mit $P = 0.114$ führen nur der erste ($d = 1$) und der letzte c-Wert ($d = 5$) über Kettenbruch-Zerlegung zur korrekten Ordnung $r = 6$.
- Die kleinsten relevanten Argument-Wahrscheinlichkeiten mit $P = 0.114$ sind um den Faktor 4 deutlich größer als die größten nicht-relevanten Argument-Wahrscheinlichkeiten mit $P = 0.0285$.

Beispiel 3: Sei $n = 33$ und $a = 5$. Dann gilt $m = 11$, $q = 2048$ und $r = 10$.

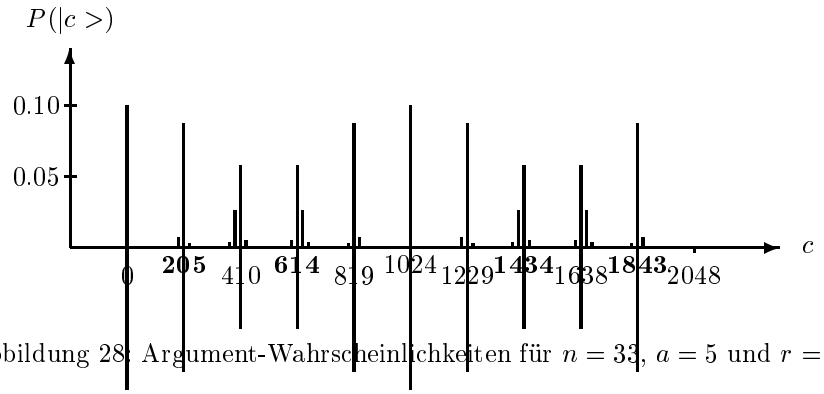

Abbildung 28: Argument-Wahrscheinlichkeiten für $n = 33$, $a = 5$ und $r = 10$.

Folgende Erkenntnisse können aus Abbildung 28 gezogen und für alle Verfahren mit Ergebnis $r = 10$ verallgemeinert werden:

- Die höchsten Argument-Wahrscheinlichkeiten liegen mit $P = 0.1$ bei $c = 0$ und $c = \frac{q}{2}$, allerdings führen beide c-Werte über Kettenbruch-Zerlegung nicht zur korrekten Ordnung r.
- Von den nächsten vier Argument-Wahrscheinlichkeiten bei $P = 0.0875$ führen nur zwei c-Werte über Kettenbruch-Zerlegung zu $r = 10$.
- Von den anderen vier Argument-Wahrscheinlichkeiten bei $P = 0.0573$ führen auch nur zwei c-Werte über Kettenbruch-Zerlegung zu $r = 10$.
- Im Gegensatz zu $r = 6$ fallen bei $r = 10$ die Abstände zwischen kleinsten relevanten Argument-Wahrscheinlichkeiten bei $P = 0.0573$ und größten nicht-relevanten Argument-Wahrscheinlichkeiten bei $P = 0.0254$ über den Faktor 2 deutlich geringer aus.

Beispiel 4: Sei $n = 35$ und $a = 2$. Dann gilt $m = 11$, $q = 2048$ und $r = 12$.
Folgende Erkenntnisse können aus Abbildung 29 gezogen und für alle Verfahren mit Ergebnis $r = 12$ verallgemeinert werden:

- Die höchsten Argument-Wahrscheinlichkeiten liegen mit $P = 0.0833$ bei $c = 0$, $c = \frac{q}{4}$, $c = \frac{q}{2}$ und $c = \frac{3q}{4}$, allerdings führen alle vier c-Werte über Kettenbruch-Zerlegung nicht zur korrekten Ordnung r.

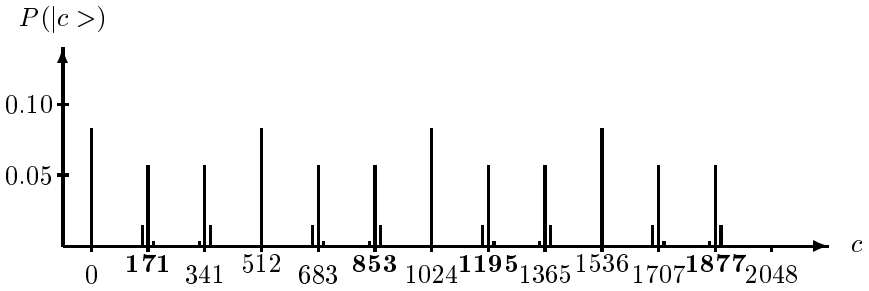

Abbildung 29: Argument-Wahrscheinlichkeiten für $n = 35$, $a = 2$ und $r = 12$.

- Von den nächsten acht Argument-Wahrscheinlichkeiten bei $P = 0.0569$ führen nur vier c-Werte über Kettenbruch-Zerlegung zu $r = 12$.
- Ähnlich wie bei $r = 6$ sind bei $r = 12$ die kleinsten relevanten Argument-Wahrscheinlichkeiten mit $P = 0.0570$ um den Faktor 4 deutlich größer als die größten nicht-relevanten Argument-Wahrscheinlichkeiten mit $P = 0.0142$.

5.4.2 ShorSped: Das Shor-Verfahren als schneller Algorithmus

Das Programm „ShorSped“ (Kurzname für „ShorSpeed “) bietet die Möglichkeit, die Ergebnisse des Shor-Verfahrens in wesentlich kürzerer Zeit zu ermitteln und verzichtet auf die Reihenfolge des Vierphasen-Modells.

Grundlage für diesen Algorithmus bildet Kapitel 4.2.3 und Gl.(70). Aus dieser Gleichung kann der Vektor $\underline{v}_3$ über die Transformationsgleichung Gl.(37) direkt berechnet werden. Die korrespondieren Basis- und Argument-Wahrscheinlichkeiten können daraus abgeleitet werden. Die Vektoren $\underline{v}_0$ bis $\underline{v}_2$ werden wie im Programm „ShorPure“ ohne großen Rechenaufwand dargestellt werden.

Einziger Nachteil dieses Verfahrens ist, daß neben n und a auch die Ordnung r als Eingangs-Parameter angegeben werden muß, da Gl.(70) dies verlangt. Diese kann allerdings über das Programm „ShorEasy“ vorher ermittelt werden.

Aufgabe dieses Programms ist es somit nicht, die Ordnung r zu ermitteln, sondern mit möglichst wenig zeitlichem Aufwand Auskunft über die Zwischenergebnisse des Shor-Verfahrens ($\underline{v}_0$ bis $\underline{v}_3$) und über die Basis- und Argument-Wahrscheinlichkeiten zu geben, sofern dies erwünscht ist. Ähnlich wie beim Programm „ShorPure“ besteht auch in diesem Programm die Möglichkeit, diese Zwischenergebnisse auszuschalten und stattdessen nur die relevanten Argument-Wahrscheinlichkeiten, die Gl.(78) erfüllen, anzeigen zu lassen.

5.5 Das Shor-Verfahren als stochastischer Algorithmus

In diesem Kapitel werden verschiedene Verfahren des stochastischen Algorithmus analysiert. Hierbei wird auf die in den Kapiteln 4.5.5 und 4.5.6 definierten Teilsummen-Algorithmen (**TSA**) zurückgegriffen.

5.5.1 ShorRand: Der Algorithmus mit einfachen Zufallswerten

Grundlage für das Programm „ShorRand“ ist das Verfahren zum Ermitteln von TSAs aus Kapitel 4.5.5. Es erstellt einen TSA mit Hilfe des Programms „RandFunc“. Hierbei werden folgende Schritte durchlaufen:

1. Abfrage der Inputwerte n und a: Bei diesen Werten darf n nicht größer als 15 sein, damit $q^2 \leq 2^{16}$ und somit innerhalb der gesetzten Grenzen bleibt, für die das Programm „RandFunc“ definiert ist.

2. Interne Berechnung der Dimensionen m und q des Argument-Hilbertraums gemäß Gl.(54).

3. Abfrage der Inputwerte k und p. Diese Inputwerte bilden zusammen mit m die Anzahl $s(m) = km^p$ der unterschiedlichen EGPn und somit die Anzahl der genutzten Elemente der Gesamt-IFT-Matrix für den TSA. Um die Wahl dieser Parameter zu erleichtern, werden die Werte k_s und p_s empfohlen, für die das Polynom $s_{max}(m) = k_s m_s^p$ als untere maximale Grenze q^2 am nächsten liegt. Diese Werte sind dem Programm „RecoExpo“ entnommen.

4. Berechnung der Vektoren $\underline{v}_0$, $\underline{v}_1$ und $\underline{v}_2$ wie im Programm „ShorPure“.

5. Ermittlung von $s(m)$ verschiedenen Zufallszahlen zwischen 0 und $q^2 - 1$ mit Hilfe des Programms „RandFunc“. Diese Zufallszahlen werden in aufsteigender Reihenfolge sortiert und in einem Hilfsvektor $\underline{v}_{Rand}$ abgelegt.

6. Bestimmung eines reduzierten $\underline{v}_{3red}$: In der Gesamt-IFT-Matrix werden alle q^2 relevanten Elemente, die mit $\underline{v}_2$ multipliziert werden, von links oben nach rechts unten durchnummeriert. Siehe hierzu auch das Beispiel in Kapitel 4.5.5. Bei der Berechnung des reduzierten $\underline{v}_{3red}$ wird vorgegangen wie im Programm „ShorPure“, allerdings mit der Ausnahme, daß nur diejenigen Elemente der Gesamt-IFT-Matrix in den TSA eingehen, die auch als nummeriertes Element im Hilfsvektor $\underline{v}_{Rand}$ vertreten sind.

7. Auswertung des reduzierten $\underline{v}_{3red}$: Ähnlich wie beim Programm „ShorPure“ gehen die reduzierten Basisvektoren über Gl.(94) in die reduzierten Argument-Wahrscheinlichkeiten $P_{red}(|c>)$ ein.

8. Auswertung der reduzierten Argument-Wahrscheinlichkeiten:
   Aus den maximalen Argument-Wahrscheinlichkeiten werden die ersten $3m$ Werte in die engere Wahl gezogen und ausgegeben. Diese Werte können nun folgenden Fragen unterzogen werden:
   Ergibt sich aus den reduzierten Argument-Wahrscheinlichkeiten mindestens ein c-Wert, der über das Kettenbruch-Verfahren zur korrekten Ordnung r führt?
   Wieviele c-Werte befinden sich unter den ersten reduzierten Argument-Wahrscheinlichkeiten, die auch in den r relevanten Argument-Wahrscheinlichkeiten von „ShorPure“ enthalten sind?

Ein Beispiel zu diesem Programm findet sich in Kapitel 4.5.5.

5.5.2 ShorPair: Der Algorithmus erweitert mit Zufallspaaren

Da das Programm „RandFunc“ das Programm „ShorRand“ auf $n = 16$ einschränkt, benutzt das Programm „ShorPair“ das erweiterte Zufallsprogramm „RandPair“ zur Generierung der Zufallszahlen.

Wie bereits is Kapitel 4.5.5 geschildert, ist dieses Programm besonders für Werte $n \geq 133$ sehr aufwendig, da neben der Ermittlung von Zufallspaaren auch noch das Sortieren dieser Paare erforderlich ist. Auf einem modernen MS-XP-Rechner (Baujahr 2005) wurde für $n = 133$, $a = 8$ und einer Anzahl gewünschter EGPn von $s(m = 15) = 2 * 15^5 = 1518750$ eine Rechenzeit von 13 Stunden und 57 Minuten registriert, von denen fast 10 Stunden für das Sortieren der Zufallspaare aufgebracht wurden.

5.5.3 ShorShort: Der verkürzte Stochastische Algorithmus

Um auf das Sortieren von Zufallspaaren verzichten zu können, realisiert das Programm „ShorShort“ das Verfahren von Kapitel 4.5.6, bei dem die Ordnung r als bekannt vorausgesetzt wird.

Mit Hilfe des Programms „RandPair“ werden zunächst Zufallspaare vom Typ (z_1, z_2) erzeugt. Beide Zufallsparameter, z_1 und z_2, liegen im Intervall von 0 bis $q - 1$. Der Summenparameter c in Gl.(70) ergibt sich dabei direkt aus z_1, während die beiden anderen Parameter x und y aus z_2 berechnet werden. Der Algorithmus lautet folgendermaßen:

$$
\begin{aligned}
z_2 < Y(X+1): \quad & c = z_1 \\
& z_2 = (X+1) \cdot y + x \\
& y = \lfloor \tfrac{z_2}{X+1} \rfloor \\
& x = z_2 \pmod{(X+1)} \\
z_2 \geq Y(X+1): \quad & c = z_1 \\
& z_3 = z_2 - Y(X+1) = X \cdot (y - Y) + x \\
& y = Y + \lfloor \tfrac{z_3}{X} \rfloor \\
& x = z_3 (\mathrm{mod}\ X)
\end{aligned}
$$

Im Gegensatz zu „ShorRand“ benötigt das Programm „ShorShort“ für $n = 133$, $a = 8$ und einer Anzahl gewünschter EGPn von $s(m = 15) = 2 * 15^5 = 1518750$ nur 4 Stunden und 9 Minuten, allerdings werden bei einer Verdopplung der Anzahl auf $s(m = 15) = 4 * 15^5 = 3037500$ EGPn 14 Stunden und 53 Minuten benötigt.

5.5.4 ShorShorts: ShorShort als Reihentest

Das Programm „ShorShorts“ bietet die Möglichkeit, mehrere Tests vom Typ „ShorShort“ direkt hintereinander durchzuführen. Dabei werden die Eingangsparameter für jeden Test in einer Sammeleingangsliste eingegeben. Das Programm ruft daraufhin jeden Test nacheinander auf und legt die Testergebnisse in einzelnen Dateien ab, deren Namen auch vorher in der Sammeleingangsliste festgelegt worden sind.

5.5.5 ShorEval: Auswertung von Shor-Algorithmen

Das Programm „ShorEval“ bietet für eine oder mehrere Ergebnisdateien der Programme „ShorRand“ oder „ShorShort“ die Möglichkeit, die Ergebnisse elektronisch auszuwerten. Dabei wird folgendermaßen vorgegangen:

- Zunächst muß sichergestellt werden, daß alle Ergebnisdateien die gleichen m- und r-Werte haben. Nur so ist sichergestellt, daß für alle Tests die gleichen relevanten c-Werte berechnet worden sind.
- In einer Eingangsdatei werden die Ordnung r, die relevanten c-Werte und danach die Namen aller Ergebnisdateien aus den Programmen „ShorRand“ oder „ShorShort“ eingetragen.
- Das Programm durchsucht die Tabellen dieser Ergebnisdateien, die die besten $3m$ Argument-Wahrscheinlichkeiten enthalten, nach den relevanten c-Werten und gibt diese sowie deren Position in der Tabelle aus.
- Die Ausgangsdatei enthält zu jeder Ergebnisdatei diejenigen c-Werte, die in der jeweiligen Tabelle aufgeführt worden sind, sowie deren Position.

Das Programm „ShorEval“ ist sehr nützlich für die Auswertung der Tests in Kapitel 6.

6 Versuchsreihen und Ergebnisauswertung

6.1 Die Frage

In diesem Kapitel wird anhand von Testreihen des stochastischen Algorithmus des Shor-Verfahrens folgende Frage untersucht: Gibt es für alle n bzgl. der Anzahl

$$s(m) = k \cdot m^p \quad \text{mit} \quad p \leq p_{max} \tag{104}$$

der EGPn einen maximalen Potenzwert p_{max}? Um diese Frage zu beantworten, müssen einige Bewertungskriterien definiert werden. Diese sollen am folgenden Beispiel gefunden werden.

6.2 Bewertungskriterien für den einzelnen Shor-Algorithmus

Für die folgenden Überlegungen wird das Beispiel mit $n = 91$ und $a = 4$ verwendet. Mit Hilfe von $m = 14$ und $q = 2^m = 16384$ errechnet der komplette Shor-Algorithmus die Ordnung $r = 6$ und insgesamt $r = 6$ relevante Argument-Wahrscheinlichkeiten $P(|c>)$:

	c	$P(\|c>)$	d	r
0.	0	0.166667	0	0
1.	**2731**	0.113986	1	**6**
2.	5461	0.113986	1	3
3.	8192	0.166667	1	2
4.	10923	0.113986	2	3
5.	**13653**	0.113986	5	**6**
	2730	0.028497		
	5462	0.028497		
	10922	0.028497		
	13654	0.028497		

Tabelle 14: Relevante Argument-Wahrscheinlichkeiten des Shor-Verfahrens für $n = 91$, $a = 4$ und $r = 6$

Unterhalb der $r = 6$ relevanten c-Werte werden in Tabelle 14 die nächstgroßen nicht-relevanten c-Werte mit den zugehörigen Argument-Wahrscheinlichkeiten dargestellt. Relevante und nicht-relevante c-Werte werden bekanntlich über Gl.(78) definiert. Folgende Dinge fallen auf:

- Von den relevanten c-Werten führen nur die Werte $c = 2731$ und $c = 13653$ zur korrekten Ordnung r. Allerdings sind deren Argument-Wahrscheinlichkeiten nicht die größten, diese liegen bei $c = 0$ und $c = 8192$.

- Die relevanten c-Werte setzen sich von den nächstgroßen nicht-relevanten Werten um den Faktor 4 deutlich ab. Dies ist eine besondere Eigenschaft für Shor-Verfahren mit Ordnung $r = 6$.

Bekanntlich ist aus Sicht des Shor-Verfahrens die Berechnung der Ordnung r und damit das Ermitteln der relevanten Argument-Wahrscheinlichkeiten das Ziel. Die Zerlegung von n in Faktoren ist hier nicht von Interesse. Um einen reduzierten Shor-Algorithmus bewerten zu können, wird untersucht,

- welche reduzierten Argument-Wahrscheinlichkeiten die relevanten c-Werte im Vergleich zu allen anderen c-Werten einnehmen.
- wie groß die Anzahl $s(m)$ der EGPn mindestens sein muß, damit sich relevante c-Werte von nicht-relevanten absetzen.

Im Beispiel mit $n = 91$ und $a = 4$ benötigt der komplette Shor-Algorithmus $q^2 = 2^{28}$, also insgesamt 268435456 EGPn. Das einfache Polynom, das dieser Zahl von unten am nächsten kommt, ist $s(m) = 2m^7 = 210827008$. Als nächstes werden vier Tests des TSAs untersucht.

Test 1: Die verringerte Anzahl an EGPn beträgt in diesem TSA $s(m) = 2m^4$. Tabelle 15 sortiert die reduzierten Argument-Wahrscheinlichkeiten $P_{red}[|c>]$ nach Größe und zeigt davon die erstbesten $3r = 18$.

	c	$\frac{P_{red}(\|c>)}{10^{-7}}$		c	$\frac{P_{red}(\|c>)}{10^{-7}}$		c	$\frac{P_{red}(\|c>)}{10^{-7}}$
1.	7166	1.009	7.	3688	0.827	13.	6265	0.738
2.	5135	0.922	8.	4500	0.791	14.	14198	0.730
3.	641	0.916	9.	15736	0.776	15.	409	0.725
4.	9648	0.878	10.	6379	0.774	16.	15522	0.720
5.	12963	0.847	11.	9611	0.770	17.	6621	0.711
6.	10207	0.843	12.	10327	0.754	18.	11394	0.698

Tabelle 15: Reduzierter Shor-Algorithmus mit $n = 91$, $a = 4$ und $s(m) = 2m^4$

Auswertung:
Keiner der 6 relevanten c-Werte aus Tabelle 14 befindet sich unter den ersten 18 Werten. Deshalb muß die Anzahl $s(m)$ der EGPn erhöht werden, damit relevante Argument-Wahrscheinlichkeiten in der Wahrscheinlichkeitstabelle der erstbesten 18 Werte auftauchen.

Test 2: Die verringerte Anzahl an EGPn beträgt in diesem TSA $s(m) = 7m^4$. Tabelle 16 sortiert die reduzierten Argument-Wahrscheinlichkeiten $P_{red}[|c>]$

nach Größe und zeigt davon die erstbesten $3r = 18$.

	c	$\frac{P_{red}(\|c>)}{10^{-7}}$		c	$\frac{P_{red}(\|c>)}{10^{-7}}$		c	$\frac{P_{red}(\|c>)}{10^{-7}}$
1.	**0**	4.247	7.	2311	2.386	13.	1868	2.187
2.	**13653**	3.252	8.	1318	2.318	14.	9176	2.163
3.	**5461**	2.949	9.	7058	2.263	15.	10862	2.153
4.	4093	2.720	10.	**8192**	2.235	16.	7978	2.111
5.	13303	2.416	11.	12145	2.227	17.	14262	2.103
6.	12444	2.400	12.	10412	2.192	18.	3690	2.032

Tabelle 16: Reduzierter Shor-Algorithmus mit $n = 91$, $a = 4$ und $s(m) = 7m^4$

Auswertung:
In Tabelle 16 sind relevanten c-Werte, soweit sie vorkommen, fett unterlegt. Dies sind an vorderster Stelle die Werte $c = 0$, $c = 5461$ und $c = 13653$ und, im weiteren Verlauf der Tabelle, der Wert $c = 8192$ an Position 10.

Test 3: Die verringerte Anzahl an EGPn beträgt in diesem TSA $s(m) = 1m^5$. Tabelle 17 sortiert die reduzierten Argument-Wahrscheinlichkeiten $P_{red}[|c>]$ nach Größe und zeigt davon die erstbesten $3r = 18$.

	c	$\frac{P_{red}(\|c>)}{10^{-7}}$		c	$\frac{P_{red}(\|c>)}{10^{-7}}$		c	$\frac{P_{red}(\|c>)}{10^{-7}}$
1.	**13653**	6.342	7.	15208	4.936	13.	10379	4.063
2.	**5461**	6.224	8.	14049	4.549	14.	1030	4.059
3.	**2731**	6.013	9.	8703	4.391	15.	16341	4.026
4.	**8192**	5.811	10.	7284	4.162	16.	15183	4.025
5.	**0**	5.327	11.	2856	4.083	17.	10593	3.992
6.	**10923**	5.032	12.	3780	4.082	18.	6222	3.988

Tabelle 17: Reduzierter Shor-Algorithmus mit $n = 91$, $a = 4$ und $s(m) = 1m^5$

Auswertung:
In Tabelle 17 finden sich alle sechs relevanten Argument-Wahrscheinlichkeiten an erster Stelle wieder. Demnach könnte man vermuten, daß bereits $s(m) = 1m^5$ EGPn genügen, um diese Eigneschaft zu bewerkstelligen.
In diesem Beispiel ist aber auch zu erkennen, daß die Grenze zwischen dem 6. und 7. Tabellenwert fließend ist. D.h. die zugehörigen Argument-Wahrscheinlichkeiten unterscheiden sich nur unwesentlich voneinander. Bei einem ähnlichen Test mit der gleichen Anzahl an EGPn kann nicht davon ausgegangen werden, daß alle $r = 6$ relevanten c-Werte wieder ganz oben stehen. Um sicherzustellen, daß zwischen relevanten und nicht-relevanten Elementen ein stabiler Abstand besteht, ist es ratsam, die Anzahl der EGPn weiter zu erhöhen.

Test 4: Die verringerte Anzahl an EGPn beträgt in diesem TSA $s(m) = 7m^5$. Tabelle 18 sortiert die reduzierten Argument-Wahrscheinlichkeiten $P_{red}[|c>]$ nach Größe und zeigt davon die erstbesten $3r = 18$.

	c	$\frac{P_{red}(\|c>)}{10^{-5}}$		c	$\frac{P_{red}(\|c>)}{10^{-5}}$		c	$\frac{P_{red}(\|c>)}{10^{-5}}$
1.	**8192**	3.51891	7.	13654	0.65536	13.	2892	0.26814
2.	**0**	3.01898	8.	5462	0.52051	14.	13708	0.25692
3.	**5461**	2.98731	9.	2730	0.51632	15.	10620	0.25307
4.	**10923**	2.78526	10.	10922	0.47739	16.	9517	0.24877
5.	**13653**	2.70292	11.	15427	0.28213	17.	5748	0.24837
6.	**2731**	1.77402	12.	2729	0.27368	18.	6212	0.24334

Tabelle 18: Reduzierter Shor-Algorithmus mit $n = 91$, $a = 4$ und $s(m) = 7m^5$

Auswertung:
In Tabelle 18 finden sich alle sechs relevanten Argument-Wahrscheinlichkeiten an erster Stelle wieder. Im Gegensatz zu Test 3 ist hier der Abstand zum 7. c-Wert deutlich größer. Die Argument-Wahrscheinlichkeit des letzten relevanten c-Werts ist mehr als doppelt so groß wie diejenige des ersten nicht-relevanten c-Werts.

6.3 Bewertungskriterien für eine Versuchsreihe von Shor-Algorithmen

Im folgenden wird das Shor-Verfahren als stochastischer Algorithmus am Beispiel für $n = 91$ und $a = 4$ näher untersucht. Hierbei wird eine fortlaufende Reihe an Tests mit unterschiedlicher Anzahl an EGPn durchgeführt.

- Um die Untersuchung einfach zu gestalten, werden für jeden Test als Anzahl $s(m)$ der EGPn einfache Polynome der Form km^p gewählt.
- Für jeden Test ergeben sich pro c-Wert reduzierte Argument-Wahrscheinlichkeiten. Diese werden der Größe nach sortiert. Nur die erstbesten $3r = 18$ Werte werden in einer Wahrscheinlichkeitstabelle abgelegt und weiter ausgewertet.

In Tabelle 19 sind für diese Testreihe die Ergebnisse abgelegt.

km^p	Rel c: Gesamt	Rel c: Erste r	Rel c: Zweite r	Rel c: Rest	Position
$1m^4$	0	0	0	0	(4)
$2m^4$	0	0	0	0	0
$3m^4$	0	0	0	0	(9)
$4m^4$	1	1	0	0	0
$5m^4$	1	1	0	0	0
$6m^4$	0	0	0	0	(6)
$7m^4$	4	3	1	0	2
$8m^4$	3	1	2	0	(12)
$9m^4$	3	2	1	0	3
$10m^4$	4	4	0	0	4
$11m^4$	4	4	0	0	1
$12m^4$	5	4	0	1	2
$13m^4$	5	4	1	0	5
$1m^5$	6	6	0	0	1
$2m^5$	6	6	0	0	2
$3m^5$	6	6	0	0	5
$4m^5$	6	6	0	0	4
$5m^5$	6	6	0	0	4
$6m^5$	6	6	0	0	3
$7m^5$	6	6	0	0	5
$8m^5$	6	6	0	0	4

Tabelle 19: Testreihe des Shor-Algorithmus mit $n = 91$ und $a = 4$

Zur Tabelle 19 einige Erläuterungen:

- Für diese Versuchsreihe wurden Tests von $s(m) = 1m^4$ bis $s(m) = 8m^5$ gewählt, weil sich die einzelnen Tests gegen Ende dieses Intervalls bzgl. der Anzahl relevanter c-Werte „stabilisieren“, wie in den folgenden Punkten näher erläutert wird.

- In der mit „Gesamt“ bezeichneten Spalte ist für jeden Versuch die Anzahl der relevanten c-Werte, die sich innerhalb der nach Größe sortierten Wahrscheinlichkeitstabelle mit den $3r = 18$ reduzierten besten Argument-Wahrscheinlichkeiten wiederfinden, eingetragen.

- Von der Anzahl dieser relevanten c-Werte wird in der nächsten Spalte („Erste r“) angezeigt, wieviele davon sich in den ersten $r = 6$ Positionen der Wahrscheinlichkeitstabelle $(1., 2., \cdots r.)$ wiederfinden.

- Eine Spalte („Zweite r“) weiter enthält die Anzahl der relevanten c-Werte in den nächsten r Positionen der Wahrscheinlichkeitstabelle $(r + 1, r + 2, \cdots 2r)$.

- Die Rest-Spalte enthält die Anzahl der gefundenen Werte, die im Rest der Wahrscheinlichkeitstabelle vorhanden sind.

- Die mit „Position“ bezeichnete Spalte stellt die Position des ersten c-Wertes innerhalb der Wahrscheinlichkeitstabelle der ersten $3r$ Werte dar, für den die Kettenbruch-Zerlegung die korrekte Ordnung r ermittelt. Befindet sich keine der in Frage kommenden Zahlen unter den erstbesten $3r$, dann wird untersucht, ob sich in der Wahrscheinlichkeitstabelle ein c-Wert befindet, der zu einem ungeradzahligen Vielfachen der Ordnung r führt. Dieses ungeradzahlige Vielfache der Form $(2k+1)r$ hat auch eine korrekte Zerlegung der Zahl n zur Folge, da

  $$a^{\frac{1}{2}(2k+1)r} = a^{kr+\frac{r}{2}} = a^{\frac{r}{2}} \pmod{n}$$

  ist. Für diesen c-Wert wird die Tabellen-Position in Klammern dargestellt.

Tabelle 19 kann folgendermaßen interpretiert werden.

- Für die Tests von $s(m) = 1m^4$ bis $s(m) = 6m^4$ ist das Vorhandensein eines relevanten c-Werts innerhalb der Wahrscheinlichkeitstabelle der erstbesten $3r = 18$ Argument-Wahrscheinlichkeiten zufällig.

- Ab der EGP-Anzahl $s(m) = 7m^4$ steigt die Anzahl relevanter c-Werte in der Wahrscheinlichkeitstabelle an. (Bei $s(m) = 7m^4$ wird sogar ein relevanter c-Wert gefunden, der über Kettenbruch-Zerlegung zur korrekten Ordnung r führt.)

- Ab der EGP-Anzahl $s(m) = 1m^5$ befinden sich alle relevanten c-Werte innerhalb der ersten $r = 6$ Positionen der Wahrscheinlichkeitstabelle.

- Untersucht man die einzelnen Tests näher, dann sind ab einer EGP-Anzahl $s(m) = 7m^5$ die Argument-Wahrscheinlichkeiten der oberen 6 relevanten c-Werte sogar mehr als doppelt so groß wie alle nachfolgenden.

Eine weitere Testreihe wird in Tabelle 20 dargestellt. Hier ist $n = 133$ und $a = 18$, auch hier gilt $r = 6$, nur die Dimension des Argument-Hilbertraums ist mit $m = 15$ größer. Die Gesamtzahl der EGPn für das komplette Shor-Verfahren liegt bei $q^2 = 2^{15}$ und wird von unten durch $s(m) = 6m^7$ angenähert.

km^p	Rel c: Gesamt	Rel c: Erste r	Rel c: Zweite r	Rel c: Rest	Position
$1m^4$	0	0	0	0	(3)
$2m^4$	0	0	0	0	(15)
$3m^4$	1	1	0	0	(4)
$4m^4$	1	0	0	1	15
$5m^4$	1	1	0	0	0
$6m^4$	1	1	0	0	0
$7m^4$	1	0	0	1	15
$8m^4$	1	1	0	0	(1)
$9m^4$	1	1	0	0	1
$10m^4$	1	1	0	0	(2)
$11m^4$	1	1	0	0	0
$12m^4$	4	4	0	0	1
$13m^4$	5	4	1	0	2
$14m^4$	2	2	0	1	4
$1m^5$	2	1	1	0	7
$2m^5$	6	5	1	0	2
$3m^5$	6	6	0	0	4
$4m^5$	6	6	0	0	4
$5m^5$	6	6	0	0	5
$6m^5$	6	6	0	0	4
$7m^5$	6	6	0	0	3
$8m^5$	6	6	0	0	5
$9m^5$	6	6	0	0	4

Tabelle 20: Versuchsreihe des Shor-Algorithmus mit $n = 133$ und $a = 18$

Überprüft man Tabelle 20 nach den gleichen Kriterien wie Tabelle 19, dann kommt man zu folgendem Schluß:

- Für die Tests von $s(m) = 1m^4$ bis $s(m) = 11m^4$ ist ist das Vorhandensein eines relevanten c-Werts innerhalb der Wahrscheinlichkeitstabelle der erstbesten $3r = 18$ Argument-Wahrscheinlichkeiten zufällig.
- Ab der EGP-Anzahl $s(m) = 3m^5$ befinden sich alle $r = 6$ relevanten c-Werte an der ersten Stelle der Wahrscheinlichkeitstabelle.
- Eine deutliche Trennung von relevanten und nicht-relevanten c-Werten ergibt sich ab der EGP-Anzahl $s(m) = 4m^5$. Von da an sind die Argument-Wahrscheinlichkeiten der relevanten c-Werte mindestens doppelt so groß wie bei allen anderen c-Werten.

6.4 Bewertung von Versuchsreihen mit konstanter Ordnung r

6.4.1 Bewertung mehrerer Versuchsreihen für r=6

In diesem Kapitel werden für verschiedene n-Werte Versuchsreihen von Shor-Algorithmen, die alle die Ordnung $r = 6$ zum Ziel haben, in Tabelle 21 zusammengefaßt.

n	a	m	r	Test Name	Test Reihe	(I) km^p	(II) km^p	(III) km^p	q^2 $k_s m^{p_s}$
$*13$	4	8	6	Shor Short	$1m^3$ $1m^5$	$2m^3$ $2m^3$	$2m^4$ $2m^4$	$4m^4$ $4m^4$	$2m^5$
$*13$	4	8	6	Shor Short	$1m^3$ $1m^5$	$4m^3$ $4m^3$	$3m^4$ $3m^4$	$5m^4$ $5m^4$	$2m^5$
$*13$	4	8	6	Shor Short	$1m^3$ $1m^5$	$2m^3$ $4m^3$	$2m^4$ $2m^4$	$6m^4$ $6m^4$	$2m^5$
$*13$	4	8	6	Shor Short	$1m^3$ $1m^5$	$4m^3$ $4m^3$	$3m^4$ $3m^4$	$3m^4$ $5m^4$	$2m^5$
$*13$	10	8	6	Shor Short	$1m^3$ $1m^5$	$3m^3$ $6m^3$	$2m^4$ $2m^4$	$5m^4$ $6m^4$	$2m^5$
$*13$	10	8	6	Shor Short	$1m^3$ $1m^5$	$3m^3$ $5m^3$	$3m^4$ $3m^4$	$4m^4$ $6m^4$	$2m^5$
$*13$	10	8	6	Shor Short	$1m^3$ $1m^5$	$2m^3$ $5m^3$	$2m^4$ $2m^4$	$3m^4$ $5m^4$	$2m^5$
$*13$	10	8	6	Shor Short	$1m^3$ $1m^5$	$5m^3$ $5m^3$	$2m^4$ $2m^4$	$4m^4$ $4m^4$	$2m^5$

Tabelle 21: Stabilität von Versuchsreihen des Shor-Algorithmus mit $r = 6$

n	a	m	r	Test Name	Test Reihe	(I) km^p	(II) km^p	(III) km^p	q^2 $k_s m^{p_s}$
21	2	9	6	Shor Short	$1m^3$ $3m^5$	$5m^3$ $1m^4$	$3m^4$ $3m^4$	$5m^4$ $1m^5$	$4m^5$
21	2	9	6	Shor Short	$1m^3$ $3m^5$	$7m^3$ $1m^4$	$3m^4$ $3m^4$	$7m^4$ $8m^4$	$4m^5$
21	10	9	6	Shor Short	$1m^3$ $3m^5$	$5m^3$ $5m^3$	$3m^4$ $3m^4$	$6m^4$ $1m^5$	$4m^5$
21	10	9	6	Shor Short	$1m^3$ $3m^5$	$6m^3$ $6m^3$	$2m^4$ $2m^4$	$5m^4$ $6m^4$	$4m^5$
21	11	9	6	Shor Short	$1m^3$ $3m^5$	$5m^3$ $7m^3$	$3m^4$ $3m^4$	$3m^4$ $8m^4$	$4m^5$
21	11	9	6	Shor Short	$1m^3$ $3m^5$	$6m^3$ $8m^3$	$2m^4$ $2m^4$	$5m^4$ $6m^4$	$4m^5$
21	19	9	6	Shor Short	$1m^3$ $3m^5$	$1m^4$ $1m^4$	$3m^4$ $3m^4$	$6m^4$ $6m^4$	$4m^5$
21	19	9	6	Shor Short	$1m^3$ $3m^5$	$8m^3$ $8m^3$	$2m^4$ $2m^4$	$5m^4$ $7m^4$	$4m^5$
$*31$	6	10	6	Shor Short	$1m^3$ $5m^5$	$9m^3$ $2m^4$	$4m^4$ $4m^4$	$8m^4$ $9m^4$	$1m^6$
$*31$	6	10	6	Shor Short	$1m^3$ $5m^5$	$1m^4$ $1m^4$	$4m^4$ $4m^4$	$9m^4$ $2m^5$	$1m^6$
$*31$	6	10	6	Shor Short	$1m^3$ $5m^5$	$2m^4$ $2m^4$	$3m^4$ $3m^4$	$2m^5$ $2m^5$	$1m^6$
$*31$	6	10	6	Shor Short	$1m^3$ $5m^5$	$9m^3$ $2m^4$	$4m^4$ $4m^4$	$7m^4$ $9m^4$	$1m^6$
$*31$	26	10	6	Shor Short	$1m^3$ $5m^5$	$8m^3$ $2m^4$	$4m^4$ $4m^4$	$9m^4$ $9m^4$	$1m^6$
$*31$	26	10	6	Shor Short	$1m^3$ $5m^5$	$1m^4$ $3m^4$	$4m^4$ $6m^4$	$6m^4$ $2m^5$	$1m^6$
$*31$	26	10	6	Shor Short	$1m^3$ $5m^5$	$1m^4$ $1m^4$	$5m^4$ $5m^4$	$9m^4$ $1m^5$	$1m^6$
$*31$	26	10	6	Shor Short	$1m^3$ $5m^5$	$2m^4$ $2m^4$	$4m^4$ $4m^4$	$8m^4$ $1m^5$	$1m^6$
35	4	11	6	Shor Short	$1m^3$ $5m^5$	$2m^4$ $2m^4$	$4m^4$ $4m^4$	$9m^4$ $1m^5$	$2m^6$

Tabelle 21: Stabilität von Versuchsreihen des Shor-Algorithmus mit $r = 6$

n	a	m	r	Test Name	Test Reihe	(I) km^p	(II) km^p	(III) km^p	q^2 $k_s m^{p_s}$
35	9	11	6	Shor	$1m^3$	$3m^4$	$3m^4$	$1m^5$	$2m^6$
				Short	$5m^5$	$3m^4$	$7m^4$	$4m^5$	
35	26	11	6	Shor	$1m^3$	$3m^4$	$6m^4$	$1m^5$	$2m^6$
				Short	$5m^5$	$3m^4$	$6m^4$	$2m^5$	
35	31	11	6	Shor	$1m^3$	$2m^4$	$3m^4$	$1m^5$	$2m^6$
				Short	$5m^5$	$2m^4$	$5m^4$	$2m^5$	
45	4	11	6	Shor	$1m^3$	$2m^4$	$5m^4$	$2m^5$	$2m^6$
				Short	$5m^5$	$2m^4$	$1m^5$	$2m^5$	
45	11	11	6	Shor	$1m^3$	$3m^4$	$5m^4$	$2m^5$	$2m^6$
				Short	$5m^5$	$3m^4$	$5m^4$	$2m^5$	
45	34	11	6	Shor	$1m^3$	$9m^3$	$8m^4$	$10m^4$	$2m^6$
				Short	$5m^5$	$2m^4$	$8m^4$	$2m^5$	
45	41	11	6	Shor	$1m^3$	$2m^4$	$5m^4$	$1m^5$	$2m^6$
				Short	$5m^5$	$2m^4$	$7m^4$	$3m^5$	
57	11	12	6	Shor	$1m^4$	$4m^4$	$8m^4$	$11m^4$	$5m^6$
				Short	$5m^5$	$4m^4$	$8m^4$	$2m^5$	
57	26	12	6	Shor	$1m^4$	$3m^4$	$6m^4$	$2m^5$	$5m^6$
				Short	$5m^5$	$5m^4$	$2m^5$	$3m^5$	
57	31	12	6	Shor	$1m^4$	$3m^4$	$7m^4$	$2m^5$	$5m^6$
				Short	$5m^5$	$3m^4$	$7m^4$	$2m^5$	
57	46	12	6	Shor	$1m^4$	$4m^4$	$8m^4$	$2m^5$	$5m^6$
				Short	$5m^5$	$4m^4$	$2m^5$	$2m^5$	
63	2	12	6	Shor	$1m^3$	$2m^4$	$8m^4$	$2m^5$	$5m^6$
				Short	$5m^5$	$2m^4$	$8m^4$	$2m^5$	
63	10	12	6	Shor	$1m^3$	$4m^4$	$8m^4$	$11m^4$	$5m^6$
				Short	$5m^5$	$4m^4$	$11m^4$	$2m^5$	
63	19	12	6	Shor	$1m^3$	$3m^4$	$8m^4$	$2m^5$	$5m^6$
				Short	$5m^5$	$3m^4$	$10m^4$	$4m^5$	
63	29	12	6	Shor	$1m^3$	$4m^4$	$7m^4$	$11m^4$	$5m^6$
				Short	$5m^5$	$4m^4$	$7m^4$	$2m^5$	
65	9	13	6	Shor	$1m^4$	$4m^4$	$11m^4$	$2m^5$	$1m^7$
				Short	$7m^5$	$7m^4$	$2m^5$	$5m^5$	
65	29	13	6	Shor	$1m^4$	$3m^4$	$9m^4$	$2m^5$	$1m^7$
				Short	$7m^5$	$5m^4$	$2m^5$	$3m^5$	

Tabelle 21: Stabilität von Versuchsreihen des Shor-Algorithmus mit $r = 6$

n	a	m	r	Test Name	Test Reihe	(I) km^p	(II) km^p	(III) km^p	q^2 $k_s m^{p_s}$
65	36	13	6	Shor	$1m^4$	$6m^4$	$12m^4$	$2m^5$	$1m^7$
				Short	$7m^5$	$6m^4$	$12m^4$	$4m^5$	
65	56	13	6	Shor	$1m^4$	$5m^4$	$12m^4$	$2m^5$	$1m^7$
				Short	$7m^5$	$7m^4$	$12m^4$	$2m^5$	
77	12	13	6	Shor	$1m^4$	$4m^4$	$8m^4$	$2m^5$	$1m^7$
				Short	$7m^5$	$6m^4$	$12m^4$	$4m^5$	
77	32	13	6	Shor	$1m^4$	$5m^4$	$10m^4$	$2m^5$	$1m^7$
				Short	$7m^5$	$5m^4$	$2m^5$	$2m^5$	
77	45	13	6	Shor	$1m^4$	$6m^4$	$10m^4$	$2m^5$	$1m^7$
				Short	$7m^5$	$6m^4$	$12m^4$	$5m^5$	
77	65	13	6	Shor	$1m^4$	$7m^4$	$9m^4$	$2m^5$	$1m^7$
				Short	$7m^5$	$7m^4$	$2m^5$	$2m^5$	
91	3	14	6	Shor	$1m^4$	$7m^4$	$2m^5$	$3m^5$	$2m^7$
				Short	$8m^5$	$10m^4$	$2m^5$	$5m^5$	
91	4	14	6	Shor	$1m^4$	$7m^4$	$1m^5$	$2m^5$	$2m^7$
				Short	$8m^5$	$7m^4$	$1m^5$	$7m^5$	
91	23	14	6	Shor	$1m^4$	$8m^4$	$13m^4$	$3m^5$	$2m^7$
				Short	$8m^5$	$8m^4$	$2m^5$	$3m^5$	
91	25	14	6	Shor	$1m^4$	$9m^4$	$2m^5$	$2m^5$	$2m^7$
				Short	$8m^5$	$9m^4$	$2m^5$	$2m^5$	
105	4	14	6	Shor	$1m^4$	$5m^4$	$2m^5$	$2m^5$	$2m^7$
				Short	$8m^5$	$8m^4$	$2m^5$	$7m^5$	
105	11	14	6	Shor	$1m^4$	$6m^4$	$2m^5$	$2m^5$	$2m^7$
				Short	$8m^5$	$10m^4$	$2m^5$	$3m^5$	
105	26	14	6	Shor	$1m^4$	$9m^4$	$12m^4$	$3m^5$	$2m^7$
				Short	$8m^5$	$9m^4$	$2m^5$	$3m^5$	
105	31	14	6	Shor	$1m^4$	$6m^4$	$2m^5$	$3m^5$	$2m^7$
				Short	$8m^5$	$9m^4$	$2m^5$	$5m^5$	
133	8	15	6	Shor	$1m^4$	$12m^4$	$2m^5$	$6m^5$	$6m^7$
				Short	$9m^5$	$1m^5$	$2m^5$	$7m^5$	
133	18	15	6	Shor	$1m^4$	$12m^4$	$3m^5$	$4m^5$	$6m^7$
				Short	$9m^5$	$2m^5$	$3m^5$	$4m^5$	
133	26	15	6	Shor	$1m^4$	$10m^4$	$3m^5$	$3m^5$	$6m^7$
				Short	$9m^5$	$1m^5$	$3m^5$	Offen	

Tabelle 21: Stabilität von Versuchsreihen des Shor-Algorithmus mit $r = 6$

n	a	m	r	Test Name	Test Reihe	(I) km^p	(II) km^p	(III) km^p	q^2 $k_s m^{p_s}$
133	37	15	6	Shor Short	$1m^4$ $9m^5$	$11m^4$ $1m^5$	$2m^5$ $2m^5$	$3m^5$ $9m^5$	$6m^7$
155	6	15	6	Shor Short	$1m^4$ $9m^5$	$7m^4$ $14m^4$	$2m^5$ $2m^5$	$3m^5$ $6m^5$	$6m^7$
155	26	15	6	Shor Short	$1m^4$ $9m^5$	$8m^4$ $14m^4$	$2m^5$ $2m^5$	$3m^5$ $5m^5$	$6m^7$
175	26	15	6	Shor Short	$1m^4$ $9m^5$	$10m^4$ $14m^4$	$2m^5$ $2m^5$	$4m^5$ $8m^5$	$6m^7$
175	74	15	6	Shor Short	$1m^4$ $9m^5$	$9m^4$ $1m^5$	$2m^5$ $2m^5$	$4m^5$ $6m^5$	$6m^7$

Tabelle 21: Stabilität von Versuchsreihen des Shor-Algorithmus mit $r = 6$

In Tabelle 21 werden auch Testreihen für n-Werte durchgeführt, die Primzahlen sind, da für einige m-Werte keine Nicht-Primzahlen zur Verfügung stehen. (Für die Bit-Dimension $m = 10$ liegt die zu zerlegende Zahl n im Intervall $[23, \ldots 31]$. Die Zahlen 25 und 27 fallen als reine Potenzen von Primzahlen aus.)

Alle n-Werte, die Primzahlen sind, sind in der Tabelle durch einen Stern $(*)$ gekennzeichnet. Da das Shor-Verfahren in primärer Hinsicht die Ermittlung relevanter c-Werte und die Bestimmung der Ordnung r und nicht die Zerlegung von n in Faktoren zum Ziel hat, können Testreihen über Primzahlen in die Gesamtanalyse einbezogen werden.

Die mit „TestName“ benannte Spalte gibt an, ob die jeweilige Testreihe mit dem Programm „ShorPair“ oder mit „ShorShort“ durchgeführt worden ist.

In der mit „TestReihe“ bezeichneten Spalte sind Anfang und Ende der jeweiligen Versuchsreihe dargestellt. Z.B. wurden für $n = 13$ die Tests beginnend mit der Anzahl der EGPn $s(m) = 1m^3$, $2m^3$, $3m^3$, usw. über $s(m) = 1m^4$, $2m^4$, $3m^4$, usw. bis hin zu $s(m) = 1m^5$ durchgeführt. Bei jedem Test wurden die reduzierten Argument-Wahrscheinlichkeiten der Größe nach sortiert. Nur die $3r = 18$ größten Argument-Wahrscheinlichkeiten wurden in einer Wahrscheinlichkeitstabelle abgelegt und nach relevanten c-Werten untersucht.

Die mit (I) bezeichnete Spalte macht eine Aussage über diejenigen Versuche innerhalb der Testreihen, bei denen mindestens die Hälfte aller relevanten c-Werte innerhalb der Wahrscheinlichkeitstabelle der $3r = 18$ größten reduzierten Argument-Wahrscheinlichkeiten enthalten sind. Da $r = 6$ ist, sind das mindestens 3 relevante c-Werte. Es werden zwei Tests angezeigt.

- Der obere Wert zeigt den Testfall an, bei dem diese Eigenschaft zum ersten Mal festgestellt wurde.

- Beim unteren Wert gilt, daß alle weiteren Tests der Versuchsreihe mit größerem $s(m)$ mindestens 3 relevante Werte enthalten. Ab hier fängt die Versuchsreihe an, sich zu „stabilisieren".

Unter (II) sind diejenigen Testfälle eingetragen, bei denen in der Wahrscheinlichkeitstabelle mit den $3r = 18$ größten reduzierten Argument-Wahrscheinlichkeiten ALLE relevanten c-Werte ganz oben stehen. Da $r = 6$ ist, sind dies insgesamt 6 relevanten c-Werte. Es werden zwei Tests angezeigt.

- Der obere Wert zeigt den Testfall an, bei dem diese Eigenschaft zum ersten Mal festgestellt wurde.
- Beim unteren Wert gilt, daß alle weiteren Tests der Versuchsreihe mit größerem $s(m)$ die 6 relevanten Werte an erster Stelle haben.

Unter (III) sind diejenigen Testfälle eingetragen, bei denen sich diejenigen $r = 6$ relevanten $c - Werte$, die sich in der Wahrscheinlichkeitstabelle ganz oben befinden, in Bezug auf die reduzierten Argument-Wahrscheinlichkeit deutlich von den nicht-relevanten Elementen der Tabelle absetzen. Von Bedeutung ist hier der Vergleich zwischen dem 6. c-Wert, der zu den relevanten gehört, und dem 7. Wert, der nicht mehr dazuzählt.

- Beim oberen Wert ist der 6. c-Wert mindestens 1.5 mal so groß wie der 7. Wert.
- Beim unteren ist er mehr als doppelt so groß.
- Bei beiden Spalteneinträgen handelt es sich um die untere Grenze, für die diese Aussage Gültigkeit besitzt. Alle weiteren Tests der Versuchsreihe haben die gleiche Eigenschaft.

Die Kriterien für Spalte (III) machen für die Versuchsreihen mit $r = 6$ Sinn, da im KMA, wie bereits weiter oben geschildert, die Argument-Wahrscheinlichkeit des 6. c-Werts mit $P(|c>) = 0.114$ ungefähr viermal so groß ist wie beim 7. c-Wert mit $P(|c>) = 0.028$.
Tabelle 21 enthält in der letzten Spalte den Ausdruck, der im Programm „RecoExpo" berechnet wird, und der die Dimension des KMA (2^{2m}) durch ein Polynom der Form $k_s m^{p_s}$ von unten annähert. Mit diesem Ausdruck können die Ergebnisse in den Spalten (I), (II) und (III) verglichen werden.

Auswertung der Tabelle 21:
Die Spalten (I), (II) und (III) geben Auskunft über die Rolle der relevanten c-Werte im TSA des Shor-Verfahrens für Testreihen mit steigendem n und m. Alle Versuchsreihen mit gleicher Bit-Dimension m werden nun zusammengefaßt und in eine Gesamtwertung einbezogen. Hierbei wird für die Werte in den jeweiligen Spalten der logarithmische Mittelwert

$$\overline{s(m)} = \sqrt[K]{\prod_{k=1}^{K} s_k(m)} \tag{105}$$

ermittelt und anschließend auf- oder abgerundet. Im Zweifelsfall wird aufgerundet. Das Ergebnis ist in Tabelle 22 dargestellt.

m	(I)	(II)	(III)
8	$2.95 \cdot m^3$ $4.20 \cdot m^3$	$2.32 \cdot m^4$ $2.32 \cdot m^4$	$4.14 \cdot m^4$ $5.06 \cdot m^4$
9	$6.23 \cdot m^3$ $7.48 \cdot m^3$	$2.58 \cdot m^4$ $2.58 \cdot m^4$	$5.12 \cdot m^4$ $7.27 \cdot m^4$
10	$1.13 \cdot m^4$ $1.77 \cdot m^4$	$3.97 \cdot m^4$ $4.17 \cdot m^4$	$8.90 \cdot m^4$ $1.25 \cdot m^5$
11	$2.11 \cdot m^4$ $2.33 \cdot m^4$	$4.64 \cdot m^4$ $6.33 \cdot m^4$	$1.15 \cdot m^5$ $2.10 \cdot m^5$
12	$3.29 \cdot m^4$ $3.51 \cdot m^4$	$7.46 \cdot m^4$ $10.90 \cdot m^4$	$1.49 \cdot m^5$ $2.29 \cdot m^5$
13	$4.84 \cdot m^4$ $6.07 \cdot m^4$	$10.03 \cdot m^4$ $1.36 \cdot m^5$	$2.00 \cdot m^5$ $3.15 \cdot m^5$
14	$7.00 \cdot m^4$ $8.69 \cdot m^4$	$1.50 \cdot m^5$ $1.83 \cdot m^5$	$2.45 \cdot m^5$ $4.00 \cdot m^5$
15	$9.72 \cdot m^4$ $1.06 \cdot m^5$	$2.21 \cdot m^5$ $2.21 \cdot m^5$	$3.64 \cdot m^5$ $6.61 \cdot m^5$

Tabelle 22: Stabilitätstabelle für $r = 6$

Die folgenden Abbildungen 30, 31 und 32 geben die Tabelle 22 graphisch und in logarithmischer Darstellung wieder. In allen drei Darstellungen ist anhand der schwarzen Punkte zu erkennen, daß mit steigendem m auch $\overline{s(m)}$, wenn auch unterschiedlich stark, anwächst. Der nahezu lineare Graph mit den Kreisen (**Kreis-Graph**) gibt als Vergleich den logarithmischen Gesamtaufwand

$$\log_m(s_{Ges}(m)) \;=\; \log_m(q^2) \;=\; \log_m(2^{2m}) \;=\; 2m \cdot \log_m(2)$$

des jeweiligen KMA in Abhängigkeit von der Bit-Dimension m wieder.

Fazit:

Auch wenn die „Stabilitäts“-Kriterien in den Fällen (I), (II) und (III) unterschiedlich zu bewerten sind, läßt sich dennoch eine gemeinsame Aussage machen: Die logarithmische Steigung der durchschnittlichen TSA-Kurven ist konstant positiv, aber flacher als beim KMA.

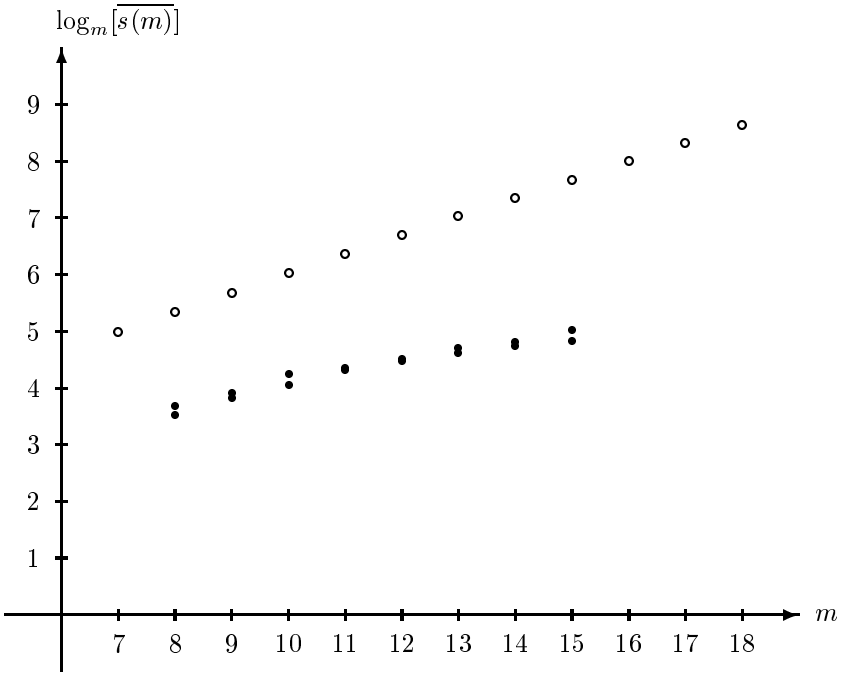

Abbildung 30: Stabilitätskurve für $r = 6$ für den Bereich (I)

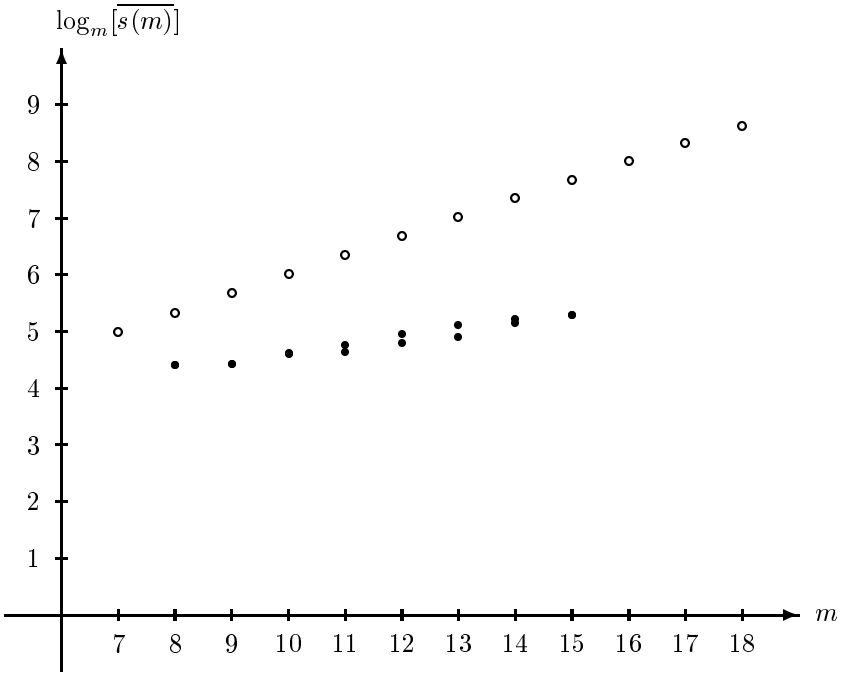

Abbildung 31: Stabilitätskurve für $r = 6$ für den Bereich (II)

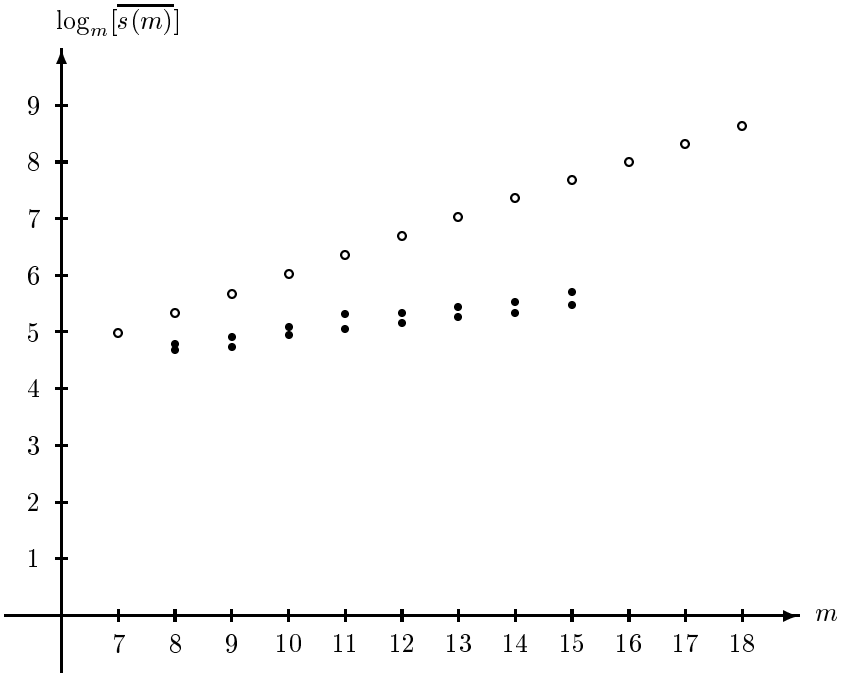

Abbildung 32: Stabilitätskurve für $r = 6$ für den Bereich (III)

6.4.2 Bewertung mehrerer Versuchsreihen für r=10

In diesem Kapitel werden mehrere Versuchsreihen mit unterschiedlichen n-Werten bei gleichbleibender Ordnung $r = 10$ untersucht. Anhand des Beispiels für $n = 33$ und $n = 2$ wird zunächst gezeigt, wie beim KMA die relevanten Argument-Wahrscheinlichkeiten im Fall $r = 10$ i.A. verteilt sind. Auch hier sind diejenigen c-Werte, die über Kettenbruch-Zerlegung zu einer korrekten Ordnung $r = 10$ führen, fett unterlegt.

	c	$P(\|c>)$	d	r
0.	0	0.100000	0	0
1.	**205**	0.087514	1	**10**
2.	410	0.057279	1	5
3.	**614**	0.057279	3	**10**
4.	819	0.087514	2	5
5.	1000	0.100000	1	2
6.	1229	0.087514	3	5
7.	**1434**	0.057279	7	**10**
8.	1638	0.057279	4	5
9.	**1843**	0.087514	9	**10**
	409	0.025458		
	615	0.025458		
	1433	0.025458		
	1639	0.025458		

Tabelle 23: Relevante Argument-Wahrscheinlichkeiten des Shor-Verfahrens für $n = 33$, $a = 2$ und $r = 10$

Unterhalb der $r = 10$ relevanten c-Werte werden in Tabelle 23 die nicht-relevanten c-Werte mit den nächstgroßen Argument-Wahrscheinlichkeiten dargestellt. Folgende Dinge fallen auf:

- Alle relevanten c-Werten unterscheiden sich in ihrem Wahrscheinlichkeitsverhalten in einer Bandbreite zwischen $P = 0.057$ und $P = 0.1$. Dieser Größenunterschied ist bei der Bewertung von reduzierten Argument-Wahrscheinlichkeiten zu berücksichtigen.
- Die niedrigsten relevanten Argument-Wahrscheinlichkeiten sind mehr als doppelt so groß wie die größten nicht-relevanten. Dieser Unterschied ist nicht so deutlich wie bei den KMAs mit $r = 6$ (siehe Tabelle 14). Aus diesem Grund müssen die Bewertungskriterien für die TSA-Testreihen mit $r = 10$, speziell in Spalte (III), anders definiert werden als bei $r = 6$.

Für die Bit-Dimensionen $m = 8$ und $m = 9$ liegen die n-Werte im Intervall $[13, \ldots 21]$. Da sich hierfür keine Beispiele finden, in denen die Ordnung $r = 10$ berechnet wird, muß auf die entsprechenden Testreihen verzichtet werden.

Innerhalb der einzelnen durchgeführten Tests werden die reduzierten Argument-Wahrscheinlichkeiten der Größe nach sortiert. Die erstbesten $3r = 30$ Werte werden in einer Wahrscheinlichkeitstabelle abgelegt und bzgl. relevanter c-Werte überprüft. In der folgenden Tabelle 24 werden die Versuchsreihen dargestellt. Die Bewertungskriterien der Spalten (I), (II) und (III) werden weiter unten erläutert.

n	a	m	r	Test Name	Test Reihe	(I) km^p	(II) km^p	(III) km^p	q^2 $k_s m^{p_s}$
$*11$	2	7	10	Shor Short	$1m^2$ $6m^4$	$2m^3$ $2m^3$	$2m^4$ $4m^4$	$6m^4$ $6m^4$	$6m^4$
$*11$	2	7	10	Shor Short	$1m^2$ $6m^4$	$3m^3$ $3m^3$	$3m^4$ $3m^4$	$3m^4$ $3m^4$	$6m^4$
$*11$	6	7	10	Shor Short	$1m^2$ $6m^4$	$1m^3$ $4m^3$	$3m^4$ $3m^4$	$4m^4$ $6m^4$	$6m^4$
$*11$	6	7	10	Shor Short	$1m^2$ $6m^4$	$1m^3$ $4m^3$	$3m^4$ $3m^4$	$3m^4$ $5m^4$	$6m^4$
$*11$	7	7	10	Shor Short	$1m^2$ $6m^4$	$3m^3$ $3m^3$	$3m^4$ $3m^4$	$5m^4$ $5m^4$	$6m^4$
$*11$	7	7	10	Shor Short	$1m^2$ $6m^4$	$2m^3$ $2m^3$	$3m^4$ $3m^4$	$3m^4$ $5m^4$	$6m^4$
$*11$	8	7	10	Shor Short	$1m^2$ $6m^4$	$3m^3$ $3m^3$	$2m^4$ $4m^4$	$4m^4$ $4m^4$	$6m^4$
$*11$	8	7	10	Shor Short	$1m^2$ $6m^4$	$2m^2$ $2m^3$	$4m^4$ $4m^4$	$4m^4$ $6m^4$	$6m^4$
$*31$	15	10	10	Shor Short	$1m^2$ $5m^5$	$7m^3$ $2m^4$	$8m^4$ $8m^4$	$1m^5$ $3m^5$	$1m^6$
$*31$	15	10	10	Shor Short	$1m^2$ $5m^5$	$2m^4$ $2m^4$	$6m^4$ $1m^5$	$1m^5$ $1m^5$	$1m^6$
$*31$	23	10	10	Shor Short	$1m^2$ $5m^5$	$2m^4$ $2m^4$	$7m^4$ $9m^4$	$2m^5$ $2m^5$	$1m^6$
$*31$	23	10	10	Shor Short	$1m^2$ $5m^5$	$1m^4$ $1m^4$	$5m^4$ $2m^5$	$2m^5$ $2m^5$	$1m^6$
$*31$	27	10	10	Shor Short	$1m^2$ $5m^5$	$9m^3$ $2m^4$	$4m^4$ $8m^4$	$3m^5$ $3m^5$	$1m^6$
$*31$	27	10	10	Shor Short	$1m^2$ $5m^5$	$4m^3$ $2m^4$	$4m^4$ $8m^4$	$2m^5$ $2m^5$	$1m^6$
$*31$	29	10	10	Shor Short	$1m^2$ $5m^5$	$8m^3$ $2m^4$	$4m^4$ $1m^5$	$3m^5$ $3m^5$	$1m^6$

Tabelle 24: Stabilität von Versuchsreihen des Shor-Algorithmus mit $r = 10$

n	a	m	r	Test Name	Test Reihe	(I) km^p	(II) km^p	(III) km^p	q^2 $k_s m^{p_s}$
$*31$	29	10	10	Shor Short	$1m^2$ $5m^5$	$2m^4$ $2m^4$	$5m^4$ $1m^5$	$3m^5$ $3m^5$	$1m^6$
33	5	11	10	Shor Short	$1m^2$ $5m^5$	$3m^4$ $3m^4$	$8m^4$ $1m^5$	$2m^5$ $4m^5$	$2m^6$
33	7	11	10	Shor Short	$1m^2$ $5m^5$	$2m^4$ $2m^4$	$5m^4$ $1m^5$	$4m^5$ $4m^5$	$2m^6$
33	13	11	10	Shor Short	$1m^2$ $5m^5$	$3m^4$ $3m^4$	$9m^4$ $9m^4$	$10m^4$ $2m^5$	$2m^6$
33	14	11	10	Shor Short	$1m^2$ $5m^5$	$3m^4$ $3m^4$	$6m^4$ $2m^5$	$3m^5$ $3m^5$	$2m^6$
33	19	11	10	Shor Short	$1m^2$ $5m^5$	$2m^4$ $2m^4$	$7m^4$ $10m^4$	$3m^5$ $3m^5$	$2m^6$
33	20	11	10	Shor Short	$1m^2$ $5m^5$	$2m^4$ $4m^4$	$10m^4$ $10m^4$	$2m^5$ $2m^5$	$2m^6$
33	26	11	10	Shor Short	$1m^2$ $5m^5$	$3m^4$ $3m^4$	$10m^4$ $10m^4$	$3m^5$ $3m^5$	$2m^6$
33	28	11	10	Shor Short	$1m^2$ $5m^5$	$2m^4$ $2m^4$	$2m^5$ $2m^5$	$2m^5$ $2m^5$	$2m^6$
55	4	12	10	Shor Short	$1m^4$ $7m^5$	$3m^4$ $3m^4$	$11m^4$ $2m^5$	$2m^5$ $5m^5$	$5m^6$
55	6	12	10	Shor Short	$1m^4$ $7m^5$	$3m^4$ $5m^4$	$11m^4$ $2m^5$	$3m^5$ $3m^5$	$5m^6$
55	9	12	10	Shor Short	$1m^4$ $7m^5$	$4m^4$ $4m^4$	$11m^4$ $11m^4$	$2m^5$ $4m^5$	$5m^6$
55	14	12	10	Shor Short	$1m^4$ $9m^5$	$3m^4$ $3m^4$	$9m^4$ $11m^4$	$4m^5$ $7m^5$	$5m^6$
55	41	12	10	Shor Short	$1m^4$ $7m^5$	$4m^4$ $4m^4$	$1m^5$ $1m^5$	$5m^5$ $5m^5$	$5m^6$
55	46	12	10	Shor Short	$1m^4$ $7m^5$	$5m^4$ $5m^4$	$10m^4$ $3m^5$	$4m^5$ $4m^5$	$5m^6$
55	49	12	10	Shor Short	$1m^4$ $7m^5$	$5m^4$ $5m^4$	$10m^4$ $10m^4$	$2m^5$ $4m^5$	$5m^6$
55	51	12	10	Shor Short	$1m^4$ $9m^5$	$3m^4$ $3m^4$	$11m^4$ $3m^5$	$5m^5$ $8m^5$	$5m^6$

Tabelle 24: Stabilität von Versuchsreihen des Shor-Algorithmus mit $r = 10$

n	a	m	r	Test Name	Test Reihe	(I) km^p	(II) km^p	(III) km^p	q^2 $k_s m^{p_s}$
75	4	13	10	Shor	$1m^4$	$3m^4$	$2m^5$	$7m^5$	$1m^7$
				Short	$9m^5$	$7m^4$	$2m^5$	$7m^5$	
75	11	13	10	Shor	$1m^4$	$6m^4$	$12m^4$	$3m^5$	$1m^7$
				Short	$9m^5$	$6m^4$	$2m^5$	$3m^5$	
75	19	13	10	Shor	$1m^4$	$6m^4$	$2m^5$	$4m^5$	$1m^7$
				Short	$9m^5$	$8m^4$	$2m^5$	$8m^5$	
75	34	13	10	Shor	$1m^4$	$5m^4$	$2m^5$	$2m^5$	$1m^7$
				Short	$9m^5$	$7m^4$	$2m^5$	$4m^5$	
77	8	13	10	Shor	$1m^4$	$7m^4$	$12m^4$	$3m^5$	$1m^7$
				Short	$9m^5$	$7m^4$	$2m^5$	$6m^5$	
77	20	13	10	Shor	$1m^4$	$5m^4$	$2m^5$	$2m^5$	$1m^7$
				Short	$9m^5$	$5m^4$	$2m^5$	$4m^5$	
77	27	13	10	Shor	$1m^4$	$6m^4$	$2m^5$	$4m^5$	$1m^7$
				Short	$9m^5$	$6m^4$	$2m^5$	$8m^5$	
77	29	13	10	Shor	$1m^4$	$6m^4$	$3m^5$	$5m^5$	$1m^7$
				Short	$9m^5$	$6m^4$	$3m^5$	$8m^5$	
93	2	14	10	Shor	$1m^4$	$8m^4$	$3m^5$	$10m^5$	$2m^7$
				Short	$11m^5$	$11m^4$	$3m^5$	$10m^5$	
93	8	14	10	Shor	$1m^4$	$11m^4$	$2m^5$	$6m^5$	$2m^7$
				Short	$11m^5$	$11m^4$	$2m^5$	Offen	
93	35	14	10	Shor	$1m^4$	$9m^4$	$2m^5$	$3m^5$	$2m^7$
				Short	$11m^5$	$9m^4$	$2m^5$	$10m^5$	
93	46	14	10	Shor	$1m^4$	$9m^4$	$2m^5$	$4m^5$	$2m^7$
				Short	$11m^5$	$9m^4$	$2m^5$	Offen	

Tabelle 24: Stabilität von Versuchsreihen des Shor-Algorithmus mit $r = 10$

Die Auswertung von Tabelle 24 erfolgt in ähnlicher Weise wie in Kapitel 6.4.1.

- In Spalte (I) sind die Testfälle eingetragen, bei denen mindestens $\frac{r}{2} = 5$ relevante c-Werte in der Wahrscheinlichkeitstabelle der besten $3r = 10$ Argument-Wahrscheinlichkeiten vorhanden sind.
- In Spalte (II) werden die Testfälle aufgeführt, bei denen die $r = 10$ relevanten c-Werte ganz oben in der Wahrscheinlichkeitstabelle stehen.
- In der oberen Zeile von Spalte (I) und (II) wird der erste Testfall dargestellt, bei dem diese Anforderung zum ersten Mal erfüllt wird.

- Die unteren Zeile enthält den ersten Test, bei dem beginnend diese Eigenschaft für alle weiteren Testfälle mit größerem $s(m)$ gilt.

Im Gegensatz zu Tabelle 21 muß für die Spalte (III) ein anderes Stabilitätskriterium gefunden werden. Da der Abstand zwischen relevanten und nicht-relevanten c-Werten bei $r = 10$ nicht so groß ist wie bei $r = 6$, wird bei $r = 10$ der Maßstab dort angelegt, wo der letzte relevante c-Wert mindestens 1.5-mal so groß ist wie der erste nicht-relevante.

- In der oberen Zeile der Spalte (III) wird der Testfall aufgeführt, bei dem diese Eigenschaft zum ersten Mal auftritt.
- In der unteren Zeile steht der Testfall, der der erste von allen folgenden Testfällen ist, für den diese Eigenschaft gilt.

Tabelle 25 faßt die Ergebnisse von Tabelle 24 zusammen, wobei auch hier wieder für jedes m insgesamt 6 logarithmische Mittelwerte für alle 6 Bewertungskriterien ermittelt werden.

m	(I)	(II)	(III)
7	$1.535 \cdot m^3$ $2.769 \cdot m^3$	$2.810 \cdot m^4$ $3.342 \cdot m^4$	$3.884 \cdot m^4$ $4.884 \cdot m^4$
10	$1.062 \cdot m^4$ $1.834 \cdot m^4$	$5.204 \cdot m^4$ $9.898 \cdot m^4$	$1.958 \cdot m^5$ $2.246 \cdot m^5$
11	$2.450 \cdot m^4$ $2.671 \cdot m^4$	$6.458 \cdot m^4$ $1.119 \cdot m^5$	$2.301 \cdot m^5$ $2.769 \cdot m^5$
12	$3.663 \cdot m^4$ $3.904 \cdot m^4$	$10.589 \cdot m^4$ $1.497 \cdot m^5$	$3.146 \cdot m^5$ $4.772 \cdot m^5$
13	$5.359 \cdot m^4$ $6.441 \cdot m^4$	$1.734 \cdot m^5$ $2.104 \cdot m^5$	$3.452 \cdot m^5$ $5.646 \cdot m^5$
14	$9.188 \cdot m^4$ $9.950 \cdot m^4$	$2.213 \cdot m^5$ $2.213 \cdot m^5$	$5.180 \cdot m^5$ $11.219 \cdot m^5$

Tabelle 25: Stabilitätstabelle für $r = 10$

Die folgenden Abbildungen 33, 34 und 35 geben die Tabelle 25 graphisch und in logarithmischer Darstellung wieder. In allen drei Darstellungen ist ähnlich wie bei $r = 6$ anhand der schwarzen Punkte zu erkennen, daß mit steigendem m auch der Durchschnittswert $\overline{s(m)}$ [siehe Gl.(105)] ansteigt. Auch hier ist zum Vergleich der Kreis-Graph dargestellt, der den logarithmischen Gesamtaufwand des KMA darstellt.

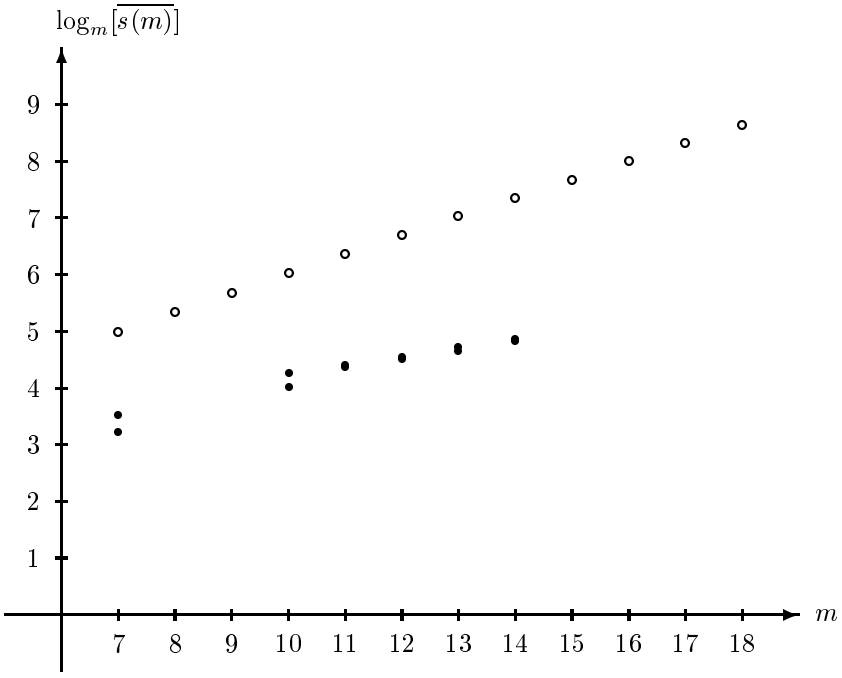

Abbildung 33: Stabilitätskurve für $r = 10$ für den Bereich (I)

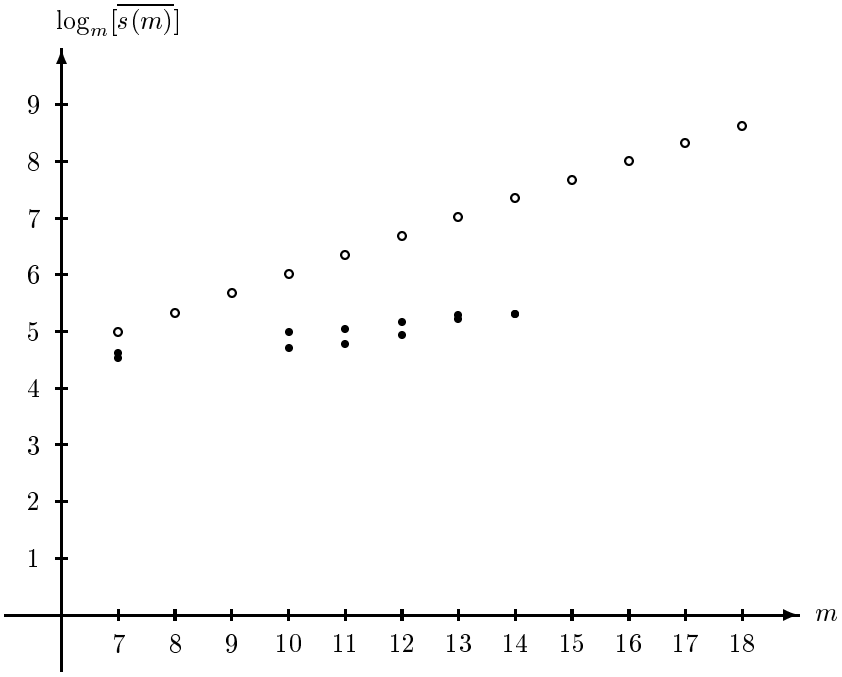

Abbildung 34: Stabilitätskurve für $r = 10$ für den Bereich (II)

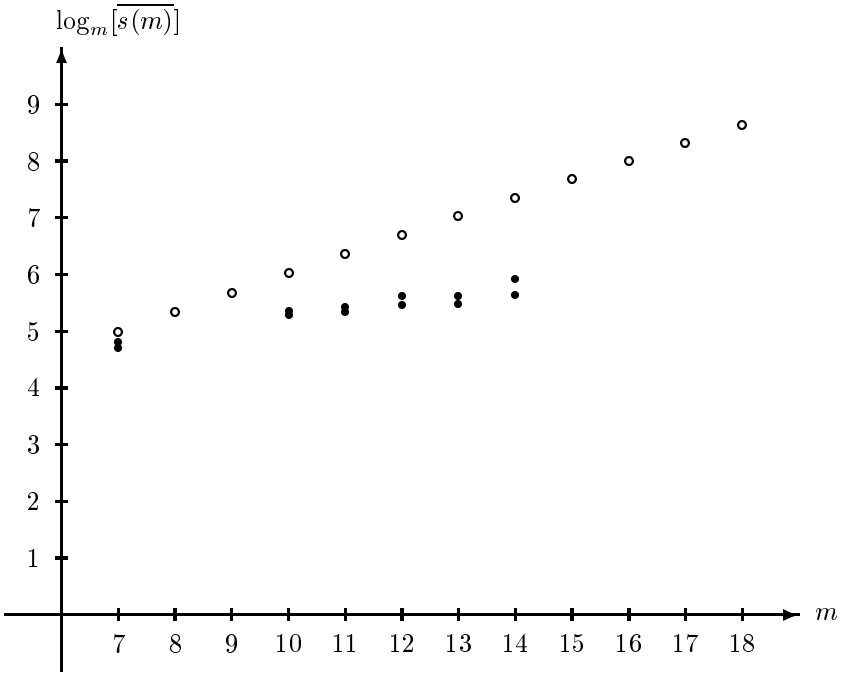

Abbildung 35: Stabilitätskurve für $r = 10$ für den Bereich (III)

6.5 Die Antwort

Die Antwort lautet NEIN.

Auf die in Kapitel 6.1 gestellt Frage, ob es für das Shor-Verfahren als reduzierten stochastischen Algorithmus einen Schwellwert $p = p_{max}$ für Gl.(104) gibt, muß aufgrund der Testergebnisse für $r = 6$ und $r = 10$ mit NEIN geantwortet werden.

Dieses Ergebnis wird in Kapitel 7 detailliert erläutert.

7 Ausblick

7.1 Bewertung des stochastischen Algorithmus

Die Zuverlässigkeit der Auswertungsergebnisse von stochastischen Algorithmen steigt bekannterweise mit der Anzahl der Versuche. Daß die Anzahl der in Kapitel 6 beschriebenen Testreihen (und auch deren Umfang) als gering bezeichnet werden muß, ist bei der Bewertung der Ergebnisse zu berücksichtigen.

- Eine Vergrößerung der Anzahl der Testreihen würde dazuführen, daß sich die Mittelwerte der 6 logarithmischen Durchschnittskurven stabilisieren. Allerdings wird hierdurch keine Veränderung dieser Mittelwerte nach unten, sondern eher nach oben erwartet, so daß darauf verzichtet wird.

- Eine Erweiterung der einzelnen Testreihen durch Anfügen weiterer Tests mit höherer EGP-Anzahl vergrößert die Wahrscheinlichkeit, daß Ausnahmefälle in diesem erweiterten EGP-Bereich gefunden werden, die die Stabilitätskriterien der Spalte (III) nicht erfüllen. Durch diese erweiterten Tests müßten bisherige Zahlenwerte in dieser Spalte nach oben korrigiert werden, so daß auch die logarithmischen Durchschnittswerte der TSA-Kurven steigen und sich dem Kreis-Graph der KMAs weiter annähern. Dies gilt insbesondere bei den Testreihen mit hohen m-Werten, für die im oberen Grenzbereich nur wenige Tests durchgeführt worden sind.

  Als Beispiel sei hier die Versuchsreihe mit $n = 133$ und $a = 37$ genannt. Da die Testsequenz am oberen Rand durch $s(m) = 9m^5$ begrenzt ist, muß der ermittelte untere Wert der Spalte (III) mit $s(m) = 9m^5$ als instabil gewertet werden. Trotzdem wurde er in der Berechnung von $\overline{s(m)}$ gemäß Gl.(105) berücksichtigt.

- Bei allen Überlegungen zur Spalte (III) sollte berücksichtigt werden, daß die Kriterien hier bereits streng gewählt sind. Eine weitere Verschärfung dieser Kriterien würde die logarithmischen Durchschnittswerte aller Erwartung nach in die Höhe treiben und die durchschnittlichen TSA-Kurven der KMA-Kreiskurve annähern.

Da für keine der sechs Kriterien ein konstanter Potenzwert p_{max} für Gl.(104) gefunden werden kann, weder für $r = 6$ noch für $r = 10$, begnügt sich diese Arbeit mit der Feststellung, daß für ansteigende n- und m-Werte alle 6 logarithmischen Durchschnittskurven des stochastischen Algorithmus flacher als der Kreis-Graph des KMA verlaufen.

7.2 Die ursprüngliche Aufgabensstellung: Faktorisieren

Um die Auswirkungen des Shor-Verfahrens als stochastischen Algorithmus auf die ursprüngliche Aufgabenstellung dieser Arbeit, das Faktorisieren der Zahl n, zu untersuchen, wird zunächst kurz zusammengefaßt, welchen Einschränkungen das komplette Shor-Verfahren (KMA) ausgesetzt ist.

- Grundlage für das Shor-Verfahren ist die Faktorisierung nach G. L. Miller (Siehe Kapitel 2.2). Hierbei wird neben der zu faktorisierenden Zahl n eine teilerfremde ganze Basiszahl a benötigt. Der Wahrscheinlichkeit P, daß

mit dieser Basiszahl eine Ordnung r berechnet wird, die ein erfolgreiches Faktorisieren von n ermöglicht, ist durch Gl.(15) eine obere Grenze gesetzt.

- Das eigentliche Shor-Verfahren (siehe Kapitel 4) kann sich bei der Berechnung der Ordnung r auf drei Verfahren stützen, die in Kapitel 4.2.4 beschrieben werden. Bei Variante 3 ist zu beachten, daß aus der Anzahl r an c-Werten, die Gl.(77) erfüllen, nur eine limitierte Anzahl von $\varphi(r)$ Werten [siehe Gl.(3)] über Kettenbruch-Zerlegung zur korrekten Ordnung r führen.

Neben diesen Einschränkungen existieren beim kompletten Shor-Verfahren aber auch folgende Sicherheiten:

- „Relevante“ und „nicht-relevante“ c-Werte werden eindeutig über Gl.(78) definiert. Dabei liegt die maximale Argument-Wahrscheinlichkeit immer bei $c = 0$.
- Unter den r „relevanten“ c-Werten führt der kleinste von Null verschiedene c-Wert immer zur korrekten Berechnung der Ordnung r.

Der reduzierte Shor-Algorithmus (TSA) besitzt zwar auch die für das Shor-Verfahren gültigen Einschränkungen, allerdings nicht die oben erwähnten Sicherheiten. Bei der Ergebnis-Bewertung ergeben sich somit folgende Konsequenzen:

- Eine definierte Trennung zwischen relevanten und nicht-relevanten c-Werten, so wie sie Gl.(77) vorgibt, existiert hier nicht. Die Grenze ist unbekannt und daher fließend.
- Wird die Anzahl $s(m)$ der EGPn gemäß den Werten aus der Spalte (III) der Tabellen 22 oder 25 gewählt, dann kann mit hoher Wahrscheinlichkeit davon ausgegangen werden, daß sich die r relevanten Argument-Wahrscheinlichkeiten in der Tabelle der besten c-Werte ganz oben befinden. Daß es sich hierbei tatsächlich um r Werte handelt, kann nicht garantiert werden.
- Sind die relevanten c-Werte ganz oben, dann kann $c = 0$ ignoriert werden. Mit hoher Wahrscheinlichkeit fällt auch $c = \frac{q}{2}$ nicht in Betracht. Es gibt aber auch Beispiele, wie z.B. bei $n = 15$ und $a = 4$, für die die korrekte Ordnung $r = 2$ mit Hilfe von $c = \frac{q}{2}$ ermittelt werden muß.

Damit werden die Unterschiede zwischen KMA und TSA über die Eigenschaften in Kapitel 4.2.4 signifikant:

- Beim KMA genügt die Anzahl der relevanten c-Werte zum Ermitteln der Ordnung r, auch wenn Shor die Kettenbruch-Zerlegung ursprünglich als weitere Maßnahme vorsah.
- Beim TSA ist die Anzahl der relevanten c-Werte nur ein Gütekriterium für die Bewertung des Polynoms $s(m)$. Da die tatsächliche Anzahl der relevanten Werte, die in der Hierarchie ganz oben stehen, nicht bekannt ist, muß die Kettenbruch-Zerlegung als weiterer Schritt zur Berechnung der Ordnung r mitberücksichtigt werden.

Hierdurch ergeben sich folgende Richtlinien zum Faktorisieren der Zahl n über den TSA:

- Man wählt ein Polynom $s(m)$, bei dem die Wahrscheinlichkeit groß ist, daß alle relevanten c-Werte an erster Stelle stehen. Dabei eignet sich ein $s(m)$, das größer als in Spalte (II) und kleiner als in Spalte (III) ist.

- Sind die c-Werte nach Größe der Argument-Wahrscheinlichkeit sortiert, unterzieht man die oberen c-Werte der Reihe nach einer Kettenbruch-Zerlegung und zwar solange, bis die korrekte Ordnung r ermittelt wird, die wiederum zu einer korrekten Faktorisierung der Zahl n führt. Da die Computational Resources der Kettenbruch-Zerlegung mit $O(m^3)$ im polynomialen Rahmen liegen, ist diese Vorgehensweise sinnvoll.

Zu diesen Überlegungen ein Beispiel:
In Tabelle 21 ist für $n = 91$ und $a = 4$ die Stabilität der relevanten c-Werte gemäß Spalte (III) für $s(m) = 7m^5$ gegeben. Für diese Anzahl werden im 4.Test aus Kapitel 6.2 gemäß Tabelle 18 der Reihe nach die Werte $c = 8192$, 0, 5461, 10923, 13653 und 2731 ausgegeben. Man erkennt, daß die eigentlichen Lösungeswerte für die Ordnung r erst an Position 5 und 6 zu finden sind.

7.3 Fazit: Ergebnis entspricht der Erwartung

Daß das Shor-Verfahren als stochastischer Algorithmus nicht zur erwünschten polynomialen Reduktion der Rechen-Operationen auf einem klassischen Rechner führt, muß als positiv gewertet werden.

Begründung:
Wäre es möglich, die Zahl n mit polynomialen Aufwand zu faktorisieren, dann könnten Computer-Hecker ohne große Mühe verschlüsselte Informationen entschlüsseln (siehe RSA-Algorithmus in Kapitel 8.2).

Ob es andere Methoden gibt, die eine Reduktion der Operationen bewirkt, ist nicht bekannt. Mit dem Verfahren der phasenbehafteten Messungen ist dies nicht möglich.

8 Anhang

8.1 Gehört nicht zum Thema: Eine Feststellung

In Tabelle 9 von Kapitel 2.2.2 wurden einige Beispiele für das Verfahren von G. L. Miller aufgeführt, die das Gleichheitszeichen innerhalb des „$\geq$“ von Gl.(15) belegen. Bei diesen Beispielen ist aufgefallen, daß die zu faktorisierenden Zahlen n nur aus Primzahlen der Form $p = 3 + 4 \cdot l$ oder deren Potenzprodukt p^{α} bestehen.

Zur Darstellung von Gl.(15) gibt es unterschiedliche Auffassungen. [NI] und [SH](1) behaupten im Gegensatz zu [SH](2), daß im Nenner „2^K“ anstelle von „2^{K-1}“ stehen muß (Siehe [1]). Daher soll in diesem Kapitel für das Verfahren von G. L. Miller bestätigt werden, daß für Zahlen

$$n = n_1 \cdot n_2 \cdot \ldots \cdot n_K \qquad \text{mit} \qquad n_k = p_k^{\alpha_k}$$

folgende Wahrscheinlichkeit gilt

$$P[\{r \text{ ist gerade}\} \wedge \{a^{\frac{r}{2}} \neq -1 \pmod{n}\}] \leq 1 - \tfrac{1}{2^{K-1}}$$

Um diese Aussage nachzuweisen, muß auf den Anhang 4 von [NI] zurückgegriffen werden. Aus diesem Anhang werden hier die wesentlichen Kernaussagen zusammengefaßt.

1. Gegeben sei das Verfahren nach G. L. Miller mit den bekannten Koeffizienten a, n, r und $a^r = 1 \pmod{n}$. Hierbei sei $a \in \mathbb{Z}_n^*$, wobei $\mathbb{Z}_n^*$ gemäß Gl.(1) definiert ist. Dann existiert ein maximales δ mit $2^{\delta} \mid r$, d.h. „2^{δ} teilt r“. (Siehe [2].)

2. Mit dem Euler-Theorem gilt $a^{\varphi(n)} = 1 \pmod{n}$, wobei $\varphi(n)$ gemäß Gl.(3) definiert ist. Außerdem gilt: $r \mid \varphi(n)$.

3. Ist p Primzahl und φ positiv und ganzzahlig , dann ist $\varphi(p^{\alpha}) = (p-1)p^{\alpha-1}$.

4. Zu jedem a werden Werte a_k über das „Chinesische Resttheorem“ mittels $a = a_k \pmod{n_k}$ ermittelt. Es gilt $a_k \in \mathbb{Z}_{n_k}^*$ und die Abbildung $a \Leftrightarrow (a_1, a_2, \ldots, a_K)$ ist eine Äquivalenzrelation.

5. Mittels ${a_k}^{r_k} = 1 \pmod{n_k}$ wird die Ordnung r_k von a_k modulo n_k ermittelt. Für alle r_k existiert ein maximales δ_k mit $2^{\delta_k} \mid r_k$.

6. Ist p Primzahl und $\delta = \delta_{max}$ der höchste Exponent, für den $2^{\delta_{max}} \mid \varphi(p)$ gilt, dann teilt $2^{\delta_{max}}$ mit der Wahrscheinlichkeit $\frac{1}{2}$ die Ordnung r von a modulo n, wobei $a \in \mathbb{Z}_n^*$ zufällig gewählt wurde.

7. Für das Verfahren von G. L. Miller wird über $a^r = 1 \pmod{n}$ ein ungerades r ermittelt, genau dann wenn $\delta = \delta_1 = \delta_2 = \cdots = \delta_K = 0$ ist.

[1] In Abweichung von [NI] wird der Wert m in dieser Arbeit durch K dargestellt, da m schon für die Bit-Dimension des Shor-Verfahrens verwendet wird.

[2] In Abweichung von [NI] wird der Wert d in dieser Arbeit durch δ dargestellt, da d schon in der Kettenbruch-Zerlegung verwendet wird.

8. Für das Verfahren von G. L. Miller wird $a^{\frac{r}{2}} = -1 \pmod{n}$, genau dann wenn $\delta = \delta_1 = \delta_2 = \cdots = \delta_K \geq 1$ ist.

Mit Tabelle 26 und 27 folgen zwei Beispiele ($n = 15 = 3 \cdot 5$ und $n = 21 = 3 \cdot 7$), für die die Werte n_k, a_k, r_k und δ_k gesondert berechnet werden.

a	a_1	a_2	r	r_1	r_2	δ	δ_1	δ_2	Ergebnis
1	1	1	1	1	1	0	0	0	r ungerade
2	2	2	4	2	4	2	1	2	$3 \cdot 5 = 15$
4	1	4	2	1	2	1	0	1	$3 \cdot 5 = 15$
7	1	2	4	1	4	2	0	2	$3 \cdot 5 = 15$
8	2	3	4	2	4	2	1	2	$3 \cdot 5 = 15$
11	2	1	2	2	1	1	1	0	$5 \cdot 3 = 15$
13	1	3	4	1	4	2	0	2	$3 \cdot 5 = 15$
14	2	4	2	2	2	1	1	1	$a^{\frac{r}{2}} = -1$

Tabelle 26: Zerlegung von $n = 15$ in $n_1 = 3$ und $n_2 = 5$

a	a_1	a_2	r	r_1	r_2	δ	δ_1	δ_2	Ergebnis
1	1	1	1	1	1	0	0	0	r ungerade
2	2	2	6	2	3	1	1	0	$7 \cdot 3 = 21$
4	1	4	3	1	3	0	0	0	r ungerade
5	2	5	6	2	6	1	1	1	$a^{\frac{r}{2}} = -1$
8	2	1	2	2	1	1	1	0	$7 \cdot 3 = 21$
10	1	3	6	1	6	1	0	1	$3 \cdot 7 = 21$
11	2	4	6	2	3	1	1	0	$7 \cdot 3 = 21$
13	1	6	2	1	2	1	0	1	$3 \cdot 7 = 21$
16	1	2	3	1	3	0	0	0	r ungerade
17	2	3	6	2	6	1	1	1	$a^{\frac{r}{2}} = -1$
19	1	5	6	1	6	1	0	1	$3 \cdot 7 = 21$
20	2	6	2	2	2	1	1	1	$a^{\frac{r}{2}} = -1$

Tabelle 27: Zerlegung von $n = 21$ in $n_1 = 3$ und $n_2 = 7$

In Tabelle 27, in der die Zahl 21 nur aus Primfaktoren der Form $p = 3 + 4 \cdot l$ besteht, fällt auf, daß δ, δ_1 und δ_2 nur aus den Werten 0 und 1 bestehen, während in Tabelle 26 für die Zahl 15 auch höhere δ-Werte errechnet werden. Diese δ_k-Werte ergeben sich unmittelbar aus der Ordnung r_k von a_k modulo n_k, also direkt aus n_k und nicht aus n. Daher ist ein weitere Untersuchung der Primfaktorterme $n_k = p_k^{\alpha_k}$ hilfreich.

Hierzu werden in Tabelle 28 zu den Zahlen 7, 9 und 13 alle a-Werte mit $a \in \mathbb{Z}_n^*$, die Ordnungen r von a modulo n und die zu r zugehörigen δ-Werte ermittelt.

n	a	r	δ	n	a	r	δ
7	1	1	0	13	1	1	0
7	2	3	0	13	2	12	2
7	3	6	1	13	3	3	0
7	4	3	0	13	4	6	1
7	5	6	1	13	5	4	2
7	6	2	1	13	6	12	2
9	1	1	0	13	7	12	2
9	2	6	1	13	8	4	2
9	4	3	0	13	9	3	0
9	5	6	1	13	10	6	1
9	7	3	0	13	11	12	2
9	8	2	1	13	12	2	1

Tabelle 28: Zerlegung von $n = 7$, $n = 9$ und $n = 13$

Für die Auswertung von Tabelle 28 kann Kernaussage 6 aus Anhang 4 von [NI] herangezogen werden.

- Die Zahlen $n = 7$ und $n = 9$ enthalten Grundprimzahlen vom Typ $p = 3 + 4 \cdot l$ und haben daher die Eigenschaft:

  $$2^{\delta_{max}} \mid r \mid \varphi(p^\alpha) = (p-1)p^{\alpha-1} = (2+4l)(3+4l)^{\alpha-1} \quad \Rightarrow \quad \delta_{max} = 1$$

  Das bedeutet, daß mit Wahrscheinlichkeit $\frac{1}{2}$ der Wert $\delta_{max} = 1$ derjenige δ-Wert ist, der in Tabelle 28 für $n = 7$ und $n = 9$ zur Hälfte vertreten ist. Diese Aussage kann bestätigt werden. Von 6 verschiedenen a-Werten ist $\delta = 1$ dreimal vertreten.

- Die Zahl $n = 13$ ist Primzahl vom Typ $p = 1 + 4 \cdot l$ und hat daher die Eigenschaft:

  $$2^{\delta_{max}} \mid r \mid \varphi(p^\alpha) = (p-1)p^{\alpha-1} = 4l \cdot (1+4l)^{\alpha-1} \quad \Rightarrow \quad \delta_{max} = 2$$

  Das bedeutet, daß der Wert $\delta_{max} = 2$ in Tabelle 28 für $n = 13$ zur Hälfe vertreten ist. Diese Aussage kann bestätigt werden. Bei 12 verschiedenen a-Werten kommt $\delta = 2$ sechsmal vor.

- Die Zahl $n = 193$ ist Primzahl vom Typ $p = 1 + 64 \cdot l$ und hat daher die Eigenschaft:

  $$2^{\delta_{max}} \mid r \mid \varphi(p^\alpha) = (p-1)p^{\alpha-1} = 64l \cdot (1+64l)^{\alpha-1} \quad \Rightarrow \quad \delta_{max} = 6$$

Das bedeutet, daß der Wert $\delta_{max} = 6$ im Fall $n = 193$ zur Hälfe vertreten ist. Diese Aussage kann bestätigt werden. Bei 192 verschiedenen a-Werten kommt $\delta = 6$ insgesamt 96-mal vor.

Die Kernaussage 6 aus Anhang 4 von [NI] beinhaltet aber auch noch eine weitere Eigenschaft: Wenn die Hälfte aller δ-Werte durch $\delta = \delta_{max}$ vertreten ist, dann ist die andere Hälfte durch die restlichen δ-Werte mit $\delta < \delta_{max}$ vertreten.

- Für $n = 7$ und $n = 9$ gilt $\delta_{max} = 1$. D.h. die andere Hälfte der δ-Werte wird durch $\delta = 0$ belegt. Dies kann durch Tabelle 28 bestätigt werden. Der Wert $\delta = 0$ tritt bei beiden n-Werten dreimal auf.
  Allgemein gilt für Primzahlen vom Typ $p = 3 + 4 \cdot l$ oder für deren Potenzprodukte p^α, daß die Anzahl der $\delta = 0$ und $\delta = 1$ sich im Verhältnis $\frac{1}{2} : \frac{1}{2}$ auf alle $a_k \in \mathbb{Z}^*_{p^\alpha}$ aufteilt.
- Für $n = 13$ gilt, daß neben dem Wert $\delta = 2$, der sechsmal vorkommt, die Werte $\delta = 0$ und $\delta = 1$ jeweils dreimal vorkommen.
- Für $n = 193$ gilt, daß neben dem Wert $\delta = 6$, der 96-mal vorkommt, 48-mal der Wert $\delta = 5$, 24-mal der Wert $\delta = 4$, 12-mal der Wert $\delta = 3$, sechsmal der Wert $\delta = 2$ und jeweils dreimal die Werte $\delta = 0$ und $\delta = 1$ vorkommen.

Damit ist die Wahrscheinlichkeit, daß δ einen festen Wert $\delta_0 \leq \delta_{max}$ annimmt, gegeben durch:

$$P\{\delta = \delta_0\} \leq \tfrac{1}{2} \quad \text{für alle} \quad 0 \leq \delta_0 \leq \delta_{max}$$

Fazit:

Folgende Schlüsse können aus den bisherigen Erkenntnissen gezogen werden:

- Produktzahlen n werden in Faktoren $n_k = p_k^{\alpha_k}$ zerlegt. Für jedes $a \in \mathbb{Z}^*_n$ existiert in Abhängigkeit von jedem n_k ein a_k, ein r_k und ein δ_k. Jedes Zahlenquartett der Form $(n_k, a_k, r_k, \delta_k)$ hängt damit zwar von a ab, allerdings sind $(n_j, a_j, r_j, \delta_j)$ und $(n_k, a_k, r_k, \delta_k)$ für $j \neq k$ unkorreliert.
- Erfolg oder Nicht-Erfolg der Faktorisierung von n mit dem Verfahren von G. L. Miller hängen von der Wahl des a-Werts ab, über den die Werte δ_1, δ_2, ..., δ_K ermittelt werden.
- Für jedes δ_j gilt $P\{\delta_j = \delta\} \leq \frac{1}{2}$. Da die δ_j untereinander unkorreliert sind, kann gemäß Kernaussage 7 und 8 aus der Zusammenfassung von Anhang 4 von [NI] gefolgert werden, daß

$$P\{r \text{ ungerade } \vee\ a^{\frac{r}{2}} = -1 \pmod n\} =$$
$$P\{\delta_1 = \delta_2 = \ldots = \delta_K\} =$$
$$\sum_{\delta=0}^{\delta_{max}} P\{\delta = \delta_1 = \delta_2 = \ldots = \delta_K\} =$$
$$\sum_{\delta=0}^{\delta_{max}} P\{\delta_1 = \delta\} \cdot P\{\delta_2 = \delta_1\} P\{\delta_3 = \delta_1\} \ldots P\{\delta_K = \delta_1\} \leq$$
$$\sum_{\delta=0}^{\delta_{max}} P\{\delta_1 = \delta\} \cdot (\tfrac{1}{2})^{K-1} = 1 \cdot (\tfrac{1}{2})^{K-1} = \tfrac{1}{2^{K-1}}$$

Damit ist Gl.(15) nachgewiesen (q.e.d.).

Nachtrag:
Besteht n nur aus Primzahlen vom Typ $p_k = 3 + 4 \cdot l$ oder deren Potenz $p_k^{\alpha_k}$, dann ist bekanntlich das zugehörige δ_k im Verhältnis $\frac{1}{2} : \frac{1}{2}$ entweder 0 oder 1. Dadurch vereinfacht sich die Gesamtwahrscheinlichkeit.

- Hier gilt $P\{\delta_k = 0\} = P\{\delta_k = 1\} = \frac{1}{2}$. Damit kann gemäß Kernaussage 7 und 8 aus der Zusammenfassung von Anhang 4 von [NI] gefolgert werden, daß

$$\begin{aligned} P\{r \text{ ungerade}\} &= P\{\delta_1 = \delta_2 = \ldots = \delta_K = 0\} \\ &= P\{\delta_1 = 0\}P\{\delta_2 = 0\}\ldots P\{\delta_K = 0\} \\ &= (\tfrac{1}{2})^K \end{aligned}$$

$$\begin{aligned} P\{a^{\frac{r}{2}} = -1 \pmod n\} &= P\{\delta_1 = \delta_2 = \ldots = \delta_K = 1\} \\ &= P\{\delta_1 = 1\}P\{\delta_2 = 1\}\ldots P\{\delta_K = 1\} \\ &= (\tfrac{1}{2})^K \end{aligned}$$

- Die Summierung beider Wahrscheinlichkeiten ergibt damit:

$$P\{r \text{ ungerade } \vee \ a^{\frac{r}{2}} = -1 \pmod n\} \ = \ 2 \cdot (\tfrac{1}{2})^K \ = \ \tfrac{1}{2^{K-1}}$$

Damit ist das Gleichheitszeichen in Gl.(15) nachgewiesen (q.e.d.).

8.2 Der RSA-Algorithmus

Der RSA-Algorithmus wurde von den beiden Amerikanern Ronald L. Rivest, Leonard M. Adleman und dem Israeli Adi Shamir im Jahr 1977 entwickelt. Der Name dieses Verfahrens ergibt sich aus den Anfangsbuchstaben der Namen dieser drei Wissenschaftler.

Bei diesem Verfahren wird ein „Schlüsselpaar“ (n, e) veröffentlicht, das folgende Eigenschaften hat:

- $n = p \cdot q$ ist eine natürliche Zahl, wobei p und q große Primzahlen sind.
- e ist eine natürliche Zahl, die kleiner als n ist, und für die $\text{ggT}[e, \varphi(n)] = 1$ gilt. Dabei ist $\varphi(n) = (p-1)(q-1)$ die Euler-Funktion gemäß Gl.(3).

Für den weiteren Verlauf des Verfahrens muß eine Zahl d gefunden werden, die folgendes Merkmal aufweist:

- d ist eine natürliche Zahl, die kleiner als n ist, und für die $\text{ggT}[de, \varphi(n)] = 1$ oder $de = 1[\text{mod } \varphi(n)]$ gilt.

Mit diesen Eigenschaften läßt sich nun das Prinzip des RSA-Algorithmus erklären.

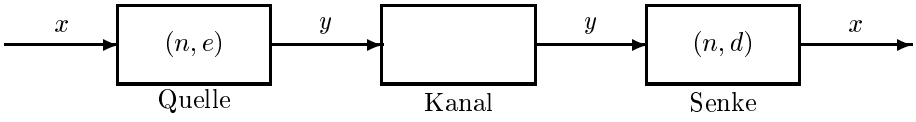

Abbildung 36: RSA-Algorithmus im Netzwerk

In Abbildung 36 soll eine Zahl x mit $x \in \mathbb{Z}_n^*$ gemäß Gl.(2) verschlüsselt werden:

- In der Quelle wird die Zahl x mit Hilfe des Zahlenpaares (n, e) über die Gleichung

$$y = x^e \pmod{n} \tag{106}$$

  verschlüsselt. Gl.(106) hat den Vorteil, daß eine direkte Rückwärtsherleitung von x aus y nicht möglich ist.

- Die verschlüsselte Zahl y wird durch den Kanal geleitet.

- In der Senke wird die Zahl x aus y mit Hilfe des Zahlenpaares (n, d) über die Gleichung

$$x = y^d \pmod{n} \tag{107}$$

  wiedergewonnen.

Die Herleitung von Gl.(107) aus Gl.(106) ergibt sich aus dem Satz von Euler gemäß Gl.(20). Angewandt auf $x \in \mathbb{Z}_n^*$ gilt:

$$x^{\varphi(n)} = 1 \pmod{n}$$

$$\Longrightarrow x^{k\varphi(n)} = 1 \pmod{n} \quad \text{für alle} \quad k \geq 1$$

$$\Longrightarrow x^{1+k\varphi(n)} = x \pmod{n}$$

Mit der Forderung, daß $de = 1[\text{mod } \varphi(n)]$ oder $de = 1 + k\varphi(n)$ ist, erhält man:

$$y^d = (x^e)^d = x^{de} = x^{1+k\varphi(n)} = x \pmod{n}$$

Da das Zahlenpaar (n, e) öffentlich ist, ergeben sich für den RSA-Algorithmus an der Senke zwei Aufgaben:

- Der Wert n muß in seine Primfaktoren p und q zerlegt werden. Mit p und q läßt sich $\varphi(n)$ berechnen.
- Die Inverse d von e modulo $\varphi(n)$ muß berechnet werden:

  $$d = e^{-1}[\text{mod } \varphi(n)]$$

  Eine elegante Berechnung erfolgt über die Euclid-Sätze zum *größten gemeinsamen Teiler* (ggT). Wird $\varphi(n)$ durch L Bits dargestellt, dann werden $O(L^3)$ Rechenoperationen benötigt. Eine ausführliche Beschreibung dieser Berechnung liefert Anhang 4 (Zahlentheorie) aus [NI].

Der RSA-Algorithmus läßt sich in folgender Weise interpretieren:

- Die Zahl x kann Zahlenwert eines einzelnen ASCII-Zeichens oder einer Kette von ASCII-Zeichen sein. Die Umwandlung in y verändert dieses Zeichen und macht es u.U. unlesbar.
- Wird ein lesbarer Text, gemäß Gl.(106) verschlüsselt, dann ist das verschlüsselte Ergebnis u.U. nicht lesbar.
- Erst die Dekodierung über Gl.(107) liefert wieder den lesbaren Ursprungs-Text.

Da das Zahlenpaar (n, e) veröffentlicht wird, müssen die nicht-öffentlichen Primzahlen p und q sehr groß sind. Durch die Größe dieser Zahlen ist es für gewöhnliche Computer-Hecker schwierig, die Primzahlen ausfindig zu machen und den verschlüsselten Code zu „knacken“.

9 Literaturverzeichnis

Literatur

[BU] P. Bundschuh: *Einführung in die Zahlentheorie.* Springer-Verlag Berlin Heidelberg 1988. ISBN 3-540-15305-5 und 0-387-15305-5.

[CR] B. Crell, A. Uhlmann: *Einführung in Grundlagen und Protokolle der Quanteninformatik.* Institut für Theoretische Physik, Universität Leipzig, Datum ca. 1998.

[GE] W. Gellert, M. Hellwich, H. Kästner, H. Küstner: *Kleine Enzyklopädie Mathematik.* VEB Bibliographisches Institut Leipzig, 1979.

[KN] D. Knuth: *The Art of Computer Programming. Volume 1 & 2.* Addison-Wesley Publishing Company, 1968 & 1969.

[NI] M. A. Nielsen, I. L. Chung: *Quantum Computation and Quantum Information.* Cambridge University Press 2000, ISBN 0 521 63235 8 hardback, ISBN 0 521 63503 9 paperback.

[SC] S. Schäffler: *Quantum Computation.* Vorlesung an der Fakultät für Elektrotechnik und Informationstechnik der Universität der Bundeswehr München, Frühjahr 2002.

[SH] (1) P. W. Shor: *Algorithms for Quantum Computation: Discrete Logarithms and Factoring.* 35th Annual Symposium on Foundations of Computer Science, Santa Fe, pages 124-134, IEEE Computer Society Press, 1994. (2) P. W. Shor: *Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer.* SIAM J. Comput., Vol. 5, pages 1448-1509, 1997.